ABSTRACT

Manually detection of plant disease is very difficult, costly and time consuming. Inaccuracy in identification of plant disease may cause the great loss in the production and economical value of market. Detection of plant disease may require huge amount of knowledge and work on plant disease. Therefore, we use the image processing technique for detection of plant's leaf detection. Initially, user i.e. farmer clicks a picture of the affected leaf and uploads the picture it to the server via the android application. The proposed system aims to overcome the pitfalls of the existing system and provides features such as detection of plant disease, feature extraction, analysis of data. Then the result consisting of the disease name with the accuracy is retrieved using CNN algorithm.

Keywords: Leaf Disease, CNN.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1. Overview

Nepal is a developing country and agriculture is the backbone for the earlier development of the country. Nepali Economy is highly dependent on agriculture. Around 60% of people are directly engaged on farming [1]. Now a day's technology plays a vital role in all fields but we are still using the old methodologies in agriculture. Identifying plant disease is high yield, money, time and quality product. Identifying the condition of plant is very important in successful cultivation. An experienced person does identification manually but due to the environmental condition, prediction is very tough. Thus, we can use image-processing technique for the detection of plant diseases.

Plant Tech Solutions is a mobile application that detects the disease of the plant and gives the information about the plant to the user. System architecture and algorithms used in each stage is described in this proposal. The "Plant Tech Solutions" is an application of disease detection. Conventional neural network algorithm is used to develop this project. Generally, the system can detect the leaf disease of the plant but here we have used the Paddy leaf for the identification of disease.

1.2. Statement of Problem

Farmer's economic growth depends on the quality of food that they produce. In an agriculture, detection of plant disease is very important. However, it is very difficult to identify the disease manually. Plants are highly prone to disease that affect the growth of plant which turns affect the ecology of the farmer [1]. Plant disease may affect the different part of the plants like leaves, stems, seeds, etc. Manually detection of plant disease is very difficult, costly and time consuming. Inaccuracy in identification of plant disease may cause the great loss in the production and economical value of market. Hence, it is required to detect the disease in computational method by the classification of leaf automatically.

1.3. Objectives

The main objectives of the project are:

- To detect the disease of Paddy leaf.
- To display accurate result about the leaf disease to the user.

1.4. Scope and Limitations

The scopes and limitations of the project are:

- The application will be able to provide efficiency in detecting the Paddy leaf disease automatically other than manual detection.
- The application will not be only limited on detection of Paddy leaf, the scope can be extended to the detection of stem, seed, root, etc. Not only this, but also this procedure can be used to detect other plants as well.

1.5. Report Organization

Our report is organized into 5 chapters:

Chapter 1: Introduction

In this section the brief introduction of our project, statement of the problem and its objectives are discussed.

Chapter 2: Requirement analysis

The previous work related to our projects were studied and different feasibility analysis are summarized in this section.

Chapter 3: System Design

In this section, we have designed the structuring system requirement like activity diagram, sequence diagram, database, etc.

Chapter 4: Implementation and Testing

In this section, various implementation method and tools are described. This part also contains the description of various testing and results we got after performing.

Chapter 5: Conclusion and Enhancement

CHAPTER 2: REQUIREMENT ANALYSIS

2.1. Literature Review

Plant diseases have turned into a dilemma as it can cause significant reduction in both quality and quantity of agricultural products [1]. Finding whether the plant leaf has disease or not is common issue to farmer. Farmers used to predict the plant disease manually with their naked eyes. However, some techniques were implemented but are limited in practice. Also, there is an application called "E-agrovet" which detects whether the plant is healthy or not. It also shows the pH value, temperature, water level and soil moisture.

The project called image-processing technique for Leaf & Stem disease detection used a set of leaf images from Jordan's AI Ghor Area. The five plant diseases namely: Early scorch, ashen mold, late scorch, Cottony mold and tiny white less is tested by image processing technique. In this technique at starting, image acquisition is obtained and then K-Means clustering method is used for segmentation. After that in feature extraction, CCM (Color Co-occurrence Method) is used for texture for neural network in classification of plant diseases. Result of this image processing technique shows accurate detection and classification of plant disease [2].

A web enabled disease detection system (WEDDS) based on compressed sensing (CS) is proposed to detect and classify the diseases in leaves [3]. The diseased leaf is segmented using statistical based on thresholding. The analysis and classification is done using support vector machine classifier. The performance of the proposed WEDDS has been evaluated in terms of accuracy and also evaluated experimentally using Raspberry pi 3 board.

[3] In Vision Based Plant Leaf Disease Detection on The Color Segmentation through Fire Bird V Robot [4], the detection of plant leaf disease is based on the texture of the leaf. The processing scheme consists of image acquisition through digital camera connecting to the Fire Bird V Robot [4]. It uses the image pre-processing techniques that includes image enhancement and image segmentation where the affected leaf are segmented feature extraction and classification.

The main difference between these projects and our plant tech solutions is that it has high Image transformation in frequency domain, which gives better classification. Specifically, we are working on paddy Leaf for detecting diseases. The Convolutional Neural Network uses the different layers i.e. input layer, output layer and other hidden layers. The image is passed through a series of convolutional, nonlinear, polling layers and fully connected layers, and then generated the output. Result of our Plant tech solutions technique will show accurate detection and classification of plant diseases with high accuracy.

2.2. Requirement Analysis

2.2.1 Functional Requirements

Functional requirements identify the provision of the system and the system's reaction to the certain output and how the system should behave in day to day basis. This system is focused on capturing the image of paddy leaf, classify and detect the disease.

The functional requirements of "Plant Tech Solutions" includes the following tasks: -

- System should upload the image of plant leaf and classify them in no time.
- System should not take long to classify and test the image.
- System should detect the disease correctly.
- The result must be display to the user after detection.

