Leaf Disease Identification System

(Android-Based)

Submitted by: Abdur Rehman

Supervised by: Dr. Khalid Saleem

Department of Computer Sciences

Quaid-i-Azam University, Islamabad

Dedicated to my beloved parents and family. For their love, endless support, encouragement and sacrifices. Special thanks to my best friend Danish Alam, Amad Iftikhar, Mohammad Usman and Ishtiaq Ismail.

ABSTRACT

The basic idea of this report is to translate the mechanism of diagnosing the specific diseases of wheat plants (Leaf Rust, Stripe Rust) into a software application, that enables users to diagnose various diseases and collection of data at a single site and develop disease resistant varieties. The traditional process of collecting data by visiting the fields in the whole country is not suitable.

The project begins by studying and understanding the process of diagnosing diseases in the real-life scenarios, so that a system could be built that fulfills user needs. This project is based on image-processing and uses Android-Studio as the main tool for development purpose.

The idea is to take image with the help of mobile phone. The system will do three things i.e. 1) Process the image of leaf on the mobile for detecting any disease area with level of disease severity.2) Compute leaf size and 3) Send the image and results to the server.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

Chapter 1 Introduction

1.1 Introduction

This section throughs light on the techniques used and the problem to be solved by the system.

1.1.1 Digital Image Processing

An image may be defined as a two-dimensional function f (x, y) where f is the intensity or gray level of the image at specific spatial coordinates x and y. When x, y and f are all finite and discrete quantities we call the image a digital image. Digital image processing refers to processing digital images by means of a digital computer. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels and pixels. Pixel is the term most widely used to denote the elements of a digital image.

Images can be represented as RGB images where each pixel consists of a separate value for red, green and blue channels. Below shown representation shows an example of RGB image representation. Each pixel is represented as pixel (R, G, B)

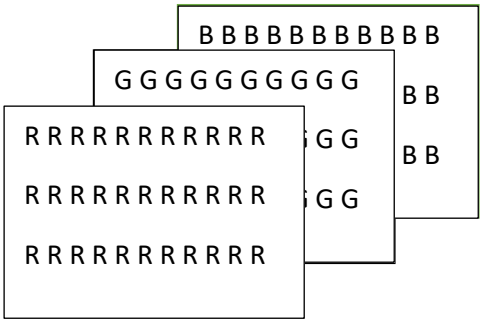


Figure 1.1 RGB format

Images can also be grayscale which is a single channel representation for images. Grayscale pixels values generally range from 0-255.Where 0 represents full- black color and 255 represents full-white color. Below is representation of a grayscale image with imaginary values which may or may not constitute a meaningful image.

Table 1.1 Gray-scale

12	23	0	1
78	234	67	88
81	45	47	23
78	45	35	82

Binary format is also use to represent images. Where each pixel is either 0 (black) or 1 (white).

Table 1.2 Binary

0	1	0	1
1	1	1	1
0	0	1	0
1	0	0	1

1.1.1.1 Basic operations on images

1. Convolution

It is the most common operation in image processing and basis of other operations i.e. smoothing, edge-detection. It is denoted by symbol "*". In a very general sense, convolution is an operation between every part of an image and an operator (kernel). A kernel is essentially a fixed size array of numerical coefficients along with an anchor point in that array, which is typically located at the center. Figure depicts a 3-by-3 convolution kernel with the anchor located at the center.

