

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", MACHHE, BELGAUM-590014



PROJECT REPORT

ON

“IDENTIFYING THE AMOUNT OF FERTILIZATION REQUIRED FOR RICE CROPS USING IMAGE PROCESSING SYSTEM TO INCREASE THE YIELD”

Submitted in partial fulfillment of the requirements for the award of the degree

Bachelor of Engineering

In

Computer Science and Engineering

Of Visvesvaraya Technological University, Belgaum

Submitted by:

ROMANA TAZEEN	1CD12CS093
SHILPA H N	1CD12CS104
USHA P	1CD12CS115
YASHASWINI C	1CD12CS121

Under the Guidance of:

Prof Jayanthi M G, Dept. of CSE

Dr. Shashi Kumar D R
Prof & HOD, Dept. of CSE



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CAMBRIDGE INSTITUTE OF TECHNOLOGY

BANGALORE – 560036

2015 – 2016

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible because *“Success is the abstract of hard work and perseverance, but steadfast of all is encouraging guidance.”* So we acknowledge all those whose guidance and encouragement served as a beacon light and crowned our efforts with success.

We would like to profoundly thank our Chairman **Shri. D K Mohan**, Cambridge Group Of Institutions for providing such a healthy environment for the successful completion of Project Work.

We would like to express our thanks to the Principal **Dr. Suresh L**, for his encouragement that motivated us for the successful completion of Project Work.

We wish to express our gratitude to our guide **Dr. ShashiKumar D R**, Professor and Head of Department of Computer Science and Engineering, for his help, motivation, sharing his technical expertise and providing inspiration required for taking the project to its completion.

Also we would like to thank our Project coordinator **Mr. Raghavendra T S**, Associate Professor, Department of CSE, for his constant support and guidance throughout the Project Work.

We would like to thank our college administration for providing a conducive environment and also suitable facilities for this project. We thank all the teaching and non-teaching staff of the Department of Computer Science and Engineering for providing resources for the completion of the project.

Finally we would hereby acknowledge and thank all our parents and friends who have been a source of inspiration and also instrumental in the successful completion of this Project.

Romana Tazeen (1CD12CS093)

Shilpa H N (1CD12CS104)

Usha P (1CD12CS115)

Yashaswini C (1CD12CS121)

DECLARATION

We, Romana Tazeen, Shilpa H N, Usha P and Yashaswini C students of eighth semester BE, Computer Science & Engineering, Cambridge Institute of Technology hereby declare that the project work entitled **“Image Processing For Fertilization Management Of Crops”** has been carried out by us at Cambridge Institute of Technology, Bangalore and submitted in partial fulfillment of the course requirements for the award of the degree of **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum**, during the academic year 2015-2016.

We also declare that, to the best of our knowledge and belief, the work reported here does not form part of any other dissertation on the basis of which a degree or award was conferred on an earlier occasion on this by any other student.

Date:

Place: Bangalore

Romana Tazeen (1CD12CS093)

Shilpa H.N (1CD12CS104)

Usha P (1CD12CS115)

Yashaswini C (1CD12CS121)

ABSTRACT

The project focuses on providing the information regarding the pesticide/insecticide and the amount of pesticide/insecticide to be used for an unhealthy crop. The user, who is the farmer clicks a picture of the crop and uploads it to the server via the android application. After uploading the image the farmer gets an unique ID displayed on his application screen. The farmer has to make note of that ID since that ID has to be used by the farmer later to retrieve the message after a while. The uploaded image is then processed and accordingly the features of that image are extracted. Based on those features the clustering of image is done and the best cluster giving the maximum information regarding the affected part is selected. Then the result consisting of the disease name and the affected area is retrieved. This result is then uploaded into the message table in the server. Now the Farmer will be able to retrieve the complete information in a presentable format by entering the unique ID he had received in the Application.

LIST OF CONTENTS

CHAPTER NAME	PAGE NO
1 INTRODUCTION	01
1.1 Principle	01
1.2 Scope	02
2 LITERATURE SURVEY	03
3 SYSTEM ANALYSIS	05
3.1 Present Scenario	05
3.2 Proposed System	06
3.3 Requirement Specification	07
3.3.1 Functional Requirements	07
3.3.2 Non-Functional Requirements	08
4 SYSTEM DESIGN	09
4.1 Application Development	10
4.1.1 Creation of server	10
4.1.2 Uploading Image	10
4.1.3 View Message	12
4.2 Image Processing	13
4.2.1 Image preprocessing	13
4.2.2 Segmentation	14
4.2.3 Feature Extraction	16
4.2.4 Classification	16
4.3 UML Diagrams	20
4.3.1 Use Case Diagram	20
4.3.2 Sequence Diagram	21

5 IMPLEMENTATION	22
5.1 Android Code	22
5.1.1 Layout Code	22
5.1.2 Main Activity	24
5.1.3 View Message	26
5.2 PHP Code	28
5.2.1 Upload	28
5.2.2 Get Message	29
5.3 Matlab Code	30
 6 TESTING	 34
6.1 Aim Of Testing	35
6.2 Unit Testing	35
6.3 Integration Testing	36
6.4 Validation Testing	36
6.5 User Acceptance Testing	37
6.5.1 White Box Testing	37
6.5.2 Black Box Testing	38
 7 SNAPSHOTS	 39
 CONCLUSION	
 REFERENCES	

LIST OF FIGURES

FIGURE	NAME	PAGE NO
Fig 3.1	Block Diagram for Farmer's portal	05
Fig 3.2	Fertilization Management for crops	07
Fig 4.1	Application Layout	09
Fig 4.2	Image table in server where the images are uploaded	11
Fig 4.3	Message table in the server where the message contents are uploaded	12
Fig 4.4	Image Processing Architecture	14
Fig 4.5	SVM Classification	17
Fig 4.6	SVM Dataset	18
Fig 4.7	Use Case Diagram	20
Fig 4.8	Sequence Diagram	21
Fig 7.1	Application Layout	39
Fig 7.2	Gallery Images	39
Fig 7.3	Image Selected and Displayed	40
Fig 7.4	Image uploaded and ID displayed	40
Fig 7.5	Image table in server where the images are uploaded	41
Fig 7.6	Mat lab User Interface where the selected image is displayed	42
Fig 7.7	Clustering of the input Image	42
Fig 7.8	Help Dialog displaying the Crop Disease Name	43
Fig 7.9	Mat lab Command prompt where the affected area is displayed	43
Fig 7.10	Message table in the server where the message contents are uploaded	44
Fig 7.11	Application Displaying the Message	45

LIST OF TABLES

TABLE	NAME	PAGE NO
Table 6.1	Unit Testing Table	36
Table 6.2	Validation Testing Table	37

CHAPTER 1

INTRODUCTION

An Image is a two dimensional signal. Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. Advent of new technologies such as Digital image processing and Image analysis technology has many applications in the biological field. About 78% of the farmers are small and marginal in the country and they are poor in resources. Therefore, they are not in a position to use optimum quantity of inputs in their crops which are essential for increasing the productivity. Most of farmers may not know the amount of fertilizer required for crops and thus it may lead to unbalanced use of fertilizer and they may also not know which pesticide/insecticide to be used for the diseased crop. Hence the yield gets affected.

A user friendly application installed on Android phone may to some extent help farmer solve the problem of a farmer to detect a disease. The farmer clicks the image of the crop and sends it. The image is processed using the Image Processing techniques and the disease is detected. The details of the disease and the area affected along with the amount of pesticide/insecticide are sent to the farmer and the farmer can see the details in his application. This may prove benefits in monitoring large fields of crops, and thus automatically detect the symptoms of diseases as soon as they appear on plant leaves.

1.1 PRINCIPLE

The purpose of this project is to obtain an image from the farmer of the diseased crop preferably the stem or the leaves through the Android Application installed on farmer's phone. The image is then processed using image-processing technique and the disease type is detected. The diseases affected to the crop and the amount of fertilizer or the pesticide/insecticide to be used is updated to the Android Application that was previously used by the farmer to upload image.

1.2 SCOPE

Agriculture is the backbone of the Indian Economy''- said Mahatma Gandhi six decades ago. Even today, the situation is still the same, with almost the entire economy being sustained by agriculture. It contributes 16% of the overall GDP and accounts for employment of approximately 52% of the Indian population. Rapid growth in agriculture is essential not only for self-reliance but also to earn valuable foreign exchange. It is felt that with provision of timely and adequate inputs such as fertilizers, seeds, pesticides and by making available affordable agricultural credit /crop insurance, Indian farmers are going to ensure food and nutritional security to the Nation. Hence the detection of plant diseases is an important aspect in increasing the yield of a crop. By detecting the plant disease and identifying which pesticide/insecticide and the amount of pesticide/insecticide to be used increases the crop yield which leads in growth of country's economy.

CHAPTER 2

LITERATURE SURVEY

Plant diseases have turned into a dilemma as it can cause significant reduction in both quality and quantity of agricultural products. Plant pests and diseases affect food crops, causing significant losses to farmers and threatening food security. The spread of transboundary plant pests and diseases has increased dramatically in recent years. Globalization, trade and climate change, as well as reduced resilience in production systems due to decades of agricultural intensification, have all played a part. Outbreaks and upsurges can cause huge losses to crops and pastures, threatening the livelihoods of vulnerable farmers and the food and nutrition security of millions at a time.

Since the beginning of agriculture, generations of farmers have been evolving practices for combating the various plagues suffered by our crops. Following the discovery of the causes of plant diseases in the early nineteenth century, growing understanding of the interactions of pathogen and host has enabled us to develop a wide array of measures for the control of specific plant diseases.

From this accumulated knowledge base, we can distil some general principles of plant disease control that can help us address the management of new problems on whatever crop in any environment. One such set of principles, first articulated by H. H. Whetzel in 1929 and modified somewhat by various authors over the years, has been widely adopted and taught to generations of plant pathology students around the world. These "traditional principles", as they have come to be known, were outlined by a committee of the US National Academy of Sciences, 1968.

Automatic detection of plant diseases provides benefits in monitoring large fields of crops, and thus automatically detects the diseases from the symptoms that appear on the plant leaves. This enables machine vision that is to provide image based automatic inspection.

From the advent Digital Image Processing many people have tried and classified diseases using many techniques.

Kim et al. (2009) have classified the grape fruit peel diseases using color texture features analysis. The texture features are calculated from the Spatial Gray-level Dependence Matrices (SGDM) and the classification is done using squared distance technique. Grape fruit peel might be infected by several diseases like canker, copper burn, greasy spot, melanose and wind scar (Kim et al., 2009). Helly et al. (2003) developed a new method in which Hue Saturation Intensity (HIS) - transformation is applied to the input image, then it is segmented using Fuzzy C-mean algorithm. Feature extraction stage deals with the color, size and shape of the spot and finally classification is done using neural networks (Helly et al., 2003). Real time specific weed discrimination technique using multilevel wavelet decomposition was proposed by Siddiqil et al. (2009).