USE CASE DIAGRAM

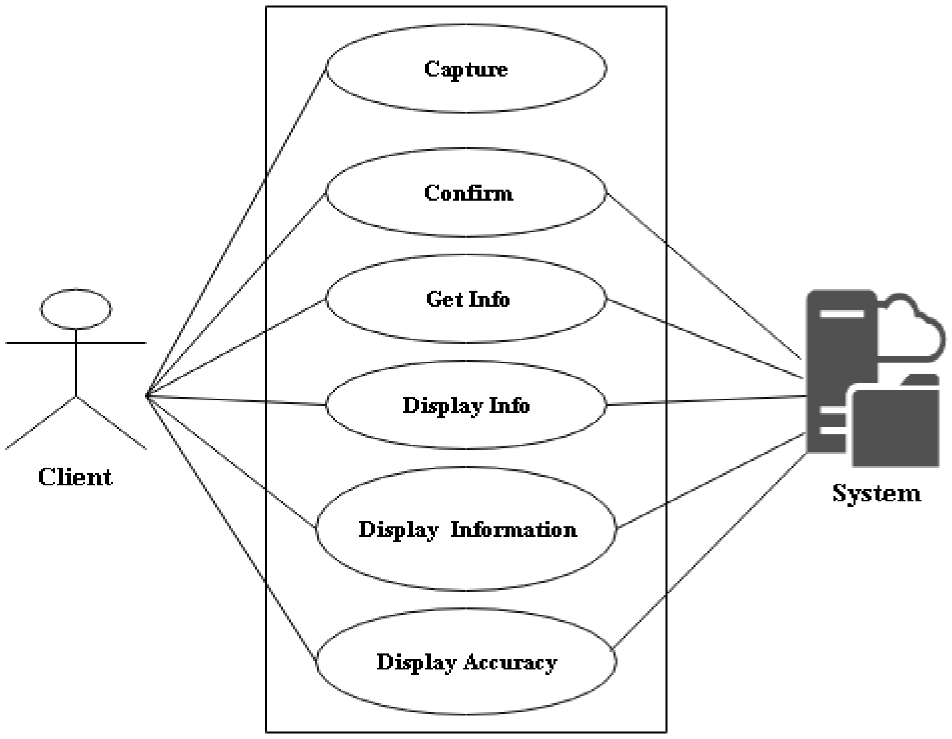


Figure 2.1: Use Case Diagram

2.2.2 Non-Functional Requirements

- **Usability:** The system needs to be usable by every user.
- **Maintenance:** The system needs to be maintainable.
- **Extendable and Scalability:** The system is extendable and scalable for future enhancements.

2.3. Feasibility Analysis

2.3.1. Economic Feasibility

The purpose of economic feasibility assessment is to determine the positive economic benefits to the organization that proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefit analysis. The probable cost at the first step of the project includes all the hardware and software costs, which are minimal for our project. This may increase as we scale the app for more users and more UX elements. Hence, we can say that this project has been found to be economically feasible.

2.3.2. Technical Feasibility

This assessment is based on an outline design of system requirements, to determine whether the company/team has the technical expertise to handle completion of the project. It is an evaluation of the hardware and software and how it meets the need of proposed system.

As per the requirement analysis, it was observed that technical expertise in the programming language Python, Machine Learning, JAVA and in the markup language XML is involved in the project. Familiarity with the application development in the Android environment is necessary. Owing to our familiarity with OOP based programming language and markup language we found the project will be technically feasible.

2.3.3. Operational Feasibility

Operational Feasibility is the measure of how a proposed system solves a problem and takes advantages of the opportunities identified during scope definitions and how it satisfies the requirements identified in the requirement analysis phase of the project development. To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters such as reliability, maintainability, usability, productivity, sustainability and others. This project “Plant Tech Solutions” an android application makes all of these parameters. Hence, we will be able to say that our project is operationally feasible.

2.3.4. Schedule Feasibility

The team members will be responsible for different aspects of the system development. Project is intended to be completed within 3 months, including our internal deadlines.

CHAPTER 3: SYSTEM DESIGN

3.1. System Architecture

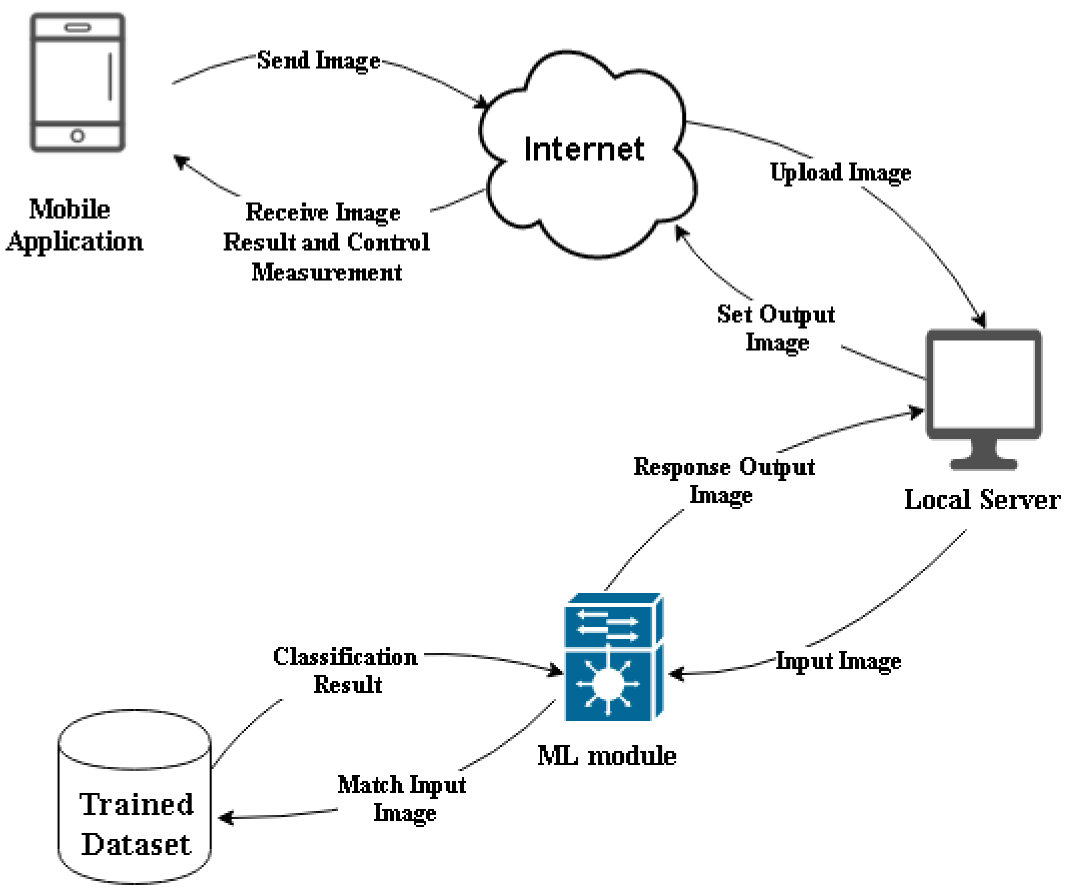
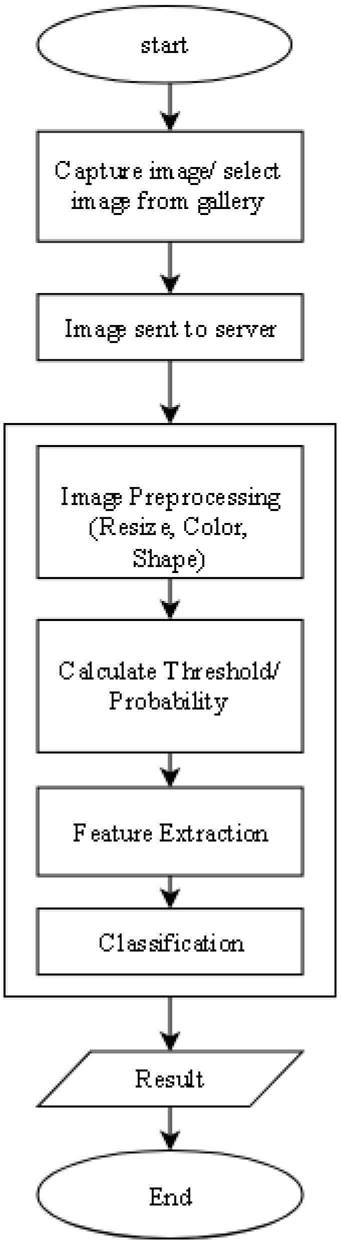