Table 1.3 Kernel

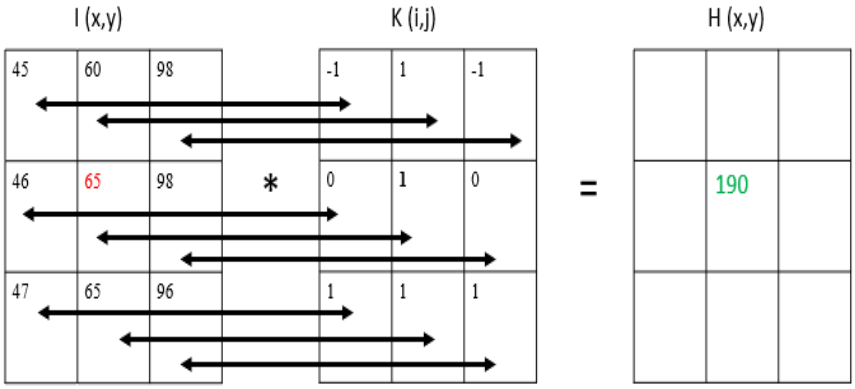
1	-2	1
2	**-4**	2
1	-2	1

The value of the convolution at a particular point is computed by first placing the kernel anchor on top of a pixel on the image with the rest of the kernel overlaying the corresponding local pixels in the image. For each kernel point, we now have a value for the kernel at that point and a value for the image at the corresponding image point. We multiply these together and sum the result; this result is then placed in the resulting image at the location corresponding to the location of the anchor in the input image. This process is repeated for every point in the image by scanning the kernel over the entire image.

We can, of course, express this procedure in the form of an equation. If we define the image to be I (x, y), the kernel to be K (i, j), and the anchor point to be located at (a_i, a_j) in the coordinates of the kernel, then the convolution H (x, y) is defined by the following expression:

$$H(x,y) = \sum_{i=0}^{M_i-1} \sum_{j=0}^{M_j-1} I\,(x+i-a_i, y+j-a_j)K(i,j)$$

Table 1.4 Convolution



2. Smoothing

Smoothing, also called blurring, is a simple and frequently performed operation on images. It is usually done to reduce noise. There are many types of filters that perform smoothing operation, but we will discuss only Gaussian filter.

Gaussian filter, is probably the most useful. Gaussian filtering is done by convolving each point in the input array with a Gaussian kernel and then summing to produce the output array. Gaussian kernel is defined by the following equation:

$$g\,(x,y) = \frac{1}{\sqrt{2\pi}\,\sigma} e(-\ (x^2+y^2)/(2\sigma^2\,))$$

Here "x" is the distance from the origin in the horizontal axis, "y" is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution. Image can be smoothed using equation $S = g * \mathrm{I}$. Here "S" represents smoothed image "g" represents gaussian kernel, "*" represents convolution operation and "I" represents original image.

3. Edge Detection

An edge is a place of rapid change in the image intensity function. The goal of edge detection is to identify sudden changes (discontinuities) in an image. Intuitively, most semantic and shape information from the image can be encoded in the edges. At edges intensity or color changes. There are many edge-detection methods, but we will discuss only Canny edge detector.

Canny edge detector has following steps performed in order:

1. Smooth image with Gaussian filter.
2. Compute derivative of filtered image.
3. Find magnitude and orientation of gradient.
4. Apply "Non-maximum Suppression".

5. Apply “Hysteresis Threshold”.

The first two steps are intuitive and convolution based. The steps which involves gradient magnitude and orientation computation is performed by first calculating gradient of the image. Gradient of an image I (x, y) can be calculated by convolving it with the first derivative of gaussian in x and y directions given by first two equations, magnitude and direction of gradient can be calculated by third and fourth equations respectively.

$$S_x = f_x(x,y) = f(x,y) * \left(\frac{-x}{\sigma^2}\right) e\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$S_y = f_y(x,y) = f(x,y) * \left(\frac{-y}{\sigma^2}\right) e\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

$$magnitude = |S|\sqrt{\left(S_x^2 + S_y^2\right)}$$

$$direction = \theta = \tan^{-1}\frac{S_y}{S_x}$$

Non-maximum suppression is another technique used by Canny to detect edges. Here we use direction of gradient to decide whether a particular pixel is an edge pixel or not. This removes pixels that are not considered to be part of an edge. Hence, only thin lines (candidate edges) will remain. It suppresses the pixels in |S| which are not local maximum.