In this histogram equalization is used for preprocessing. Features are extracted from wavelet decomposition and finally classified by Euclidean distance method (Siddiqil et.al, 2009) Al-Bashish et al. (2011) developed a fast and accurate method in which the leaf diseases are detected and classified using k-means based segmentation and neural networks based classification. Automatic classification of leaf diseases is done based on high resolution multispectral and stereo images (Bauer et al., 2011). Sugar beet leaves are used in this approach. Segmentation is the process that is carried out to extract the diseased region and the plant diseases are graded by calculating the quotient of disease spot and leaf areas. An optimal threshold value for segmentation can be obtained using weighted Parzen-window (Jun and Wang, 2008). This reduces the computational burden and storage requirements without degrading the final segmentation results.

CHAPTER 3

SYSTEM ANALYSIS

3.1 PRESENT SCENARIO

Plant diseases cause periodic outbreak of diseases which leads to large scale death and famine. It is estimated that the outbreak of helminthosporiose of rice in north eastern India in 1943 caused a heavy loss of food grains and death of a million people. Since the effects of plant diseases were devastating, some of the crop cultivation has been abandoned. It is estimated that 2007 plant disease losses in Georgia (USA) is approximately \$653.06 million (Jean, 2009). In India no estimation has been made but it is more than USA because the preventive steps taken to protect our crops are not even one-tenth of that in USA.

The naked eye observation of experts is the main approach adopted in practice for detection and identification of plant diseases. But, this requires continuous monitoring of experts which might be prohibitively expensive in large farms. Further, in some developing countries, farmers may have to go long distances to contact experts, this makes consulting experts too expensive and time consuming and moreover farmers are unaware of non-native diseases.



Fig 3.1: Block Diagram for Farmer's portal.

It is envisaged to make available relevant information and services to the farming community and private sector through the use of information and communication technologies, to supplement the existing delivery channels provided for by the department. Farmers' Portal is an endeavor in this direction to create one stop shop for meeting all information needs relating to agriculture of an Indian farmer. Once in the Farmers' Portal, a farmer will be able to get all relevant information on specific subjects around his village/block/district or state. Using the farmers' portal the farmer calls this service center and clears all his queries. The people present at the service search for relevant information or they are trained with all information to help farmers.

This also has a disadvantage as the people at the service center cannot see the exact problem the farmer is facing. They cannot imagine the severity of the disease completely and hence at times it may result in wrong disease detection. This may also lead in destruction of the crop.

3.2 PROPOSED SYSTEM

In the proposed system at first the images are acquired from the farmer. The images are received from the farmer via the Android Application developed exclusively for the service of the farmer. The images are uploaded by the farmer by choosing the appropriate image of the leaf or the stem preferably from the Choose File option. On uploading a image the farmer receives a ID which has to be used later by him to check the pesticides for the affected disease.

The image uploaded by the farmer is processed by the MATLAB. Then image-processing techniques are applied to the acquired images to extract useful features that are necessary for further analysis. After that, several analytical techniques are used to classify the images according to the specific problem at hand.

The disease type is detected and displayed by the MATLAB. The affected area is also displayed to identify the severity of the disease. The pesticides for the detected disease and the amount to be given to the plant are entered into the database. The farmer in order to see the details has to click another button in the app which is View Message. On entering the ID which was previously displayed to the farmer the farmer can view the details that were uploaded.

Automatic detection of plant diseases is an important research topic as it may prove benefits in monitoring large fields of crops, and thus automatically detect the diseases from the symptoms that appear on the plant leaves. This enables machine vision that is to provide image based automatic inspection, process control and robot guidance. Comparatively, visual identification is labor intensive, less accurate and can be done only in small areas.

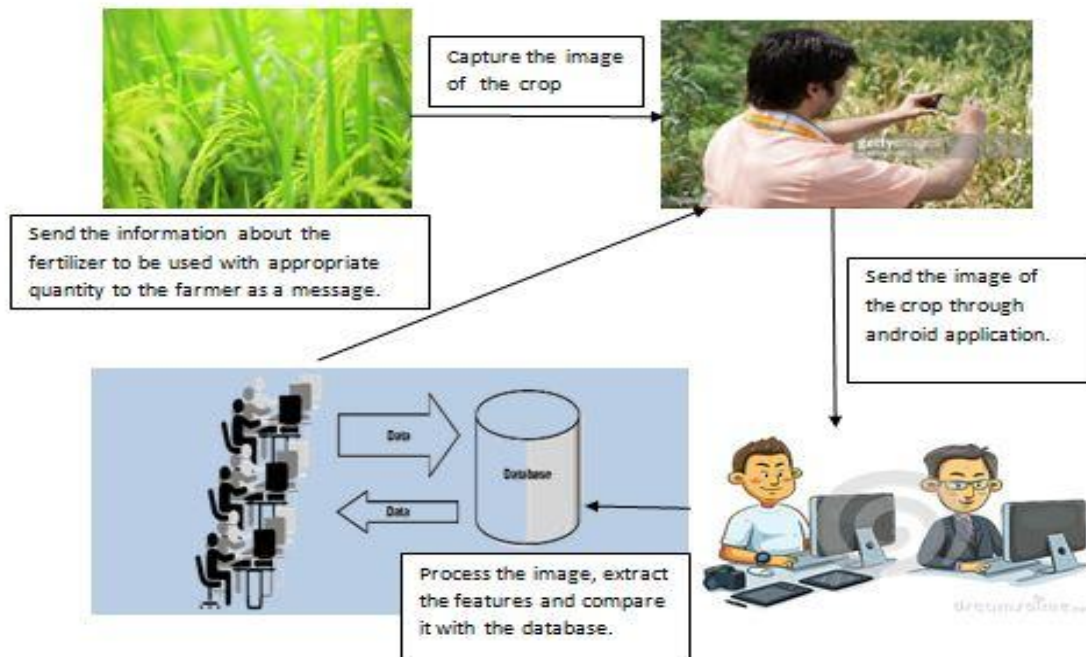


Fig 3.2: Fertilization Management for crops.

3.3 REQUIREMENT SPECIFICATION

3.3.1 FUNCTIONAL REQUIREMENTS

Functional requirements define the internal workings of the software: that is, the technical details, data manipulation and processing and other specific functionality that show how the use cases are to be satisfied. They are supported by non-functional requirements, which impose constraints on the design or implementation.

SOFTWARE REQUIREMENTS

Language : JAVA

OS : Windows 10 (64 bit)

IDE : Android Studio, MATLAB

HARDWARE REQUIREMENTS

Processor : Above 1.5GHZ

Hard Disk : 80GB

RAM : 2GB

3.3.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical non-functional requirements are reliability, scalability, and cost. Non-functional requirements are often called the ilities of a system. Other terms for non-functional requirements are "constraints", "quality attributes" and "quality of service requirements".

Reliability: If any exceptions occur during the execution of the software it should be caught and thereby prevent the system from crashing.

Scalability: The system should be developed in such a way that new modules and functionalities can be added, thereby facilitating system evolution.

Cost: The cost should be low because a free availability of software package.

CHAPTER 4

SYSTEM DESIGN

4.1 APPLICATION DEVELOPMENT

The android application for the farmer was developed using the android studio. **Android Studio** is the official Integrated Development Environment (IDE) for Android platform development. The development of the app had various stages such as creation of server, storing images in the database, accessing images from database, updating information into the database and extracting message from the database. The procedure first was started by creating a server by creating account in Hostinger and a database and consequently a table for storing images was created. Later a php script was written for connecting to the server and the database to the Android Application.

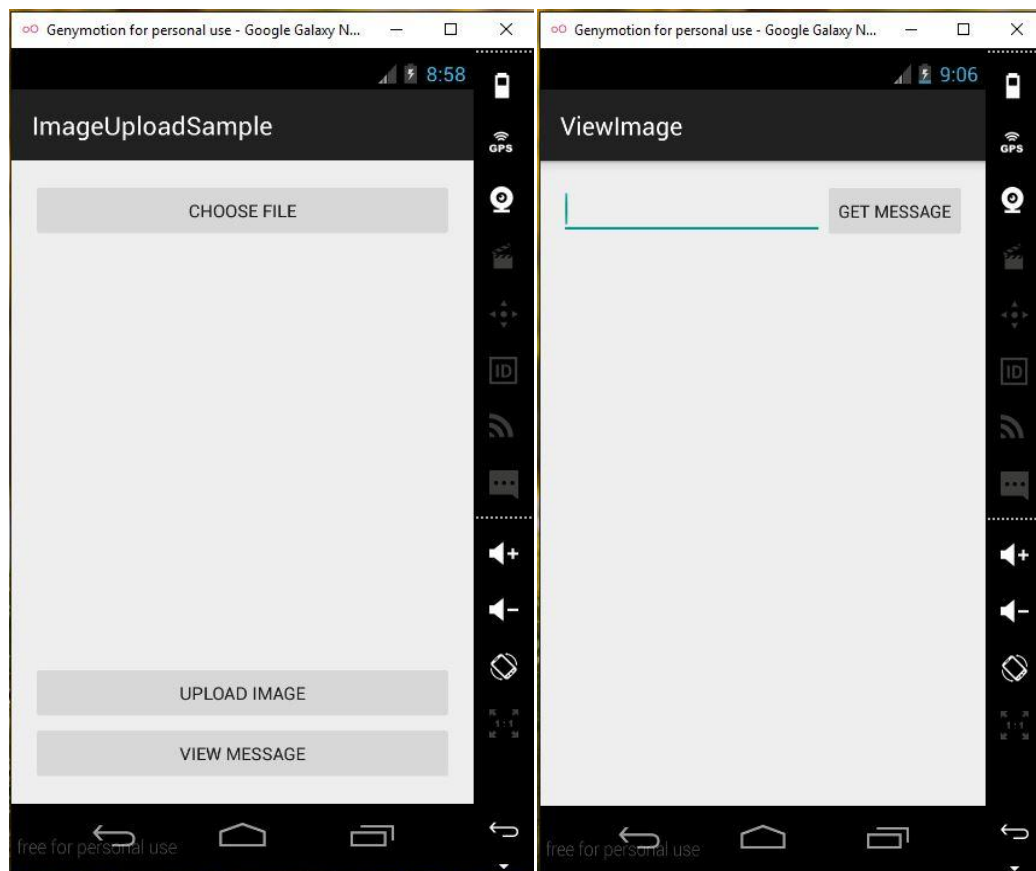


Fig 4.1: Application Layout

The layout of Android app contained of three buttons which are Choose File, Upload Image and View Message. The View Message in turn would open a layout which contained a text box and a button called Get Message and a text area to display the message as shown above.

4.1.1 CREATION OF SERVER

The first and foremost task is to create a server and this is done by creating a account. A relevant database is created in PHPMyadmin. The table images is subsequently created with two columns one being ID and the other being 'image' which is of type BLOB. When an image is uploaded the image is uploaded into this table. Another table called message table is also created so that after processing image we can store values into the table.

4.1.2 UPLOADING IMAGE

When the farmer selects the Choose File option gallery folder opens so that the farmer can select image from the folder. On selection of the picture the picture gets displayed. On selection of right image and the image being displayed in the app the farmer can click on Upload image option. The image gets uploaded to the server and gets stored in the database. An ID is returned to the farmer with a message that image was uploaded successfully.