Figure 3.1: System Architecture

3.2. Flow Diagram of a System



start
Capture image/ select image from gallery
Image sent to server
Image Preprocessing (Resize, Color, Shape)
Calculate Threshold/ Probability
Feature Extraction
Classification
Result
End

Figure 3.2: Flow Diagram of a System

3.3. Sequence Diagram

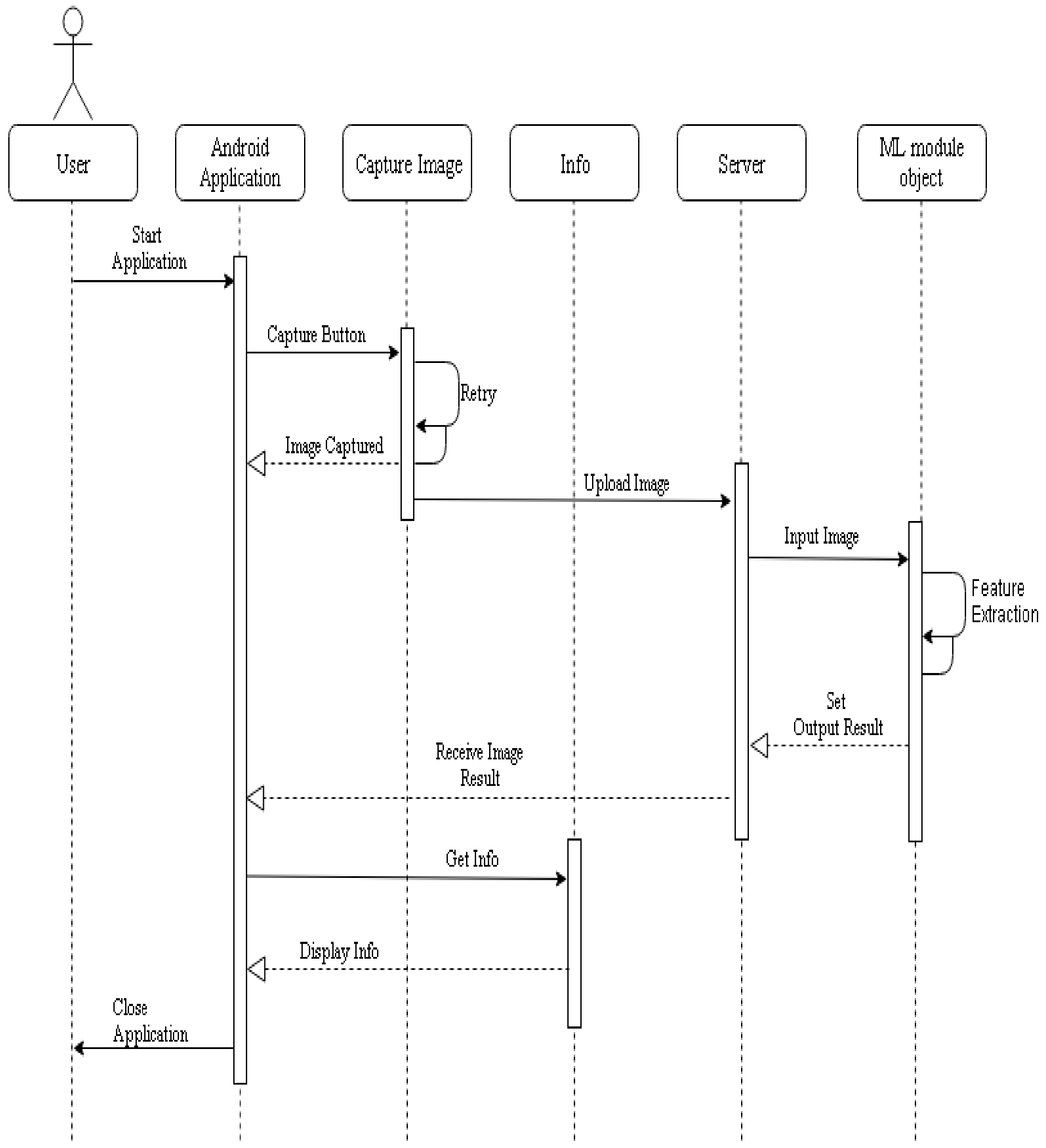


Figure 3.3: Sequence Diagram

3.4. Class Diagram

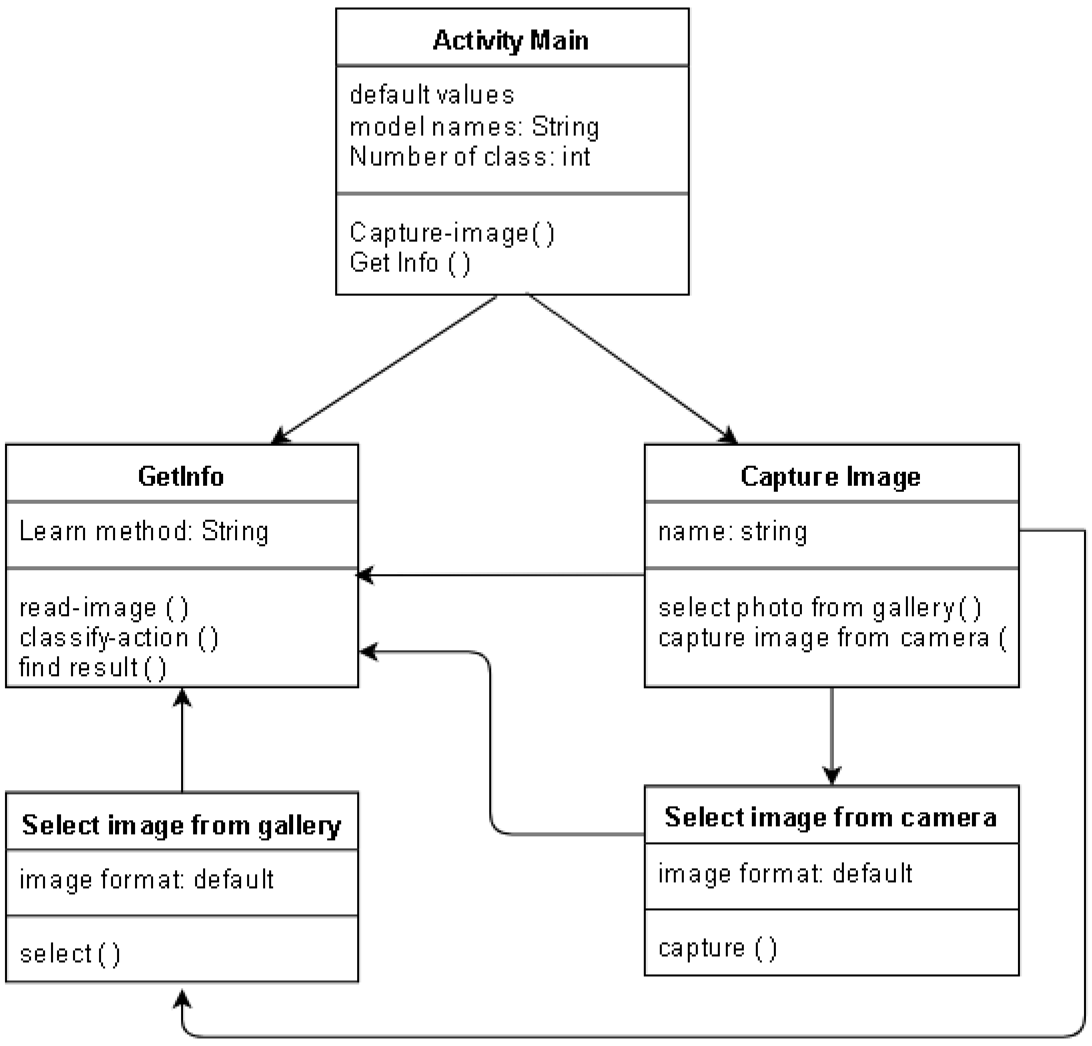


Figure 3.4: Class Diagram

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1. Implementation

Incremental Prototyping Model

Based on the Incremental Prototyping Model we first design a prototype of the script that will use the Kaggle datasets and Sklearn's CNN functions that will detect the disease on paddy leaves and then confirms whether the data is healthy or not, if not healthy then gives the unhealthy percentage.

At First, we capture the image, which is then send to the system and after the analysis is done; the result of image is thus given back to the user. Finally, again the CNN algorithm will be implemented in the script with our datasets collected from the system server. Further components. Further components (detection of diseases occurred on stem, root) can be added in the future as per the user needs.

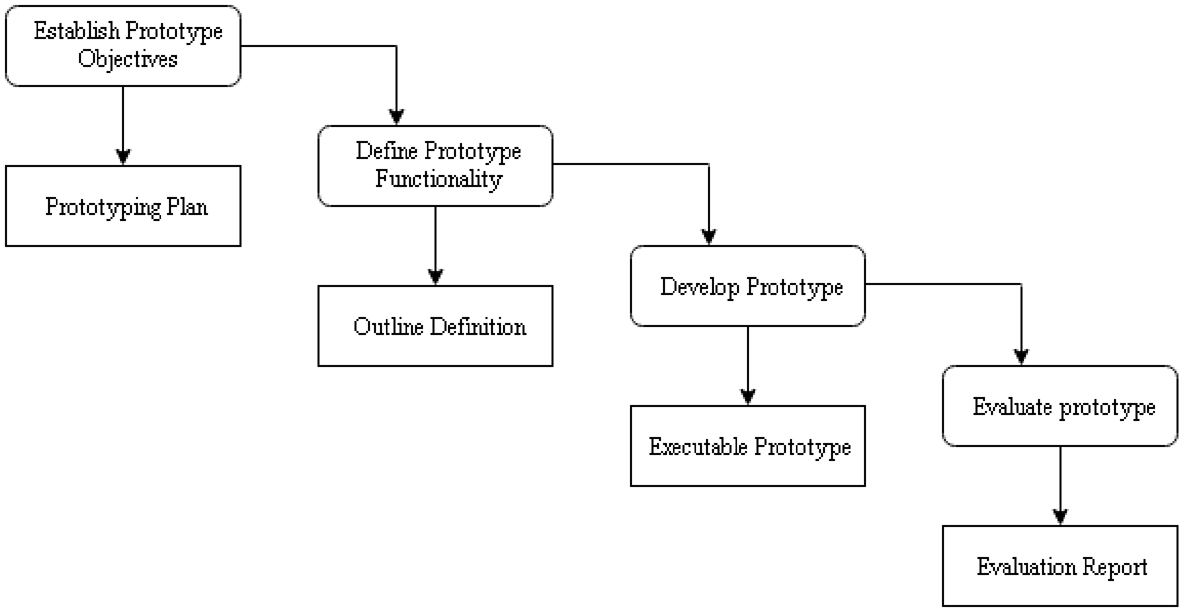


Figure 4.1: Incremental Prototyping Model

4.1.1. Analysis and Design Tools

There is various presence of designing tools to create figures and diagrams like use case diagram, sequence diagram and another desired diagram. In this project, Draw.io (i.e. flow chart maker and Online Diagram Software) was used for diagrammatic design of the proposed system.

Client Side

- Front end

We have used the following tools for our Front-End design:

XML- Android provides a straightforward XML vocabulary that corresponds to the view classes and sub-class such as for widgets and layouts. We used XML for android app design layout.

- Back end

Java- A general-purpose computer programming language that is object-oriented. We have implemented to design the required dependencies in application.

  - **Android Software Development Kit (SDK)**

    The Android SDK includes a comprehensive set of development tools. This includes a debugger, libraries, sample code, etc.