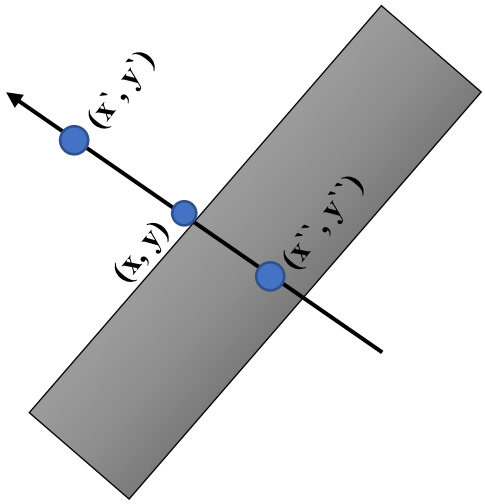


Figure 1.2 Non-Maximum suppression

$$M(x,y) = \begin{cases} |S|(x,y) \;\; if \;\; |S|(x,y) > \;\; |S|(x`,y`) \; and \; if \;\; |S|(x,y) > \;\; |S|(x``,y``) \\ 0 \; otherwise \end{cases}$$

Here |S| (x`, y`) and |S| (x``, y``) are gradient magnitudes on both sides of edge at (x, y) in the direction of gradient as shown in figure.

Hysteresis Threshold is the final step. Canny does use two thresholds (upper and lower):

1. If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge.
2. If a pixel gradient value is below the lower threshold, then it is rejected.
3. If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

1.1.2 Classification

Classification in general means to group things with similar characteristics. The classification problem consists of taking input data-points and deciding which of N classes they belong to, based on training from exemplars (data-points) of each class. The most important point about the classification problem is that it is discrete i.e. each example belongs to precisely one class, and the set of classes covers the whole possible output space. These two constraints are not necessarily realistic sometimes examples might belong partially to two different classes.

A good example of a classification problem is to classify emails as spam and not-spam. The algorithm would need to be train on labeled data set of emails. After training the algorithm could classify any email as spam and not-spam with sufficient accuracy.

1.1.3 Wheat Diseases

Rust diseases are fungal diseases in wheat and other cereal crops and are widely distributed across wheat growing regions. They cause large scale of economic loss. They have natural ability to mutate, such that they can attack previously resistant varieties. Two types of rust diseases which are affecting wheat are following:

- Leaf Rust (Brown Rust)
- Stripe Rust (Yellow Rust)

1. Leaf Rust

Leaf rust is caused by Puccinia triticina. It is also known as brown rust forms orange-brown round pustules on the upper surface of leaves. The pustules are round or slightly elongated. The leaf rust fungus is an airborne whose spores are spread by wind over long distances. Leaf rust mostly found in Faisalabad, Lahore and Okara. Leaf rust shown in figure 1.7.

Figure 1.3 Leaf-Rust

2. Stripe Rust

Stripe rust is caused by Puccinia striiformis. The fungus is dispersed as wind-blown spores which produce new infections. Stripe rust also known as yellow rust forms orange-yellow stripes on the upper surface of leaves. It occurs mostly on upper surface of leaves. Stripe rust mostly found in KPK, Islamabad, Jhelum and Sialkot.

Wind is the main means of the spread of stripe rust. Once the spores become airborne their spread becomes a matter of chance. Most will land on soil or other plants, while some stay airborne until sunlight kills them in a few days. However, they are produced in such high numbers that some land on other living wheat plants. Stripe rust shown in figure 1.8.

Figure 1.4 Stripe-Rust

1.2 Motivation

Project motivation is to make it easy for the researchers to collect plants data for analyzing different diseases. Through this project users will be able to diagnose their plants in real time. This project will help in collecting data at single place for research purpose.

1.3 Contributions

Dr. Umer Masood Qureshi (Assistant Professor, Plant Sciences, QAU)

Zahid Mahmood (Senior Scientific Officer, NARC Islamabad)

Muhammad Saeed (Assistant Research Officer, Wheat research substation, Murree)

Chapter 2 Software Project Management Plan

2.1 Introduction

This section gives an overview of the project and the project deliverables.