For uploading the image a php script was uploaded into the server. The script was given a link in Main Activity. So that on clicking upload image button in turn the php script is called. The Upload Image inserts the image into the table by using INSERT query. It displays "Successfully Uploaded Image" and the ID s displayed. The ID is displayed using the attribute LAST_INSERT_ID. The ID is auto incremented value which is made so while creating the table.

localhost » u360185672_pdata » images

Browse Structure SQL Search Insert Export Import

✓ Showing rows 0 - 6 (7 total, Query took 0.0204 sec)

```
SELECT *
FROM `images`
LIMIT 0, 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

	ID	image
<input type="checkbox"/> Edit Copy Delete	1	[BLOB - 255.3 KiB]
<input type="checkbox"/> Edit Copy Delete	2	[BLOB - 400.2 KiB]
<input type="checkbox"/> Edit Copy Delete	3	[BLOB - 268.8 KiB]
<input type="checkbox"/> Edit Copy Delete	4	[BLOB - 57.5 KiB]
<input type="checkbox"/> Edit Copy Delete	5	[BLOB - 12.5 KiB]
<input type="checkbox"/> Edit Copy Delete	6	[BLOB - 124 KiB]
<input type="checkbox"/> Edit Copy Delete	7	[BLOB - 664.2 KiB]

Check All / Uncheck All With selected: Change Delete Export

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Fig 4.2: Image table in server where the images are uploaded

4.1.3 VIEW MESSAGE

The View Message is in turn another activity which gets opened. It has a text box where the ID that was generated for the farmer which is unique to every farmer has to be entered by him correctly.

A php script GETMESSAGE is been uploaded to the server. This script fetches the message from the database. In the database created earlier which had table 'images' where the uploaded images are stored, another table called 'message' was created. After analyzing the image in the MATLAB the details of the disease and the amount of fertilizer are updated into the table as per the ID of the image.

The message is fetched from the table and its displayed in the text area of the activity layout. Details such as ID, Crop Name, Disease name, Pesticide, Amount of pesticide, Affected Region are displayed.

Showing rows 0 - 7 (8 total, Query took 0.0002 sec)