  - **Android Implementation**

    The API level we used is API 22. We used real device as our android virtual device to launch our application. Our application will run on Minimum API level API 15 Android 4.0.3 (Ice Cream Sandwich). App will run on approximately 100 % of device. Android studio version we used is 3.4.1 for windows.

Server Side

We have used the following tools for our Server-side design:

- **Python:** interpreted high-level programming language. Both train and testing data is done.
- **NumPy:** a library for programming language, which support language, multi-dimensional arrays and matrices.
- **SK Learn:** software Machine Learning library for the Python programming language. This helps to designed python Numerical and scientific libraries NumPy and SciPy.
- **Matplotlib:** a plotting library for Python programming language and for numerical mathematics extension.

Flask API

Flask API provides an implementation of browsable APIs(Application Platform Interface).API is a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service. Flask API gives proper content negotiated-responses and smart request parsing. It is designed to start quickly and easily, with the ability to scale up to complex application.

In our case, we use simple API for uploading ML module to the local server.To make local server, flask API is accessed through a URL. We import the different utils such as redirect, url etc which is used to connect with android application.

ngrok

ngrok is simply secure tunnels to localhost. Tons of programmers and designers rely on ngrok for quick testing of websites and apps on their mobile devices. When ngrok is running, it listens on the same port that our local web server is running on and proxies external requests to our local machine.
This means, connect the local server with android application with the URL, which returns and load it on our phone for immediate access to app.

In our case, it is a step to get it to listen to our local web server i.e generated using Flask. We ran the local server on port 5000. In terminal, we type ngrok http 5000. This starts ngrok listening on port 5000 and creates the secure tunnel.