2.1.1 Project Overview

This project is about developing a mobile-based application for "Leaf Disease Identification System". Project is based on image-processing using OpenCV (Open source Computer Vision). The images are taken from a researcher at NARC Islamabad (National Agriculture Research Centre) and some are taken from a researcher at Murree. This project is strictly related to disease of leaves and no other portion of plants. We are working on just wheat plant with major focus on two types of diseases (Stripe-Rust and Leaf-Rust) mentioned in the previous chapter.

2.1.2 Project Deliverables

- Software Project Management Plan
- Software Requirements Specification
- Software Design Description
- Software Test Documentation
- Software Implementation

2.2 Project Organization

This section throws light on the "Software Process Model", used for this project. This section also lists "tools and techniques" which are used for the project during analysis, design and implementation stages of the project. This section also highlights the "roles and responsibilities" for this project.

2.2.1 Software Process Model

The software process model for this project is Water-Fall model. As there is no risk factor involved and no change in the requirements of the project, Water-Fall model is good for use.

Because requirements are clear at the start of project and they are not going to change throughout the project, we are using Water-Fall model. This model is usually used for small projects, where requirements are not going to change during the project. For large-scale projects other models are used.

2.2.2 Roles and Responsibilities

The roles of Requirements gathering, Analysis, Design, Implementation and Testing are all to be performed by Abdur-Rehman. It is the responsibility of Abdur-Rehman to devote sufficient time to fulfill these roles under supervision of Dr. Khalid Saleem.

2.2.3 Tools and Techniques

Following are the tools used:

Table 2.1 Tools and techniques

Tool	**Description**
MS-Word	Used for documentation
Project-Libre	Used for making project time table
Microsoft-Visio	Used for drawing diagrams
Star UML	Used for drawing diagram
Android Studio (v 3.1)	For implementation of software product
Android Mobile Phone	Used for running the application
SQLite Database Browser Portable	Used for examining the database part of the android application.

2.3 Project Management Plan

This section contains the description of tasks and deliverables of project. It also includes assignments and project-scheduling

2.3.1 Tasks

This section covers the description of tasks and deliverables that will help in managing the project.

2.3.1.1 Problem Understanding

i. **Description**
   To understand the various diseases that affect leaves of plants and their symptoms.
ii. **Deliverables and Milestones**
   Understanding of various plant diseases is the milestone in this task.
iii. **Resources Needed**
   Information provided by supervisor, internet and Abdur Rehman are the resources needed for this task.
iv. **Dependencies and Constraints**
   Understanding the mechanism of various diseases and their symptoms is time consuming and a constraint for the project.
v. **Risks and Contingencies**
   There are no risks and contingencies involved.

2.3.1.2 Software Project Management Plan

i. **Description**
   This task deals with the development of a plan for managing the project.

ii. Deliverables and Milestones

The milestone of this task includes Project Management Plan.

iii. Resources Needed

"Project Libre", "Microsoft-Word" and Abdur Rehman

iv. Dependencies and Constraints

Problem should be well understood before this task.

v. Risks and Contingencies

There are no risks and contingencies involved.

2.3.1.3 Analysis and Requirements

i. Description

Various plants diseases will be analyzed, requirements will be gathered from stakeholder, SRS is developed and finalized. Following diagrams will be drawn:

- Use-case Diagram
- Entity-Relationship Diagram
- Domain Model

ii. Deliverables and Milestones

SRS is the deliverable for this task.

iii. Resources Needed

"Microsoft-Word", "Start UML" and Abdur Rehman are the resources required for the completion of this task.

iv. Dependencies and Constraints

Software Management Plan is prerequisite for the completion of this task.

v. Risks and Contingencies

There are no risks and contingencies involved.

2.3.1.4 Design

i. Description

A solution to the problem solved by the software will be designed here. Following diagrams will be drawn:

- Architectural Design Diagram
- Sequence Diagram
- Class Diagram

ii. Deliverables and Milestones

Software Design Description is the deliverable for this task.

iii. Resources Needed

"Microsoft-Word", "Star UML", "Software Requirements Specification" document and Abdur Rehman.

iv. Dependencies and Constraints

Analysis is prerequisite for the completion of this task. During analysis, we cannot jump to design.

v. Risks and Contingencies

There are no risks and contingencies involved for the completion of this task.