```
SELECT *
FROM `message`
LIMIT 0, 30
```

Profiling [Inline] [Edit] [Exp]

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

ID	Crop Name	Affected Disease	Affected Area	Pesticide Name	Amount Of Pesticide
1	Rice ರೈಸ್ (ಭತ್ತ)	Leaf Blast ಲೀಫ್ ಬ್ಲಾಸ್ಟ್	15.2555%	Tricyclazole ತ್ರಿಸೈಕ್ಲಜೋಲೆ	75 WP @ 2 g/kg
2	Rice ರೈಸ್ (ಭತ್ತ)	Brown Spot ಬ್ರೌನ್ ಸ್ಪಾಟ್	15.0732%	Mancozeb ಮಂಕೊಜೆಬ್	75 WP @ 2.5 g/litre
3	Rice ರೈಸ್ (ಭತ್ತ)	Stem Rot ಸ್ಟೆಮ್ ರಾಟ್	17.3926%	Isoprothiolane ಈಸೋಪ್ರೋಥಿಯೋಲಿನ್	40 EC @ 1.5 ml/litre.
4	Rice ರೈಸ್ (ಭತ್ತ)	Sheath Blight ಶೀತ್ ಬ್ಲೈಟ್	16.1347%	Validamycin ವಾಲಿಡಾಮೈಸಿನ್	3 L @ 2.5 ml/litre

Fig 4.3: Message table in the server where the message contents are uploaded

4.2 IMAGE PROCESSING

Digital image processing is the use of computer algorithms to perform image processing on digital images. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be model in the form of [multidimensional systems].

The image is acquired from the server by using another php script GetImage. It has query to access the image. The image can be accessed by MATLAB using the URL of the php file in the server. The GUI of MATLAB has an interface which consists of a button called Image Select. On clicking Image Select the image directly gets downloaded from server. The acquired image is preprocessed so that it can be used for further use.

The following steps are followed for detecting disease in crop:

- Image Preprocessing
- Segmentation
- Feature Extraction
- Classification

4.2.1 IMAGE PREPROCESSING

The image acquired is preprocessed. The preprocessing starts by converting the RGB image to $L^*a^*b^*$ color space. The $L^*a^*b^*$ color space consists of Luminosity layer L^* , chromacity layer a^* and b^* . All of the color information is stored in the layers a^* and b^* . It requires to make color form so that the RGB colored image is converted to $L^*a^*b^*$ space. The function is `makecform()`, later the format is applied to the image that was acquired.

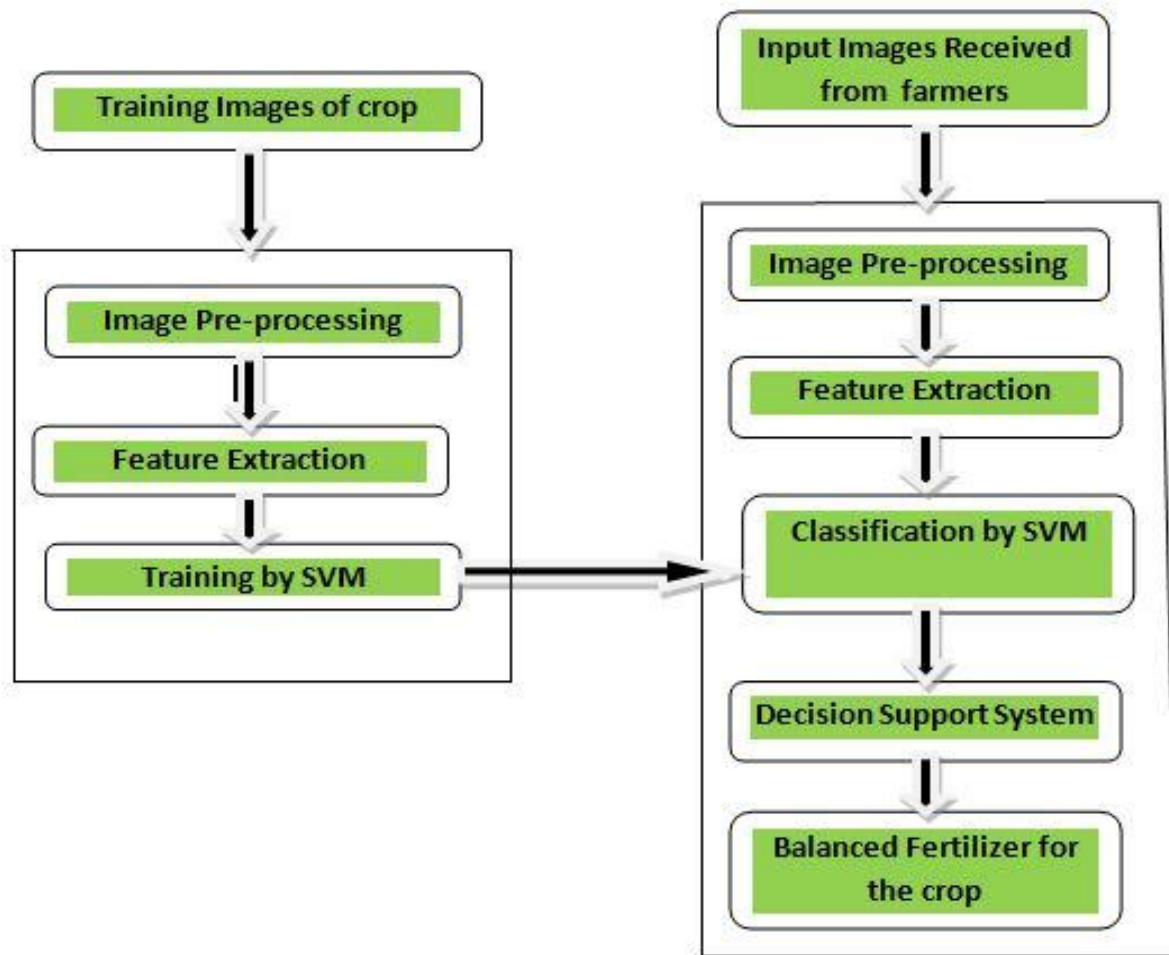


Fig 4.4: Image Processing Architecture

4.2.2 SEGMENTATION

There are several algorithms used for segmentation but one of the best methods used for detection of disease is k -means clustering. **k -means clustering** is a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k -means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k -means clustering tends to find clusters of comparable spatial extent, while the expectation-maximization mechanism allows clusters to have different shapes.

The algorithm has a loose relationship to the k -nearest neighbor classifier, a popular machine learning technique for classification that is often confused with k -means because of the k in the name. One can apply the 1-nearest neighbor classifier on the cluster centers obtained by k -means to classify new data into the existing clusters.

Classify the colors $a*b*$ color space using k -means clustering. Since the image has 3 colors we create three clusters. Measure the distance Euclidean Distance Metric. Label every pixel in that image using results from K means.

Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$, where each observation is a d -dimensional real vector, k -means clustering aims to partition the n observations into k ($\leq n$) sets $\mathbf{S} = \{S_1, S_2, \dots, S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (sum of distance functions of each point in the cluster to the K center). In other words, its objective is to find:

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \quad \dots\dots (4.1)$$

where $\boldsymbol{\mu}_i$ is the mean of points in S_i .

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters).

In cluster analysis, the k -means algorithm can be used to partition the input data set into k partitions.

Label every pixel in the image using results from K means. Then a blank cell array is created to store the results of clustering. Followed by create RGB label using pixel_labels. Selection of appropriate cluster is another important aspect. The cluster which displays the maximum disease affected part is to be selected. In the next step of feature extraction, the features of the selected cluster are extracted.

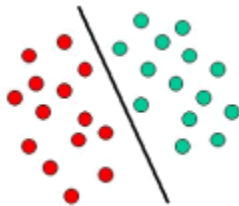
4.2.3 FEATURE EXTRACTION

The features of the selected cluster are extracted. The selected image is converted to grayscale since the image is in RGB format. At the next step the Gray Level Cooccurrence Matrices (GLCM). The required statistics are derived from Gray level cooccurrence Matrices (GLCM). The following 13 features that is extracted and evaluated:

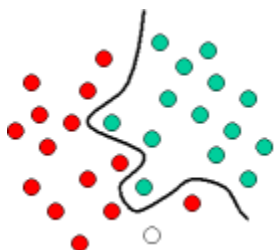
Contrast, Corelation, Energy, Homogenity, Mean, Standard Deviation, Entopy, RMS. Variance, Smoothness, Kurtosis, Skewness. The thirteen features are stored in an array.

4.2.4 CLASSIFICATION

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships.



The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the illustration below. Compared to the previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.



You can use a support vector machine (SVM) when your data has exactly two classes. An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. The *best* hyperplane for an SVM means the one with the largest *margin* between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points.

The *support vectors* are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab.

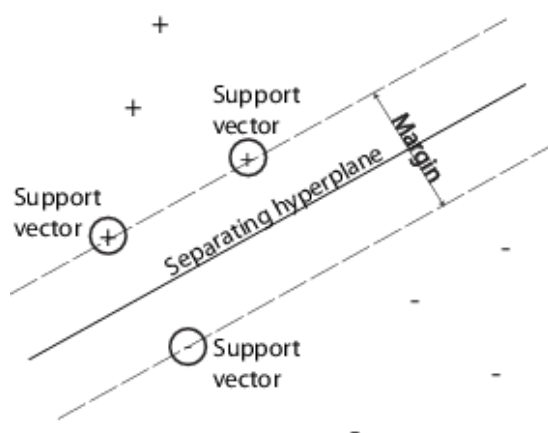


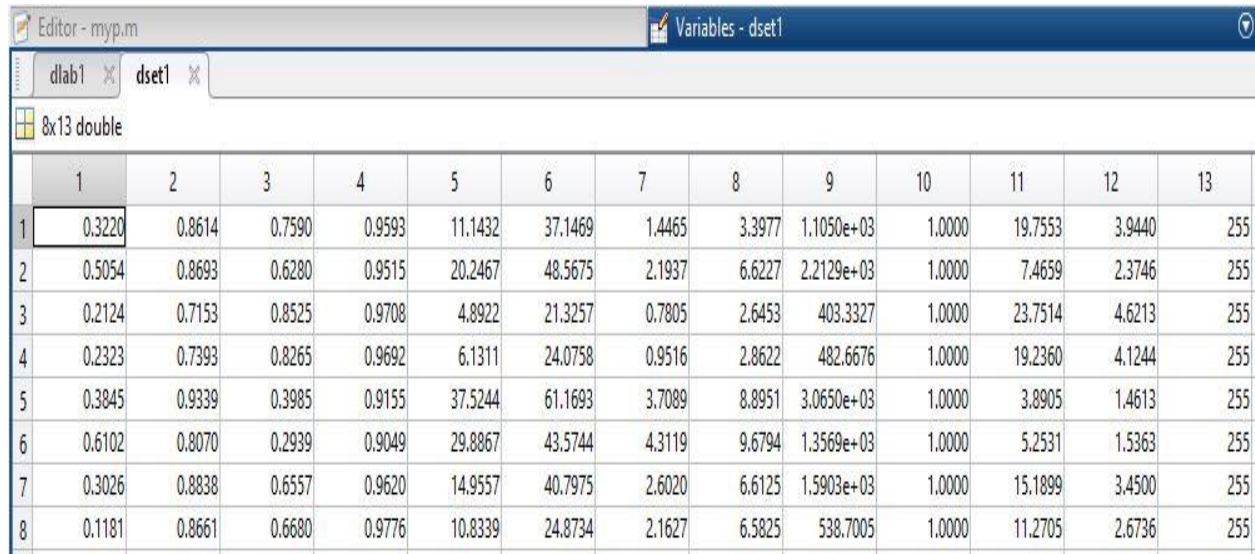
Fig 4.5: SVM Classification

Using Support Vector Machines - As with any supervised learning model, you first train a support vector machine, and then cross validate the classifier. Use the trained machine to classify (predict) new data. In addition, to obtain satisfactory predictive accuracy, you can use various SVM kernel functions, and you must tune the parameters of the kernel functions.

Using the support vector machines first requires the SVM to be trained. The SVM for disease detection was trained with images for diseased Rice crop plants. Here the images for Rice Blast disease and Brown Spot disease are downloaded. And the features were extracted for the images. And the entire data was stored in dataset which has features of images. These images are called Training Images. This entire procedure is called as Training the SVM.

The function used is: `svmStructDisease = svmtrain(our,dt);`

where `our` is the dataset containing image information of features. '`dt`' is the set containing set of diseases respectively.



The screenshot shows a MATLAB Editor window with a file named 'myp.m'. Below the editor, the 'Variables' pane displays a variable 'dset1' of type '8x13 double'. The data is presented in a table with 13 columns and 8 rows.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0.3220	0.8614	0.7590	0.9593	11.1432	37.1469	1.4465	3.3977	1.1050e+03	1.0000	19.7553	3.9440	255
2	0.5054	0.8693	0.6280	0.9515	20.2467	48.5675	2.1937	6.6227	2.2129e+03	1.0000	7.4659	2.3746	255
3	0.2124	0.7153	0.8525	0.9708	4.8922	21.3257	0.7805	2.6453	403.3327	1.0000	23.7514	4.6213	255
4	0.2323	0.7393	0.8265	0.9692	6.1311	24.0758	0.9516	2.8622	482.6676	1.0000	19.2360	4.1244	255
5	0.3845	0.9339	0.3985	0.9155	37.5244	61.1693	3.7089	8.8951	3.0650e+03	1.0000	3.8905	1.4613	255
6	0.6102	0.8070	0.2939	0.9049	29.8867	43.5744	4.3119	9.6794	1.3569e+03	1.0000	5.2531	1.5363	255
7	0.3026	0.8838	0.6557	0.9620	14.9557	40.7975	2.6020	6.6125	1.5903e+03	1.0000	15.1899	3.4500	255
8	0.1181	0.8661	0.6680	0.9776	10.8339	24.8734	2.1627	6.5825	538.7005	1.0000	11.2705	2.6736	255

Fig 4.6: SVM Dataset

The next step is to classify the images. The function used is:

`svmclassify(svmStructDisease,feat_disease)`

`feat_disease` is the array which contains the features of the selected image that is used for testing. On selecting appropriate cluster, the disease is detected. And help dialog is displayed which shows the disease that was affected.

The SVM is classified and trained for Rice plant. Two disease plant features are entered in to the dataset. Similar can be extended to multiple diseases using the Multi SVM.

Multiclass SVM aims to assign labels to instances by using support vector machines, where the labels are drawn from a finite set of several elements.

The dominant approach for doing so is to reduce the single multiclass problem into multiple binary classification problems. Common methods for such reduction include:

- Building binary classifiers which distinguish
 - (i) Between one of the labels and the rest (*one-versus-all*) or
 - (ii) Between every pair of classes (*one-versus-one*).

Classification of new instances for the one-versus-all case is done by a winner-takes-all strategy, in which the classifier with the highest output function assigns the class (it is important that the output functions be calibrated to produce comparable scores). For the one-versus-one approach, classification is done by a max-wins voting strategy, in which every classifier assigns the instance to one of the two classes, then the vote for the assigned class is increased by one vote, and finally the class with the most votes determines the instance classification.

- Directed acyclic graph SVM
- Error-correcting output codes

Crammer and Singer proposed a multiclass SVM method which casts the multiclass classification problem into a single optimization problem, rather than decomposing it into multiple binary classification problems.

In the current proposed system for detection of diseases we have created a dataset of multiple images consisting of all thirteen features extracted through feature extraction. A training set of multiple images was made and a label set containing of integer values for corresponding disease type was created. On matching of maximum features a particular type of disease is detected which is displayed through help dialog.

4.3 UML DIAGRAMS

4.3.1 USE CASE DIAGRAM

A behavioral diagram that shows a set of objects and their relationships.

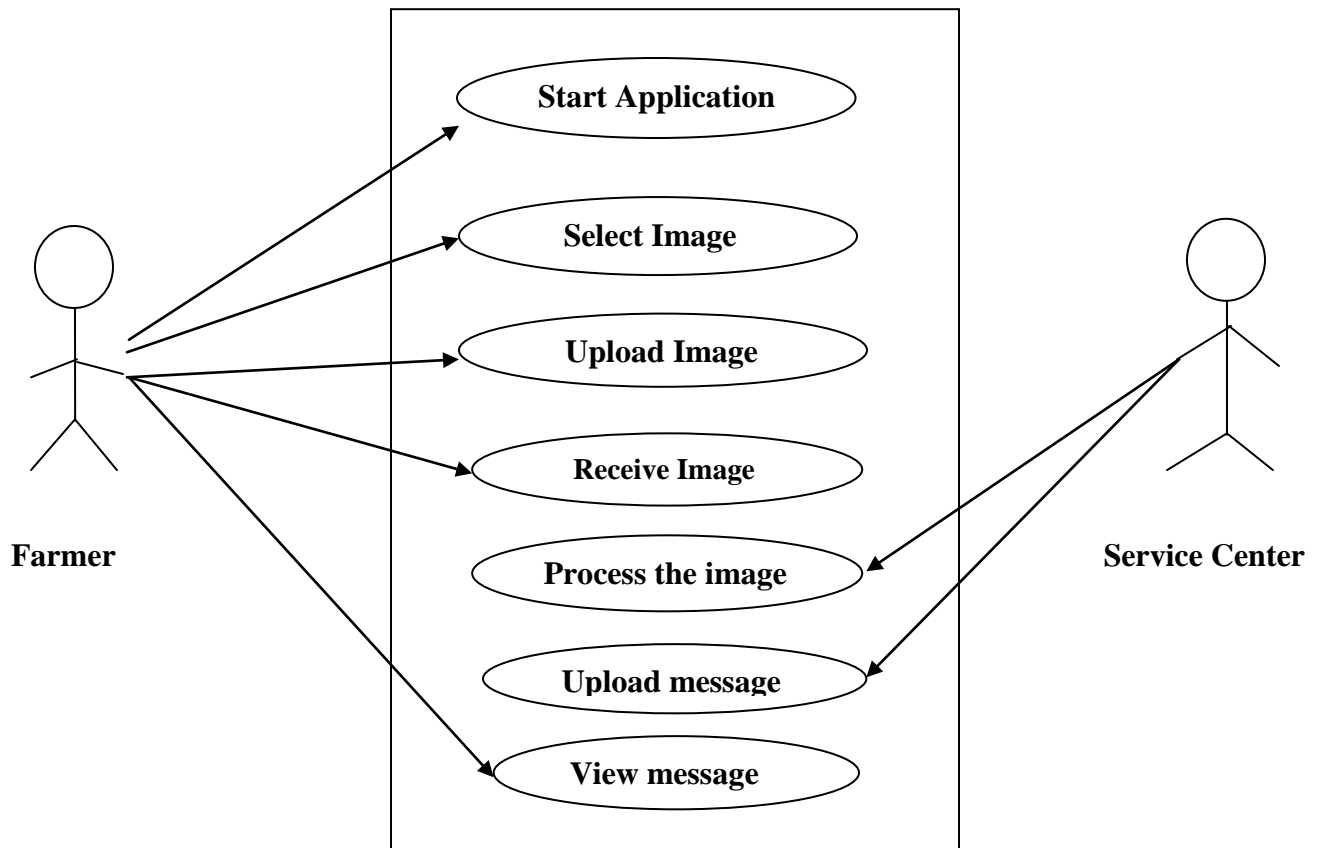


Fig 4.7: Use Case Diagram

The above diagram represents the actors, Farmer and Service Center. The Use cases which drive the process to completion are Start Application, Select Image, Upload Image, Receive Image, Process the Image, Upload Message, View message.

4.3.2 SEQUENCE DIAGRAM

A behavioral diagram that shows an interaction, emphasizing the time ordering of messages.

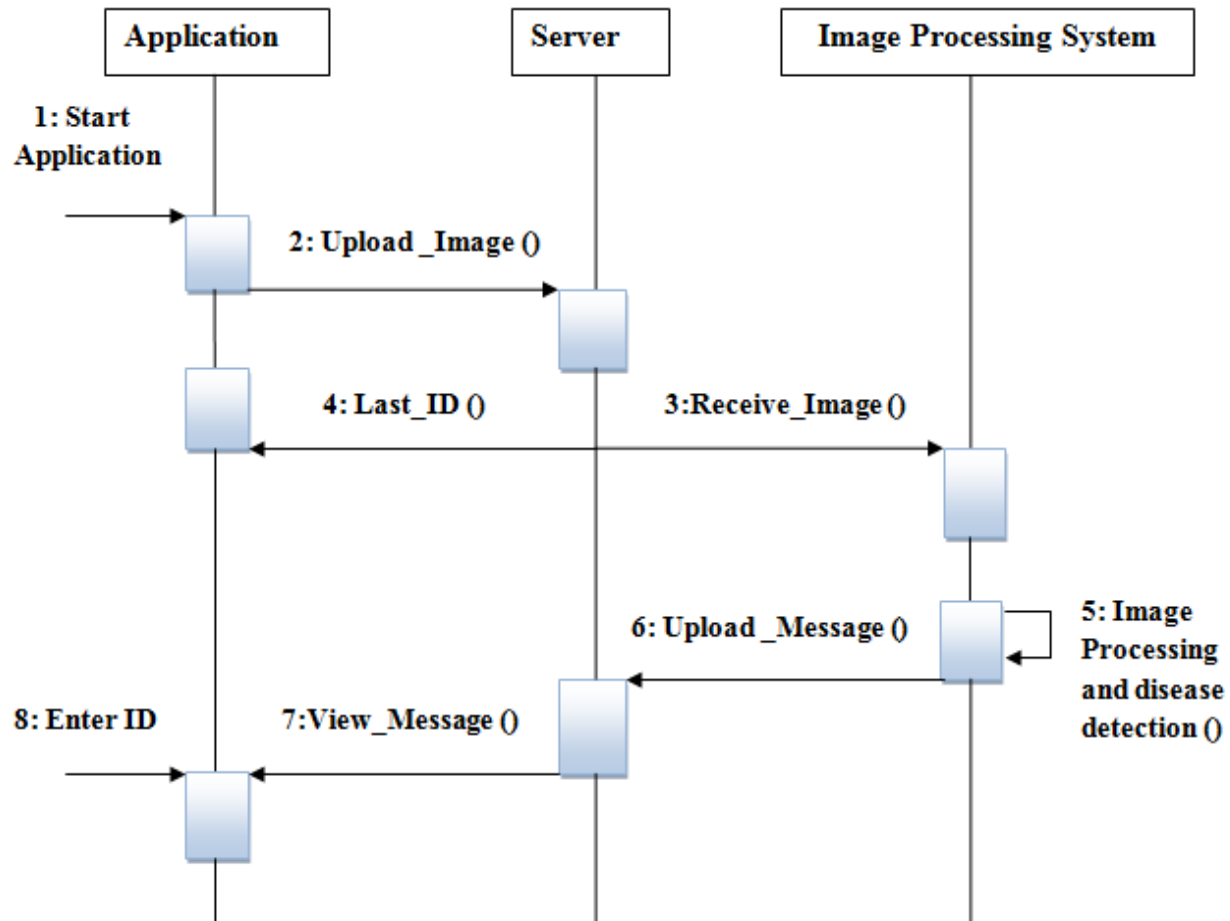


Fig 4.8: Sequence Diagram

The above diagram represents interaction between various components. Application that is installed on farmer's phone is used to upload the image of crop. An Auto incremented ID is sent back to the farmer. The image received is processed and the message is uploaded. Using the ID the farmer retrieves the message.

CHAPTER 5

CODE IMPLEMENTATION

5.1 ANDROID CODE

5.1.1 LAYOUT CODE

This code module includes the necessary part of code that is used to create the layout of the android application developed. It consists of 3 buttons and 1 image view component. The 3 buttons are used for choosing the image, uploading the image and viewing the message respectively.

Activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/string_choose_file"
        android:id="@+id/buttonChoose" />
```