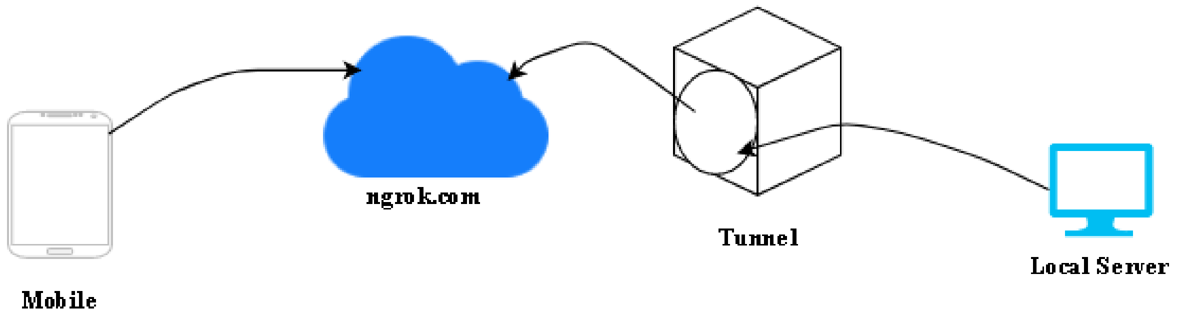


Figure 4.2: ngrok Architecture

It also creates two sets of publicly available URLs (http://8020a793.ngrok.io & https://8020a793.ngrok.io) that map to our local server. These unique URLs created each time we restart ngrok making it easy to share these out for one-time testing sessions across a team. The benefit to this, of course, is that we are now able to use RemoteIE with these public URLs to load and test your local site since to the service, your local site now looks like a staged or production system. Jinja is full feature template engine for Python.

Flask became one of the most popular Python web application frameworks began as a simple wrapper around Werkzeug and Jinja.

Werkzeug is comprehensive WSGI web application library. It is ~~the~~a collection of various utilities for WSGI application.

4.1.2. Others Tools and Platforms

- GitHub

  GitHub is a platform for code sharing and version controlling. In this project team members were from different residential location so, there was necessity of distributing working environment and for that GitHub was used for sharing the code. Each team member would pull the project, add some functionality and push those changes to the central repository to the GitHub.

- Visual Studio Code

  Visual Studio Code is a versatile and fun text editor developed by Microsoft repetitive tasks so you can focus the important stuff. It works on OS X, Windows and Linux. It is highly customizable, allowing the user to change the theme, keywords shortcut, preferences, and install extensions that add additional functionality.
- Android Studio

  Android Studio is an official integrated development environment (IDE)for Google's Android operating system. It is built on JetBrains' IntelliJ IDEA software and designed especially for Android development. It is available for windows, macOS and Linux based operating system.
- Word Processer

  Word Processor is used for documentation.

4.2. Methodology

4.2.1. Data Collection and Algorithms

The data used in this project are the images of the Paddy leaf among which of them were fed into the model for the training model and some were used for testing purpose and we downloaded it by creating the account from Kaggle.

- Kaggle

  Kaggle is an online platform that allows user to find and publish datasets. An open dataset forum help the user and the researcher to build models and solve the data science problems [5].

The detailing of data collection and modification is described below:

1. Downloaded dataset from Kaggle. We downloaded dataset labelled into four categories:
   i) BrownSpot
   ii) Healthy
   iii) Hispa
   iv) LeafBlast

Each label consists of multiple images of different size.

Attribute Information:

The format of all images is .jpg format. The images were reduced to the desired resolution for processing.

Table 4.1: Dataset Information

Datatype	Attribute Characteristics	Number of Instances				Number of attributes
Image	Categorical	BrownSpot	Hispa	Healthy	LeafBlast	Around 5
		523	565	1488	779	

The above table contains the dataset attributes that describes the shape, size, color and datatype of the dataset.

Image Processing Techniques

For the software development methodology for our project, we have chosen digital image processing. The phase in Image Processing are organized in such a way that current step provides the deliverables for the next phase. The figure below shows the following steps: -

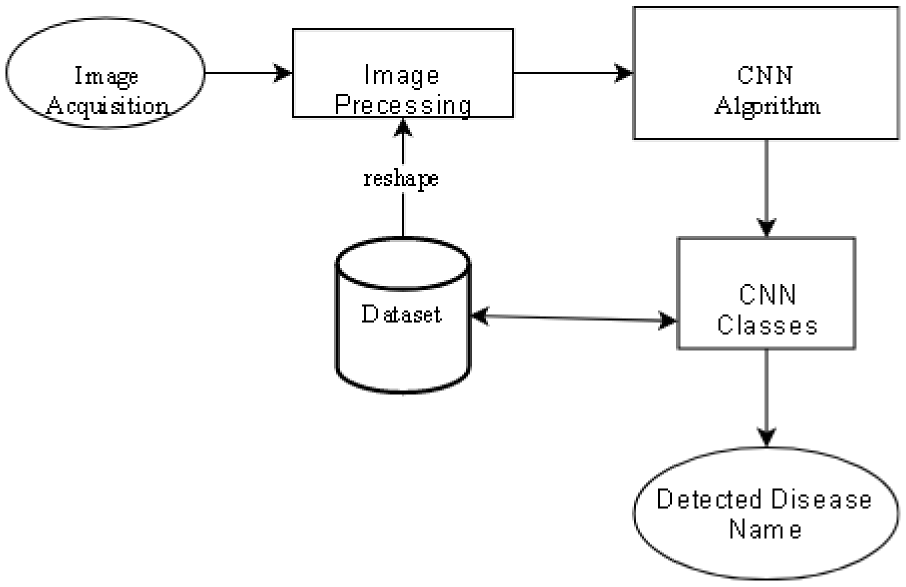


Figure 4.3: Image Processing

Initially, input image is given from the android application. The input image is then preprocessed by using library called OpenCv. The color image format is read by using Cv2.IMREAD_COLOR method and the image is resized by using the Cv2.resize() function.

Classification

We have used Convolutional Neural Network Algorithm as a image classifier. A CNN consist of number of convolutional and sampling layers optionally followed by fully connected layers. The network will consist of several convolutional neural networks mixed with non-linear and pooling layers.

4.3. Algorithm implemented

4.3.1. Convolutional Neural Network

The system use CNN to extract features and classify the image by the user in the user interface (android). Convolutional Neural Networks are very similar to ordinary Neural Networks. They are made up of neurons that have learnable weights and biases.

A convolutional Neural Network is comprised of convolutional layer and then followed by fully connected layer as in standard multilayer neural network.

The architecture of Convolutional Neural Network is:

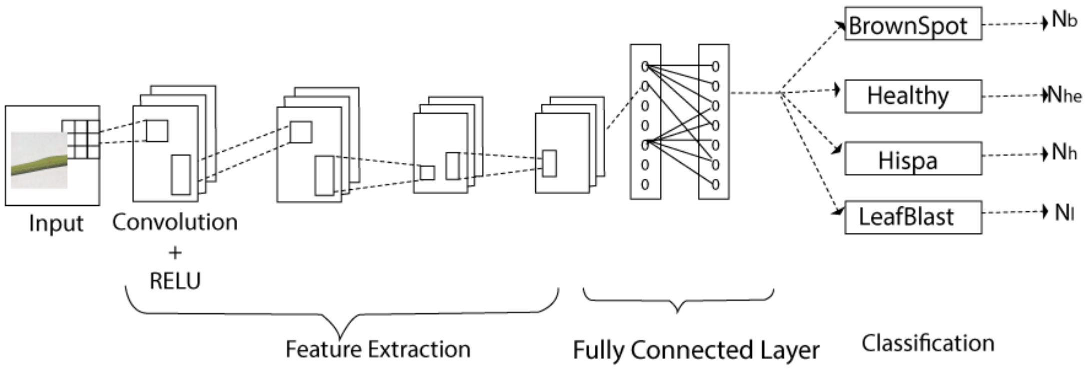


Figure 4.4: Convolutional Neural Network Architecture

Various Layers in CNN

Convolution Layer

The main building block of CNN is the convolutional layer. Convolution is a mathematical operation to merge two sets of information. The primary purpose of Convolution in case of ConvNet is to extract features from the input image Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data. A convolutional Neural Network is comprised of convolutional layer which is applied on the input data using a convolution filter to produce a feature map.