2.3.1.5 Implementation

i. **Description**

Software will be implemented.

ii. **Deliverables and Milestones**

Software Implementation is the deliverable for this task.

iii. **Resources Needed**

"Android Studio" and Abdur Rehman are the resources needed for this task.

iv. **Dependencies and Constraints**

Design is prerequisite for the completion of this task.

v. **Risks and Contingencies**

There are no risks and contingencies involved.

2.3.1.6 Testing

i. **Description**

Software will be tested to check whether it satisfies user requirements.

ii. **Deliverables and Milestones**

Documentation of the results obtained by testing is the deliverable for this task.

iii. **Resources Needed**

"Android Device", "Cloud Storage", "Software Requirements Specification" document, Abdur Rehman and user/client are the resources required.

iv. **Dependencies and Constraints**

Software should be implemented before this task. Testing cannot be started before implementation.

v. **Risks and Contingencies**

There are no risks and contingencies involved for the completion of this task.

2.3.2 Assignments

All tasks are assigned to Abdur-Rehman while Supervisor will only provide guidance. Supervisor will only provide high level guidance. Its student's responsibility to explore concepts.

2.3.3 Project Plan

The project plan for the project is developed to keep track of deadlines. This helps in measuring progress of various stages of project in terms of time.

	Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Analysis**	57 days	**9/21/17 8:00 AM**	**12/8/17 5:00 PM**		**Abdur-Rehman;Supervis..**
2	**Identify Requirements**	9 days	**9/21/17 8:00 AM**	**10/3/17 5:00 PM**		
3	Review Case study	2 days	9/21/17 8:00 AM	9/22/17 5:00 PM		
4	Define Problem	2 days	9/25/17 8:00 AM	9/26/17 5:00 PM	3	
5	Gather User Requiremen	5 days	9/27/17 8:00 AM	10/3/17 5:00 PM	4	
6	**Develop SRS**	27 days	**10/4/17 8:00 AM**	**11/9/17 5:00 PM**	**5**	
7	**Define Use-cases**	8 days	**10/4/17 8:00 AM**	**10/13/17 5:00 PM**		
8	Identify Use-cases	2 days	10/4/17 8:00 AM	10/5/17 5:00 PM	5	
9	Write Use-cases descri	3 days	10/6/17 8:00 AM	10/10/17 5:00 PM	8	
10	Draw Use-Case Diagrar	2 days	10/11/17 8:00 AM	10/12/17 5:00 PM	9	
11	Review Use-Case Diagr	1 day	10/13/17 8:00 AM	10/13/17 5:00 PM	10	
12	**Develop Analysis-Mo**	5 days	**10/16/17 8:00 AM**	**10/20/17 5:00 PM**	**11**	
13	Draw Domain-Model	4 days	10/16/17 8:00 AM	10/19/17 5:00 PM		
14	Review Domain Model	1 day	10/20/17 8:00 AM	10/20/17 5:00 PM	13	
15	**Define Requirements**	14 days	**10/23/17 8:00 AM**	**11/9/17 5:00 PM**	**14**	
16	Define Functional Requ	10 days	10/23/17 8:00 AM	11/3/17 5:00 PM		
17	Review Functional Req	1 day	11/6/17 8:00 AM	11/6/17 5:00 PM	16	
18	Define Non-Functional I	2 days	11/7/17 8:00 AM	11/8/17 5:00 PM	17	
19	Review Non-Functional	1 day	11/9/17 8:00 AM	11/9/17 5:00 PM	18	
20	Review SRS	4 days	11/10/17 8:00 AM	11/15/17 5:00 PM	19	
21	Submit SRS and Project pla	0 days	12/4/17 8:00 AM	12/4/17 8:00 AM	20	
22	Finalize SRS	5 days	12/4/17 8:00 AM	12/8/17 5:00 PM	21	
23	Analysis Done	0 days	12/8/17 5:00 PM	12/8/17 5:00 PM	22	