<ImageView

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_weight="1"  
    android:id="@+id/imageView" />
```

<Button

```
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/string_upload_image"  
    android:id="@+id/buttonUpload" />
```

<Button

```
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="View Message"  
    android:id="@+id/buttonViewMessage" />
```

</LinearLayout>

5.1.2 MAIN ACTIVITY

This code module includes the functionality of uploading the image from the android application developed. It also gives an overview of what functions have to be performed on clicking the particular buttons in the android application.

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
    public static final String UPLOAD_URL = "http://pcode.16mb.com/upload.php";  
    private void uploadImage(){  
        class UploadImage extends AsyncTask<Bitmap,Void,String>{  
            ProgressDialog loading;  
            RequestHandler rh = new RequestHandler();
```

@Override

```
    protected void onPreExecute() {  
        super.onPreExecute();  
        loading = ProgressDialog.show(MainActivity.this, "Uploading Image", "Please  
wait...",true,true);  
    }
```

@Override

```
    protected void onPostExecute(String s) {  
        super.onPostExecute(s);  
        loading.dismiss();  
        Toast.makeText(getApplicationContext(),s,Toast.LENGTH_LONG).show();  
    }
```

@Override

```
    protected String doInBackground(Bitmap... params) {  
        Bitmap bitmap = params[0];
```



```
String uploadImage = getStringImage(bitmap);
HashMap<String,String> data = new HashMap<>();
data.put(UPLOAD_KEY, uploadImage);
String result = rh.sendPostRequest(UPLOAD_URL,data);
return result;
}
}
```

```
UploadImage ui = new UploadImage();
ui.execute(bitmap);
}
```

@Override

```
public void onClick(View v) {
if (v == buttonChoose) {
    showFileChooser();
}
if(v == buttonUpload){
    uploadImage();
}
if(v == buttonView){
    viewImage();
}
}
```

```
private void viewImage() {
    startActivity(new Intent(this, ViewImage.class));
}
}
```

5.1.3 VIEW MESSAGE

This code module describes the main functionality of the application where the farmer will be able to retrieve the message uploaded in the server. The getdata function helps in accessing the message information from the server in the android application and displaying it to the farmer in the android application.

ViewMessage.java

```
public class ViewImage extends AppCompatActivity implements View.OnClickListener {
```

```
    private void getData() {
```

```
        String id = editTextId.getText().toString().trim();
```

```
        if (id.equals("")) {
```

```
            Toast.makeText(this, "Please enter an id", Toast.LENGTH_LONG).show();
```

```
            return;
```

```
        }
```

```
        loading = ProgressDialog.show(this, "Please wait...", "Fetching...", false, false);
```

```
        String url = Config.DATA_URL+editTextId.getText().toString().trim();
```

```
        StringRequest stringRequest = new StringRequest(url, new Response.Listener<String>() {
```

```
            @Override
```

```
            public void onResponse(String response) {
```

```
                loading.dismiss();
```

```
                showJSON(response);
```

```
            }
```

```
        },
```

```
        new Response.ErrorListener() {
```

```
            @Override
```

```
            public void onErrorResponse(VolleyError error) {
```

```
try {

    JSONObject jsonObject = new JSONObject(response);
    JSONArray result = jsonObject.getJSONArray(Config.JSON_ARRAY);
    JSONObject collegeData = result.getJSONObject(0);
    ID = collegeData.getString(Config.KEY_ID);
    CropName = collegeData.getString(Config.KEY_CN);
    AffectedDisease = collegeData.getString(Config.KEY_AD);
    AffectedArea = collegeData.getString(Config.KEY_AA);
    Pesticide = collegeData.getString(Config.KEY_Pest);
    AmountOfPesticide = collegeData.getString(Config.KEY_AP);
}

catch (JSONException e) {
    e.printStackTrace();
}

textViewResult.setText("ID:\t"+ID+"\n\nCrop Name:\t" +CropName+ "\n\nAffected
Disease:\t"+ AffectedDisease+"\n\nAffected Area:\t"+AffectedArea+"\n\nPesticide:\t"
+Pesticide+"\n\nAmount Of Pesticide:\t" +AmountOfPesticide);
}

@Override
public void onClick(View v) {
    getData();
}
```

5.2 PHP CODE

5.2.1 UPLOAD

This code part is used to establish connection with the server and upload the image to the server via the android application that is developed. The necessary queries are written to achieve this functionality as shown in the following code. Function to display the uploaded image ID is also included in this code.

upload.php

<?php

```
if($_SERVER['REQUEST_METHOD']=='POST'){
$image = $_POST['image'];
require_once('dbConnect.php');
$sql = "INSERT INTO images (image) VALUES (?)"
$stmt = mysqli_prepare($con,$sql);
mysqli_stmt_bind_param($stmt,"s",$image);
mysqli_stmt_execute($stmt);
$check = mysqli_stmt_affected_rows($stmt);
if($check == 1){
echo "Successfull";
$last_id = $con->insert_id;
echo "ID is: " . $last_id;
}else{
echo "Error Uploading Image";
}
mysqli_close($con);
}else{
echo "Error";
}
```

?>

5.2.2 GET MESSAGE

This module includes the part of the code that is required to get the information regarding the crop from the server. The query helps us in retrieving the message from the server by accessing every attribute of the message table.

getmessage.php

<?php

```
if($_SERVER['REQUEST_METHOD']=='GET'){  
    $id = $_GET['id'];  
    require_once('dbConnect.php');  
    $sql = "SELECT * FROM message WHERE id='".$id."'";  
    $r = mysqli_query($con,$sql);  
    $res = mysqli_fetch_array($r);  
    $result = array();  
    array_push($result,array(  
        "ID"=>$res['ID'],  
        "CropName"=>$res['Crop Name'],  
        "AffectedDisease"=>$res['Affected Disease'],  
        "AffectedArea"=>$res['Affected Area'],  
        "Pesticide"=>$res['Pesticide Name'],  
        "AmountOfPesticide"=>$res['Amount Of Pesticide']  
    )  
    );  
    echo json_encode(array("result"=>$result));  
    mysqli_close($con);  
    }  
?>
```

5.3 MATLAB CODE

This code module plays a major role in plant disease detection. The GUI created includes 3 buttons for selecting image, testing image and cancelling the application. The functionalities for those buttons are added accordingly. SVM classifier is used for classification of the plant disease. This module calls the mutlisvm function to get the results appropriately.