In our case first input image are stored in convent with image size as height, width and number of channel. Next we have 5 layers of convolution and pooling, we added a convolution layer by using the"Conv2D" function. The Conv2D function is taking 4 arguments, the first is the convent of image input size of previous, second is the dimension, third one filter, the fourth argument is the activation function i.e reLU activation to make all negative value to zero here 'relu' stands for a rectifier function.

$$W= \left(\frac{w-F+2P}{s}\right) +1$$

H = $(\frac{H-F+2P}{s})$ +1

Where,

W = width of input

H = height of input

F = filter size

P = padding

S = Stride

D = Depth or channel

Pooling Layer

A pooling layer perform a down sampling operation along the spatial dimensions (Width, Height), resulting dimensionality reduction. The primary aim of pooling operation is to reduce the size of the images as much as possible. This scan across the image using a window and compresses the image extracting features.

Max pooling and average pooling are the most common method used in pooling layers. Max pooling takes the largest value from the window of the image currently covered but the kernel, while average pooling takes the average of all values in the window.

For a given input the size of (W1xH1xD1), the output from the max pooling layer can be calculated using the formula:

.

Requiring two hyper parameters:

- Filter size f
- Strides S

· Produces a volume of size (W2xH2xD2) where,

- W2 = (w1-F)/s+1
- H2 = (H1-F)/s+1

- D2=D1

Fully Connected

After the convolution + (plus) pooling layers we add a couple of fully connected layers to wrap up the CNN architecture. It connects neurons in one layer to neurons in another layer. It is used to classify images between different categories by training. This layer takes an input volume (whatever the output is of the conv or relu of pool layer) and output an N dimensional vector where N is number of classes that the program has to choose from.

Basically, a FC layer looks at what high level features most strongly correlate to a particular weight so that when you compute the products between the weights and the previous layer, you get the correct probabilities for the different classes.

Dropout Layer

Dropout is used to prevent overfitting and the idea is very simple. During training time, at each iteration, a neuron is temporarily “dropped” or disabled with probability *p*. This means all the inputs and outputs to this neuron will be disable at the current iteration. The dropped-out neurons are resampled with probability p at every training step, so a dropped-out neuron at one-step can be active at the next one. The hyperparameter *p* is called the dropout-rate and it is typically a number around 0.8, corresponding to 80% of the neurons being dropped out.

Activation Function

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

There are various types of activation function available such as sigmoid, Tanh, ReLU, Softmax, etc. ReLU activation is used in hidden layers and Softmax at last layer.

ReLU:

Stands for Rectified linear unit. It is the most widely used activation function. Chiefly implemented in hidden layers of neural networks.

- **Equation: -** ***A(x) = max (0, x)***. It gives an output x if x is positive and 0 otherwise.
- **Value Range: -** [0, inf)

· **Nature: -** non-linear, which means we can easily back propagate the errors and have multiple layers of neurons being activated by the ReLU function.

· **Uses: -** ReLU is less computationally expensive than Tanh and sigmoid because it involves simpler mathematical operations. At a time only, a few neurons are activated making the network sparse making it efficient and easy for computation.

In simple words, RELU learns much faster than sigmoid and Tanh function.

$$F(z) = \begin{cases} z: z \geq 0 \\ 0: z < 0 \end{cases}$$

Softmax

The softmax function is also a type of sigmoid function but is handy when we are trying to handle classification problems.

· **Nature: -** non-linear

· **Uses: -** Usually used when trying to handle multiple classes. The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.

Output: - The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

$$S(y) = \frac{e^{y_i}}{\sum_e y_i}$$

Stride

Stride is the number of pixels by which sliding is done in the filter matrix over the input matrix When stride is 1 then filters are moved one pixel at a time. Having a large stride will produce smaller feature maps.

Training

When the CNN starts, the weights or filter values are randomly initialized. The filter doesn’t know to look for edge and curves. The idea of being given image and labels is training process that CNN go through.

The way the computer is able to adjust its filter value(or weights) is through a training process called back propagation can be separated into 4 distinct sections. The forward pass, the loss function. The backward pass and weight update.

Epochs, Batch Size, Iteration

One epoch is when an entire dataset is passed forward and backward through the NN only once. Since one epoch is too big to feed to the computer at once, it is divided in several smaller batches.

Batch Size is the total number of training examples present in single batch.

Iteration is the number of batches needed to complete one epoch.

Learning Rate is a hyper parameter that controls how much adjusting of weights is needed with respect the loss gradient. It takes a long time to coverage.

Wi +1 = Wi – n*gradient

If the learning rate is set too low, training will progress very slow as there are tiny updates to the weight in the network. However, if there is the learning rate is set too high it can cause undesirable divergent behavior in the loss function.

Optimizers

Optimizer helps to minimize (or maximize) an error function E(x) which is simply a mathematical function dependent on the model’s internal learnable parameters which are used in computing the target values (Y) from the set of predictors (X) used in the model.

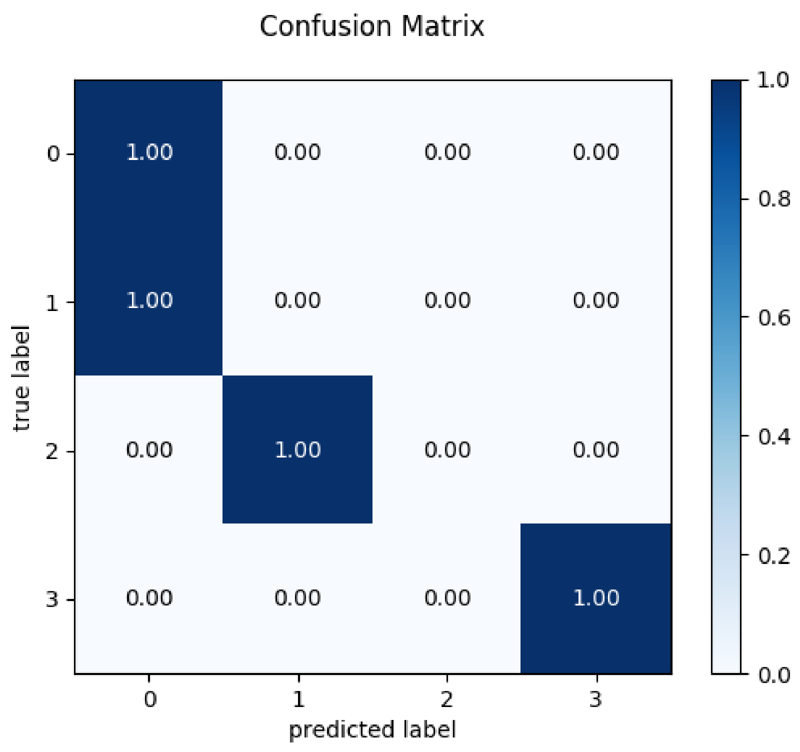
Various optimizer are Adam, AdaDelta, Gradient Descent, SGD, Momentum, Mini Batch Gradient Descent, etc.