myp.m

```
[cluster_idx, cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean', ...
                                     'Replicates',3);
pixel_labels = reshape(cluster_idx,nrows,ncols);
segmented_images = cell(1,3);
rgb_label = repmat(pixel_labels,[1,1,3]);

for k = 1:nColors
    colors = a;
    colors(rgb_label ~= k) = 0;
    segmented_images{k} = colors;
end

figure,subplot(2,3,2);imshow(a);title('Original Image');
subplot(2,3,4);imshow(segmented_images{1});title('Cluster 1');
subplot(2,3,5);imshow(segmented_images{2});title('Cluster 2');
subplot(2,3,6);imshow(segmented_images{3});title('Cluster 3');
set(gcf, 'Position', get(0,'Screensize'));
set(gcf, 'name','Segmented by K Means', 'numbertitle','off')
x = inputdlg('Enter the cluster no. containing the ROI only:');
i = str2double(x);
seg_img = segmented_images{i};
```

```
if ndims(seg_img) == 3
    img = rgb2gray(seg_img);
end

black = im2bw(seg_img,graythresh(seg_img));
m = size(seg_img,1);
n = size(seg_img,2);
zero_image = zeros(m,n);
cc = bwconncomp(seg_img,6);
diseasedata = regionprops(cc,'basic');
A1 = diseasedata.Area;
sprintf('Area of the disease affected region is : %g%',A1);
I_black = im2bw(a,graythresh(a));

kk = bwconncomp(a,6);
leafdata = regionprops(kk,'basic');
A2 = leafdata.Area;
sprintf(' Total leaf area is : %g%',A2);
Affected_Area = (A1/A2);

if Affected_Area < 0.1
    Affected_Area = Affected_Area+0.15;
end

sprintf('Affected Area is: %g%%',(Affected_Area*100))
Affect = Affected_Area*100;

% Create the Gray Level Cooccurrence Matrices (GLCMs)
glcms = graycomatrix(img);
stats = graycoprops(glcms,'Contrast Correlation Energy Homogeneity');
Contrast = stats.Contrast;
```

```
Correlation = stats.Correlation;
Energy = stats.Energy;
Homogeneity = stats.Homogeneity;
Mean = mean2(seg_img);
Standard_Deviation = std2(seg_img);
Entropy = entropy(seg_img);
RMS = mean2(rms(seg_img));

%Skewness = skewness(img)
Variance = mean2(var(double(seg_img)));
a = sum(double(seg_img(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(seg_img(:)));
Skewness = skewness(double(seg_img(:)));
m = size(seg_img,1);
n = size(seg_img,2);
in_diff = 0;

for i = 1:m
    for j = 1:n
        temp = seg_img(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
    end
end

IDM = double(in_diff);

feat_disease = [Contrast,Correlation,Energy,Homogeneity, Mean, Standard_Deviation, Entropy,
RMS, Variance, Smoothness, Kurtosis, Skewness, IDM];
load('training1.mat')
result = multisvm(dset1,dlab1,feat_disease);
```



```
if result == 0
    helpdlg('RiceBlast ');

elseif result == 1
    helpdlg(' Brownsport ');

elseif result == 2
    helpdlg(' Sheath Blight ');

elseif result == 3
    helpdlg(' Stemrot ');

end
guidata(hObject,handles);
```

CHAPTER 6

TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:

- meets the requirements that guided its design and development,
- responds correctly to all kinds of inputs,
- performs its functions within an acceptable time,
- is sufficiently usable,
- can be installed and run in its intended environments, and
- Achieves the general result its stakeholder's desire.

As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs (errors or other defects). The job of testing is an iterative process as when one bug is fixed; it can illuminate other, deeper bugs, or can even create new ones.

Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.

Software testing can be conducted as soon as executable software (even if partially complete) exists. The overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system

requirements have been defined and then implemented in testable programs. In contrast, under an Agile approach, requirements, programming, and testing are often done concurrently.

6.1 THE MAIN AIM OF TESTING

The main aim of testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources and running in different operating environments.

In this project, we have developed a Android Application and a Image Processing code which helps in detection of disease. The main aim of testing this project is to check if the disease is getting detected accurately and check the working performance when different images are given as inputs.

The testing steps are:

- Unit Testing.
- Integration Testing.
- Validation Testing.
- User Acceptance Testing.
- Output Testing

6.2 UNIT TESTING:

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs.

The following Unit Testing Table 7.1 shows the functions that were tested at the time of programming. The first column gives all the modules which were tested, and the second column gives the test results. Test results indicate if the functions, for given inputs are delivering valid outputs.

| Function Name | Tests Results |
|-------------------|--|
| Uploading Image | Tested for uploading different types and sizes of images. |
| Detecting Disease | Tested for different images of Rice plant leaves and diseases Rice Blast and Brown Spot. |
| Get Message | Tested if the message is displayed successfully. |

Table 6.1: Unit Testing Table

6.3 INTEGRATION TESTING:

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

6.4 VALIDATION TESTING

At the culmination of integration testing, software is completed assembled as a package. Interfacing errors have been uncovered and corrected. Validation testing can be defined in many ways; here the testing validates the software function in a manner that is reasonably expected by the customer.

In software project management, software **testing**, and software engineering, verification and **validation** (V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. The following table indicates validation tests done for checking functionality of the project after its integration with the front-end.

| Functionality to be tested | Input | Tests Results |
|-------------------------------|--|---|
| Working of Choose File option | User convenience to access images stored | Different folders where images are stored can be uploaded |
| Working of View Message | User enters the ID given to him | Details of disease are displayed. |

Table 6.2: Validation Testing Table

6.5 USER ACCEPTANCE TESTING

Performance of an acceptance test is actually the user's show. User motivation and knowledge are critical for the successful performance of the system. The above tests were conducted on the newly designed system performed to the expectations. All the above testing strategies were done using the following test case designs.

6.5.1 WHITE BOX TESTING

White Box Testing sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using White Box Testing, we can derive test cases that:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decision on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.

6.5.2 BLACK BOX TESTING

Black Box Testing focuses on the functional requirements of the software. This enables us to derive sets of input conditions that will fully exercise all functional requirements for a program. Black Box testing attempts to find errors in the following categories:

- Incorrect or Missing Functions.
- Interface errors.
- Performance errors.
- Initialization and Termination errors.