Adam

Adam stands for Adoptive Moment Estimation, Adam is a method that computes adaptive learning rates for each parameter. In addition storing an exponentially decaying average of past squared gradients. Adam keeps an exponentially decaying average of past gradients M (t)

Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on the set of test data for which the true values are known.



0	Healthy
1	BrownSpot
2	Hispa
3	LeafBlast

Fig 4.5: Confusion Matrix

4.4. Testing

During our development process we have carried out a series of testing to check the differences between the given input and the expected output of our system. Finally, to see whether or not the CNN works, different set of images and labels (other than the training image) were passed through the CNN and the output results are seen and compared to calculate accuracy and loss.

4.4.1. Unit Testing

After the work was divided and coding completed the modules were parallel tested and after getting bug it was made bug free.

Table 4.2: Unit Testing of Android Application

Test Id	Test Unit	Pre-Condition/Test Data	Expected Outcome	Actual Result	Status
1	Test if the system captures the image	Android camera permission must be turned on	Show the capture image to user	Captured Image	Pass
2	Test if the system captures through gallery	Android gallery permission must be turned on	View all images of gallery to the users	Select single image	Fail
3	Test if the system captures through gallery	Android gallery permission must be turned on	View all images of gallery to the users	Select single image	Pass
4	Test if get info button gives the information or not	Image should be select either from gallery or capture image from camera	Viewed the information images to the user	Nothing displays	Fail

Table 4.3: Unit Testing of Script

Test Id	Test Unit	Pre-condition/ Test Data	Expected Outcome	Actual Result	Status
1	Run the script for training data	Load the images compiled model and start training network.	Script should able to give more accuracy	Script gives the accuracy	Fail
2	Run the script for training data	Load the images compiled model and start training network.	Script should able to give more accuracy	Script gives the accuracy with 99%	Pass
3	Run the script for training data and gives images from test set	Load the images compiled model and start training network.	Script should able to give more accuracy	Fits an image to the custom model with the given dataset and gives result	Pass
4	Run the script to classify given images	Load the images compiled model and start training network.	Script should able to give more accuracy	Problem in fitting the custom model with given dataset	Fail
5	Run the script to classify given images	Load the images compiled model and start training network.	Script should able to give more accuracy	Fits an image to the custom model with given dataset by adding probability value and gives result	Pass
6	Run the script of flask APIs	Proper POST or GET method should specify in script.	Script should able to fit and provides the URL with port number	Script is able to give result as a URL.	Pass

Table 4.4: Integration Testing

Test Id	Test Unit	Pre-Condition / Test Data	Expected Outcome	Actual Result	Status
1	Run the script by integrating capture image through camera and the gallery	Live captured to the respective image where permission must be turned on.	Capture real time through camera and viewed an image of gallery to a user.	Image captured and single image is selected from gallery	Pass
2	Run the script by integrating training and classify file	Live captured dataset.	Proper result by predicting the input image in console.	Predicted result appears	Pass
3	Run the system script by integrating the flask APIs and classify file	Proper GET/POST method with value should be specify.	Gives an URL in terminal through which performance is measured.	It gives URL through which it imports an image but failed to predict the information.	Fail
4	Run the system script by integrating the flask APIs and classify file	Proper GET/POST method with value should be specify.	Gives an URL in terminal through which performance is measured.	It gives URL through which it imports an image and predict the information.	Pass
5	Run the script by integrating URL generated by task api with secure tunnel i.e ngrok.	Using port generated by flask will accept by ngrok.	Session online with proper connection status	Connection status successful message.	Pass

Table 4.5: System Testing

Test ID	Test Unit	Pre-condition/ Text Data	Expected Outcome	Actual Results	Status
1	Open the application and choose camera option to capture image.	Live captured to the real image where permission must turn on.	Show the captured image to the user and provide proper information with get info button.	Show proper information according to input image captured.	Fail
2	Open the application and choose camera option to capture image.	Live captured to the real image where permission must turn on.	Show the captured image to the user and provide proper information with get info button.	Show proper information according to input image captured.	Pass
3	Open the application and choose to the gallery option to select an image.	Live select image where permission must be turned on.	Show the selected image to the user and provide proper information with get info button.	Show proper information according to input image selected.	Pass

CHAPTER 5: CONCLUSION AND ENHANCEMENT

REFERENCES

[1] S Bankar, A Dube, P Kadam, S Deokule, ""Plant disease detection techniques using canny edge detection & color histogram in image processing."," *International Journal of Computer Science and Information Technologies* , pp. 1165-1168, 2014.

[2] S Raut, K Ingole, ""Review on leaf disease detection using image processing techniques."," *International Research Journal of Engineering and Technology (IRJET) 4,,* pp. 2044-2047, 2017.

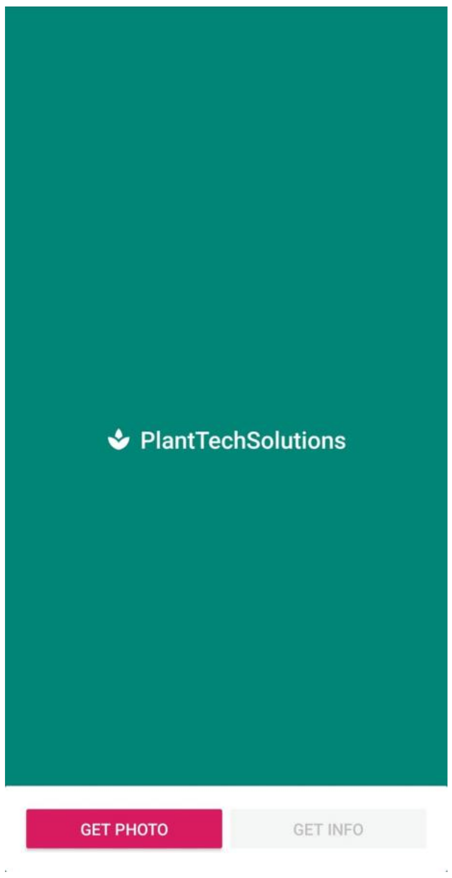
[3] S Aasha Nandhini, R Hemalatha, S Radha, and K Indumathi , "Web Enabled Plant Disease Detection System for Agricultural Applications Using WMSN," *Wireless Personal Communications,* vol. 102, no. 2, pp. 725-740, 2017.