CHAPTER 7

SNAPSHOTS

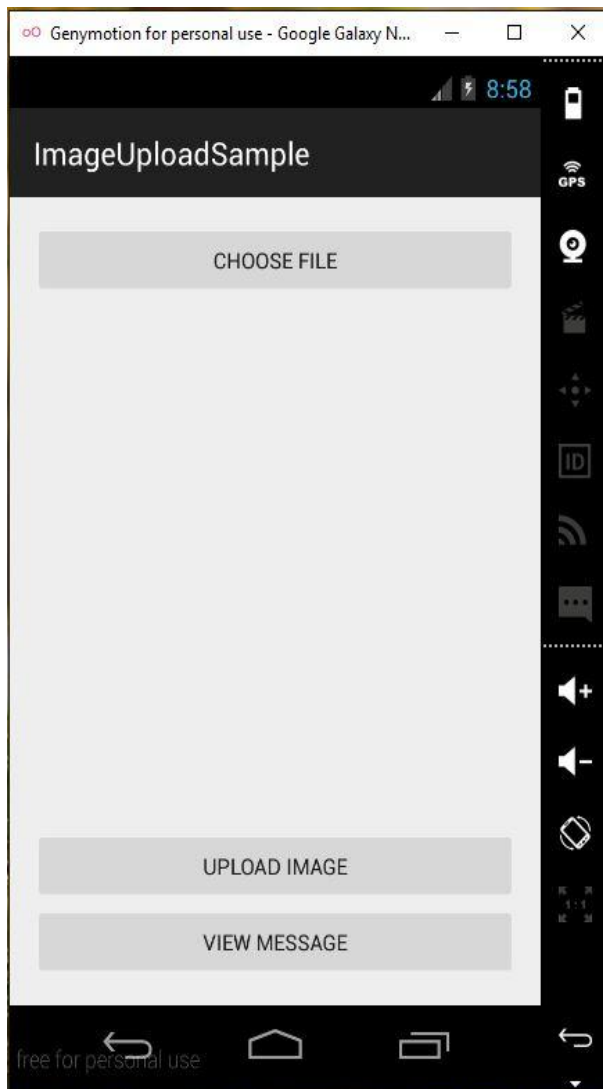


Fig 7.1 Application Layout

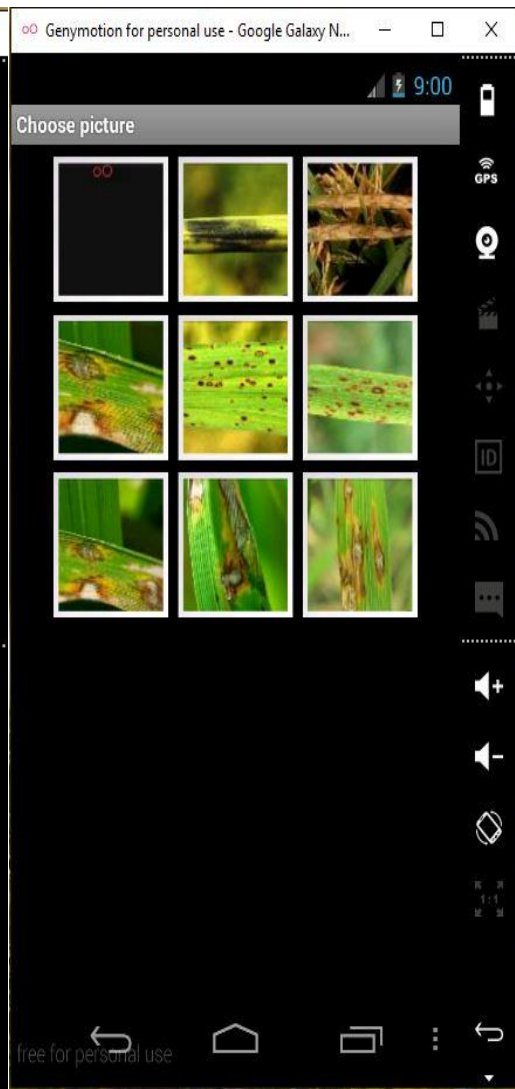


Fig 7.2: Gallery Images

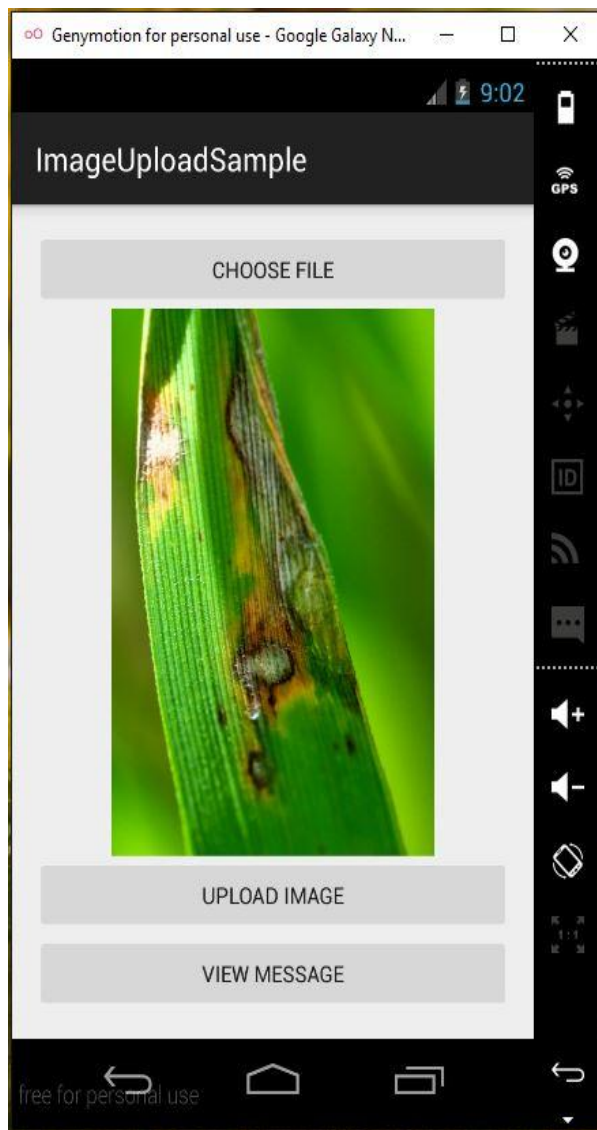


Fig 7.3:Image Selected and Displayed

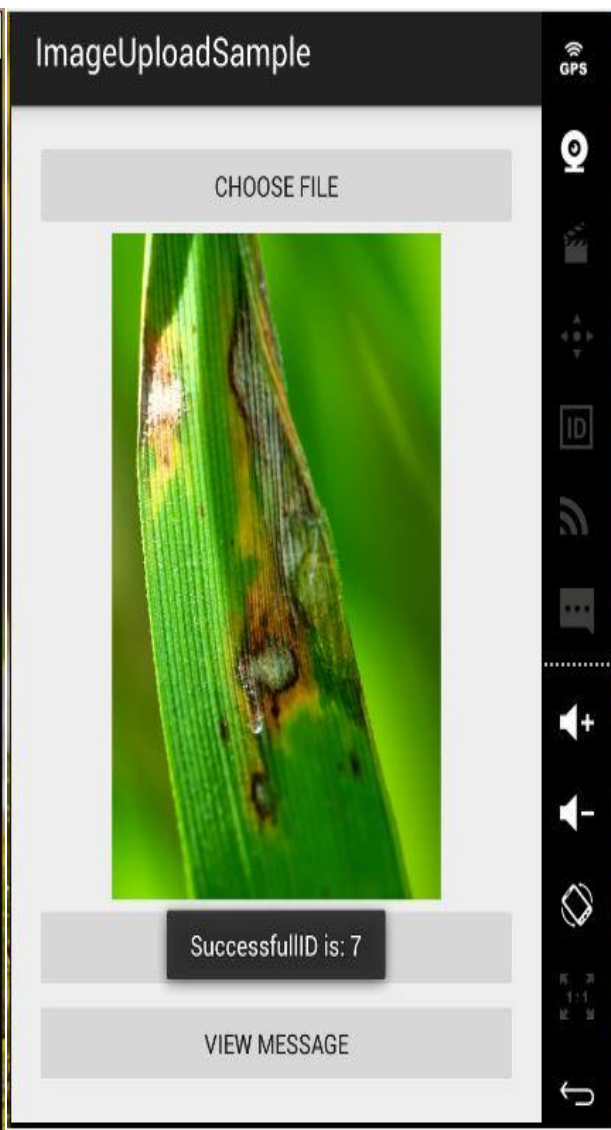


Fig 7.4: Image uploaded and ID displayed

localhost » u360185672_pdata » images

Browse Structure SQL Search Insert Export Import

✓ Showing rows 0 - 6 (7 total, Query took 0.0204 sec)

```
SELECT *
FROM `images`
LIMIT 0 , 30
```

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Sort by key: None

+ Options

| | ID | image |
|---|----|--------------------|
| <input type="checkbox"/> Edit Copy Delete | 1 | [BLOB - 255.3 KiB] |
| <input type="checkbox"/> Edit Copy Delete | 2 | [BLOB - 400.2 KiB] |
| <input type="checkbox"/> Edit Copy Delete | 3 | [BLOB - 268.8 KiB] |
| <input type="checkbox"/> Edit Copy Delete | 4 | [BLOB - 57.5 KiB] |
| <input type="checkbox"/> Edit Copy Delete | 5 | [BLOB - 12.5 KiB] |
| <input type="checkbox"/> Edit Copy Delete | 6 | [BLOB - 124 KiB] |
| <input type="checkbox"/> Edit Copy Delete | 7 | [BLOB - 664.2 KiB] |

Check All / Uncheck All With selected: Change Delete Export

Show : Start row: 0 Number of rows: 30 Headers every 100 rows

Fig 7.5 : Image table in server where the images are uploaded

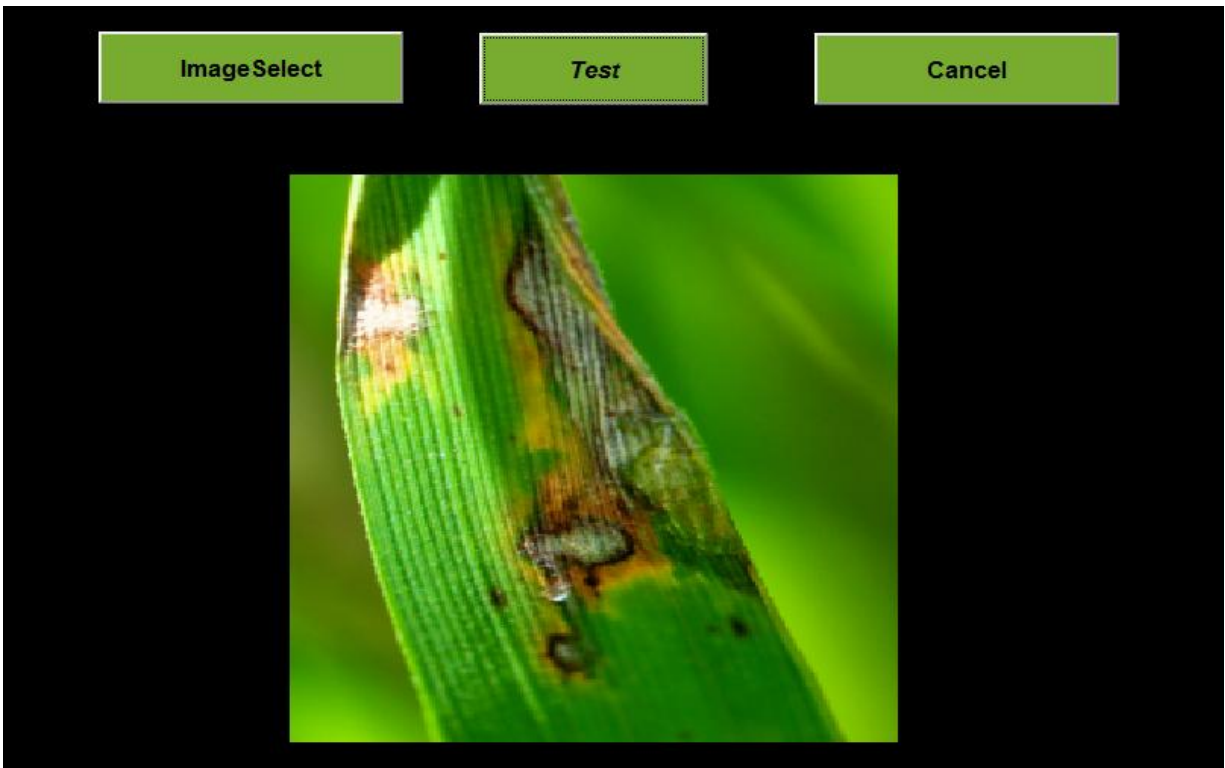


Fig 7.6: Matlab User Interface where the selected image is displayed

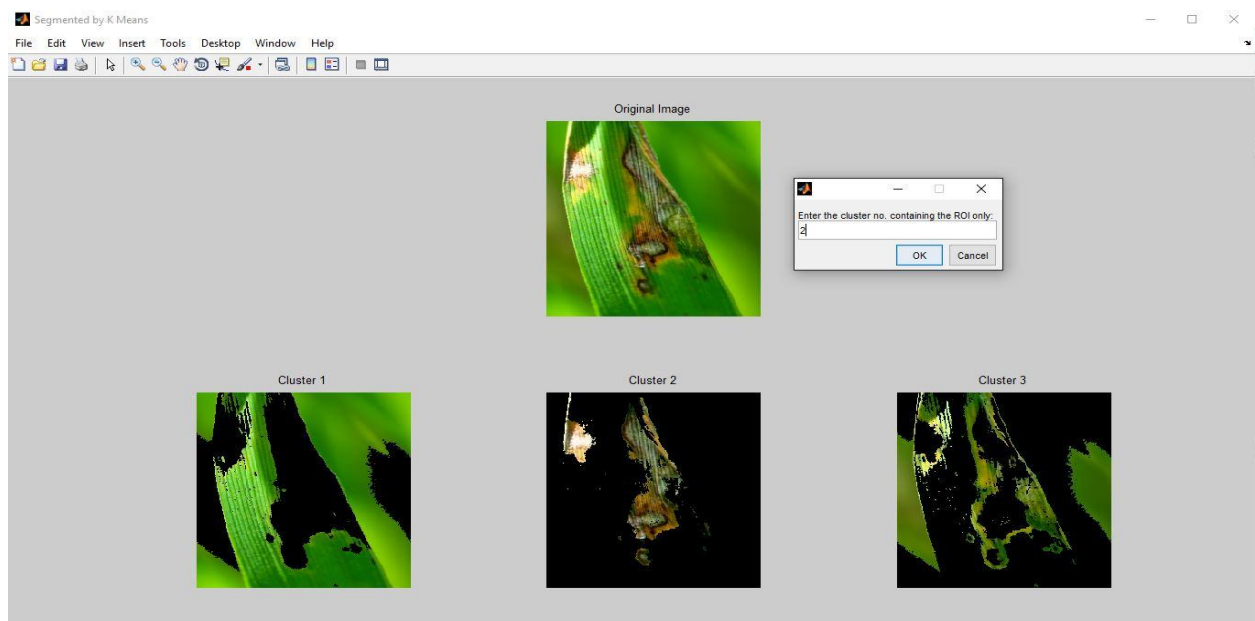


Fig 7.7: Clustering of the Input Image

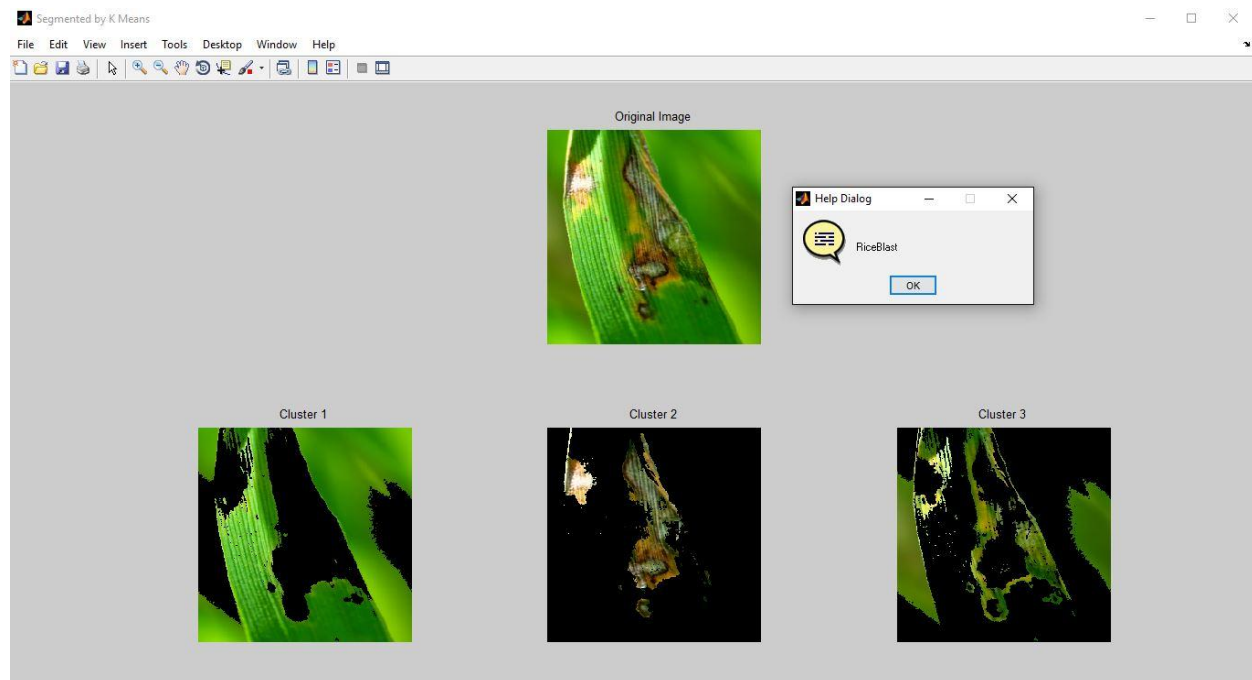


Fig 7.8: Help Dialog displaying the Crop Disease Name

```
Command Window

>> myp

ans =

Affected Area is: 15.2721%

fx >>
```

Fig 7.9: Matlab Command prompt where the affected area is displayed

localhost » u360185672_pdata » message

Browse Structure SQL Search Insert Export Import Operations Tracking

Showing rows 0 - 7 (8 total, Query took 0.0002 sec)

```
SELECT *
FROM 'message'
LIMIT 0 , 30
```

Profiling [Inline] [Edit] [Exp

Show: Start row: 0 Number of rows: 30 Headers every 100 rows

+ Options

| | ID | Crop Name | Affected Disease | Affected Area | Pesticide Name | Amount Of Pesticide |
|---|----|------------------|---------------------------|---------------|--------------------------------|-----------------------|
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 1 | Rice ರೈಸ್ (ಭತ್ತ) | Leaf Blast ಲೀಫ್ ಬ್ಲಾಸ್ಟ್ | 15.2555% | Tricyclazole ತ್ರಿಸೈಕ್ಲೋಜೋಲೆ | 75 WP @ 2 g/kg |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 2 | Rice ರೈಸ್ (ಭತ್ತ) | Brown Spot ಬ್ರೌನ್ ಸ್ಪಾಟ್ | 15.0732% | Mancozeb ಮಂಕೊಜೆಬ್ | 75 WP @ 2.5 g/litre |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 3 | Rice ರೈಸ್ (ಭತ್ತ) | Stem Rot ಸ್ಟೆಮ್ ರಾಟ್ | 17.3926% | Isoprothiolane ಈಸೋಪ್ರೋಥಿಯೋಲೇನ್ | 40 EC @ 1.5 ml/litre. |
| <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete | 4 | Rice ರೈಸ್ (ಭತ್ತ) | Sheath Blight ಶೀತ್ ಬ್ಲೈಟ್ | 16.1347% | Validamycin ವಾಲಿಡಾಮೈಸಿನ್ | 3 L @ 2.5 ml/litre |

Fig 7.10: Message table in the server where the message contents are uploaded

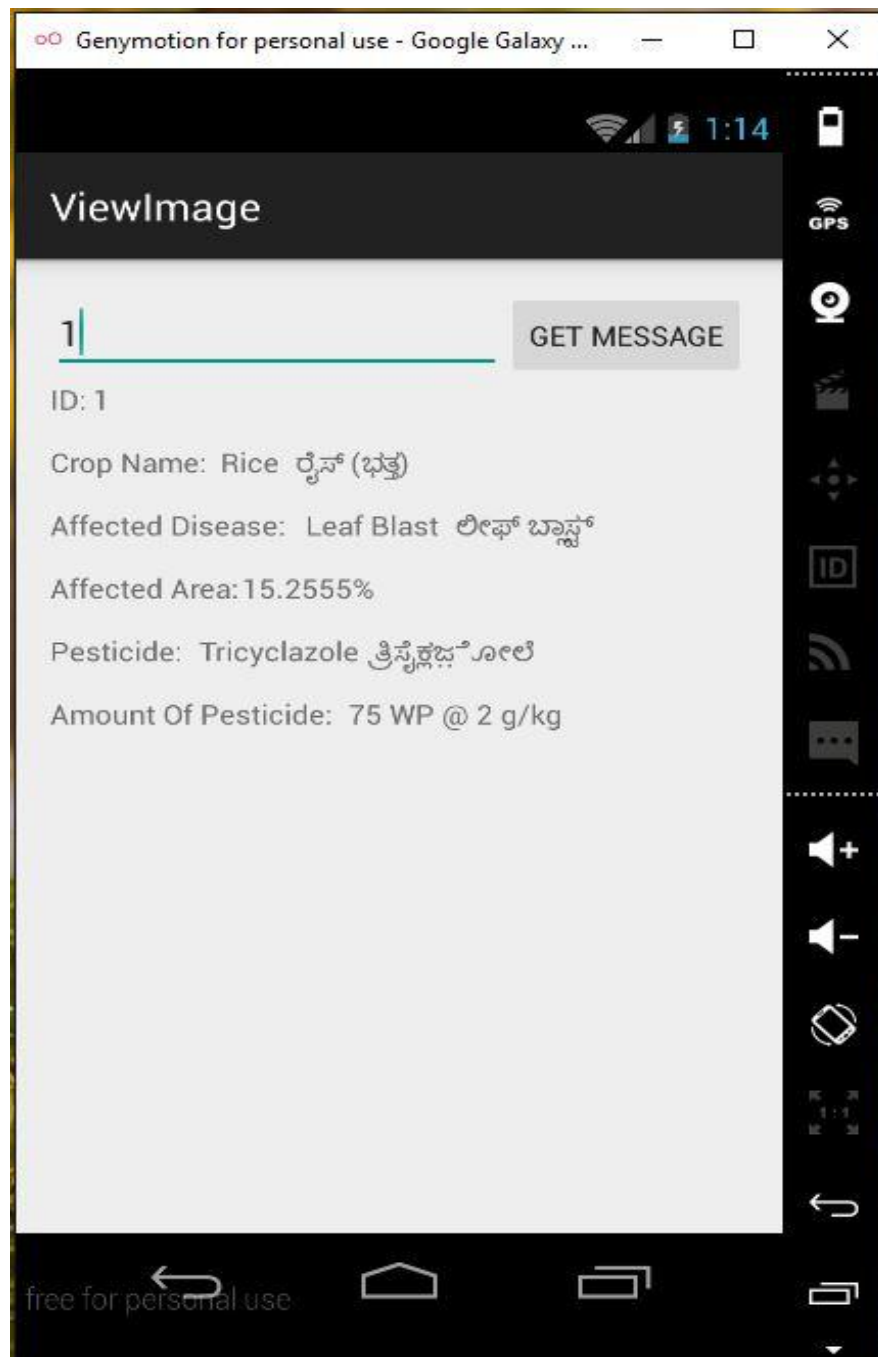


Fig 7.11: Application Displaying the Message

CONCLUSION

An application of detecting the plant diseases and providing the necessary suggestions for the disease has been implemented. Hence the proposed objective was implemented on three different types of crops namely Rice, Sugarcane and Cotton. The diseases specific to these plants were considered for testing of the algorithm. The experimental results indicate the proposed approach can recognize the diseases with a little computational effort. By this method, the plant diseases can be identified at the initial stage itself and the pest control tools can be used to solve pest problems while minimizing risks to people and the environment.

In order to improve disease identification rate at various stages, the training samples can be increased with the optimal features given as input condition for disease identification and fertilization management of the crops. As a part of Future Enhancement the complete process described in this project can be automated so that the result can be delivered in a very short time.

PUBLICATION DETAILS

1. Romana Tazeen, Shilpa H N, Usha P, Mrs. Jayanthi M G, Dr. Shashikumar D R, "Image Processing System for Fertilization Management of Crops", communicated to ICCIT- 2016 to be held on May 20th and 21st at Cambridge Institute Of Technology.
2. Romana Tazeen, Shilpa H N, Usha P, Mrs. Jayanthi M G, Dr. Shashikumar D R, "Image Processing System for Fertilization Management of Crops", communicated to International Journal of Engineering Research(IJER) - 2016 for publication.

REFERENCES

- [1] S. Arivazhagan, R. Newlin Shebiah*, S. Ananthi, S. Vishnu Varthini March 2013. Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features. *Agric Eng Int: CIGR Journal* Vol. 15, No.1 211
- [2] Al-Bashish, D., M. Braik, and S. Bani-Ahmad. 2011. Detection and classification of leaf diseases using K-means-based segmentation and neural networks based classification. *Information Technology Journal*, 10(2): 267-275
- [3] Al-Hiary, H., S. Bani-Ahmad, M. Reyalat, M. Braik, and Z. AlRahamneh. 2011. Fast and accurate detection and classification of plant diseases. *International Journal of Computer Applications*, 17(1): 31-38
- [4] Argenti, F., L. Alparone, and G. Benelli. 1990. Fast algorithms for texture analysis using co-occurrence matrices. *IEEE proceedings*, 137, (6): 443-448
- [5] Arivazhagan, S., R. N. Shebiah, S. S. Nidhyandhan, and L. Ganesan. 2010. Classification of citrus and non-citrus fruits using texture features. *Computing Communication and Networking Technologies, ICCCNT-2010*
- [6] Bauer, S. D., F. Korc, W. Forstner. 2011. The potential of automatic methods of classification to identify leaf diseases from multispectral images. *Precision Agriculture*, 12: 361-377
- [7] Chamasemani, F. F., and Y. P. Singh. Malaysia, Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference. Faculty of Information Technology, MultiMedia University. Cyberjaya, Malaysia
- [8] Chih-Wei, H., and C. Lin. 2002. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2): 415-425
- [9] Helly, M. E., A. Rafea, and Salwa-El-Gammal. 2003. An integrated image processing system for leaf disease detection and diagnosis. in *Proc. IICAI*, pp.1182-1195
- [10] Jean, W. 2009. Extension plant pathologist, georgia plant disease loss estimates, www.caes.uga.edu/publications
- [11] Jun, W., and S. Wang. 2008. Image thresholding using weighted parzen window estimation. *Journal of Applied Sciences*, 8(5):772-779
- [12] Kim, D. G., T. F. Burks, J. Qin, and D. M. Bulanon. 2009. Classification of grapefruit peel diseases using color texture feature analysis. *International Journal on Agriculture and Biological Engineering*, 2(3): 41-50
- [13] Rastogi, R., and V. K. Chadda. Applications of image processing in biology and agriculture J. K. Sainis, Molecular Biology and Agriculture Division, BARC newsletter
- [14] Siddiqil, M. H., S. Sulaiman, I. Faye, and I. Ahmad. 2009. A real time specific weed discrimination system using multi-level wavelet decomposition. *International Journal of Agriculture and Biology*, 11(5): 559-565
- [15] Gonzalez, R., R. E. Woods. 2008 *Digital image processing*. Third edition, Pearson Education, Prentice-Hall, Inc