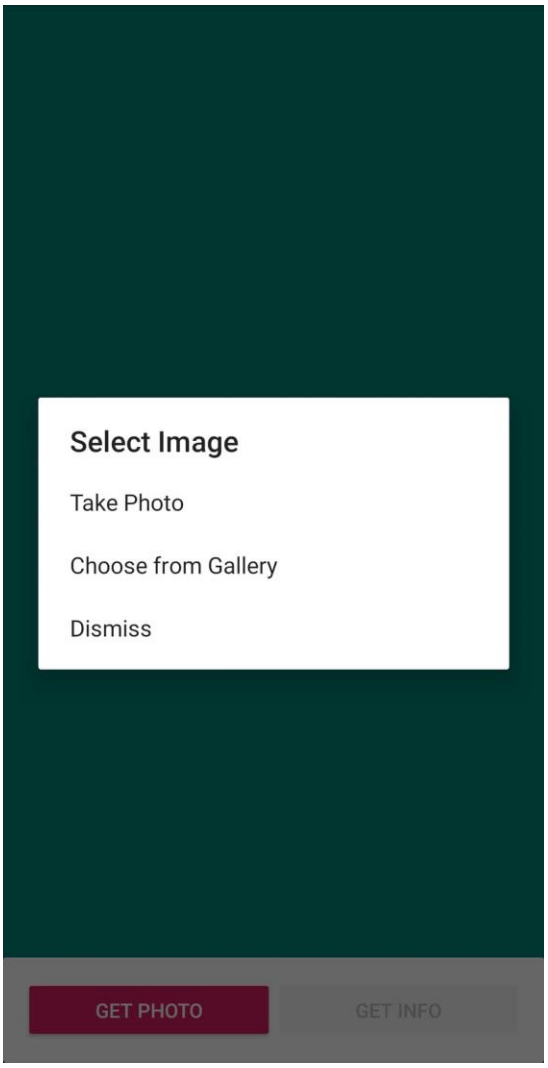
[4] R Swathi, T Mahalakshmi, A Srinivas, "Vision Based Plant Leaf Disease Detection on The Color Segmentation through Fire Bird V Robot".

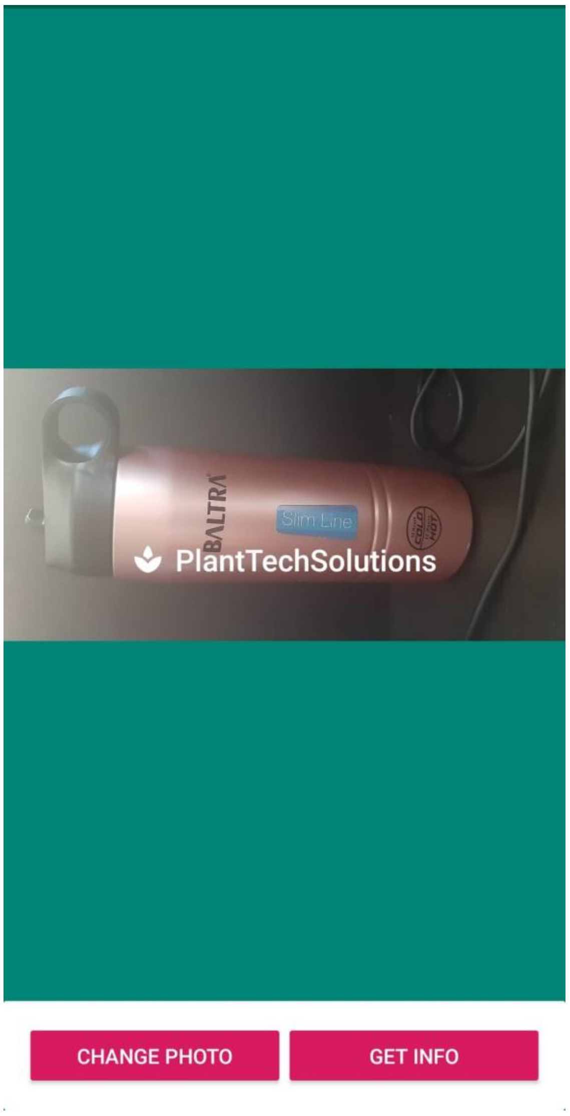
[5] https://www.kaggle.com/data/67121

APPENDICES

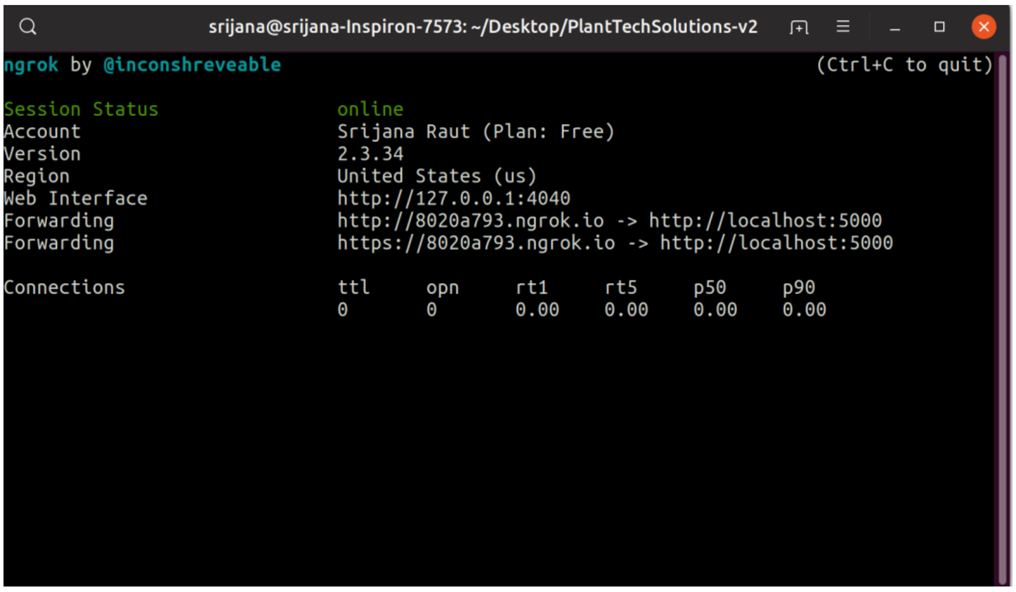
1. **Screenshot of UI**




Select Image
Take Photo
Choose from Gallery
Dismiss
GET PHOTO
GET INFO



2. Screenshot of ngrok


srijana@srijana-Inspiron-7573: ~/Desktop/PlantTechSolutions-v2

```
ngrok by @inconshreveable                                                        (Ctrl+C to quit)

Session Status                online
Account                       Srijana Raut (Plan: Free)
Version                       2.3.34
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://8020a793.ngrok.io -> http://localhost:5000
Forwarding                    https://8020a793.ngrok.io -> http://localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

3. Gantt Chart

Work Time	**March 15**	**March 30**	**April 1**	**April 20**	**April 21**	**April 25**	**May 1**	**May 31**	**June 1**	**June 15**
Planning	■	■								
Research & Analysis			■	■						
Design					■	■				
Coding						■	■	■	■	■
Implementation							■	■	■	■
Testing								■	■	■
Documentation	■	■	■	■	■	■	■	■	■	■