Figure 2.1 Project Plan

	Name	Duration	Start	Finish	Predecessors	Resource Names
24	**Design**	34 days	**12/11/17 8:00 AM**	**1/25/18 5:00 PM**	**23**	**Abdur-Rehman;Super**
25	**Develop Design Diagra**	14 days	**12/11/17 8:00 AM**	**12/28/17 5:00 PM**		
26	Draw Architectral Design	2 days	12/11/17 8:00 AM	12/12/17 5:00 PM		
27	Review Architectral Desig	1 day	12/13/17 8:00 AM	12/13/17 5:00 PM	26	
28	Dvevelop Interface Desig	2 days	12/14/17 8:00 AM	12/15/17 5:00 PM	27	
29	Review Interface Design	1 day	12/18/17 8:00 AM	12/18/17 5:00 PM	28	
30	Draw Sequence Diagram	2 days	12/19/17 8:00 AM	12/20/17 5:00 PM	29	
31	Review Sequence Diagra	1 day	12/21/17 8:00 AM	12/21/17 5:00 PM	30	
32	Draw Class Diagram	4 days	12/22/17 8:00 AM	12/27/17 5:00 PM	31	
33	Review Class Diagram	1 day	12/28/17 8:00 AM	12/28/17 5:00 PM	32	
34	**Decide Algorithms**	11 days	**12/29/17 8:00 AM**	**1/12/18 5:00 PM**	**33**	
35	Identify Machine Learnin	5 days	12/29/17 8:00 AM	1/4/18 5:00 PM		
36	Choose Algorithm	2 days	1/5/18 8:00 AM	1/8/18 5:00 PM	35	
37	Review Algorithm	2 days	1/9/18 8:00 AM	1/10/18 5:00 PM	36	
38	Finalize Algorithm	2 days	1/11/18 8:00 AM	1/12/18 5:00 PM	37	
39	**Evaluate Design**	6 days	**1/15/18 8:00 AM**	**1/22/18 5:00 PM**	**38**	
40	Validate Design	3 days	1/15/18 8:00 AM	1/17/18 5:00 PM		
41	Verify Design	3 days	1/18/18 8:00 AM	1/22/18 5:00 PM	40	
42	Review Design	2 days	1/23/18 8:00 AM	1/24/18 5:00 PM	41	
43	Submit Design Document	0 days	1/24/18 5:00 PM	1/24/18 5:00 PM	42	
44	Finalize Design	1 day	1/25/18 8:00 AM	1/25/18 5:00 PM	43	
45	Design Phase Completed	0 days	1/25/18 5:00 PM	1/25/18 5:00 PM	44	

Figure 2.2 Project Plan

Chapter 3 Software Requirements Specification

3.1 Introduction

This section gives an overview of every item included in the SRS (Software Requirements Specification). The purpose of the SRS (Software Requirements Specification) and the scope of the software is included in this section and a table of definitions is also included.

3.1.1 Purpose

The purpose of this chapter is to give a detailed description of the requirements for the "Leaf Disease Identification System" software. This section will give all the necessary details required for the development of the software.

3.1.2 Scope

The "Leaf Disease Identification System" is an android based mobile-application which helps people in diagnosing the disease of plants. The application takes a photo of the leaf of a plant and diagnose the possible disease on the basis of different color patterns on leaf. This project is only related to two diseases (Stripe-Rust and Leaf-Rust) of wheat plants. This system does not implement leaf-recognition feature, because

The application will use image of leaf as input and display results to the user. The application will store the results including some meta-data of processing in a local database of application and also upload these on to a Cloud Storage, if the user is a registered user of the application for review by researchers.

All information related to diseases diagnosed in various places will be maintained on a Cloud Storage. Processing of the data at the cloud site is out of the scope of this project.

3.1.3 Definitions, acronyms, abbreviations

Table 3.1 Definitions, acronyms, abbreviations

Term	**Definition**
User	Someone who interacts with the mobile application
Biologist	A person related to the branch of science concerning living organisms
GPS	Global Positioning System
Surface area	Area of uppermost or outer part of something
Cloud Storage	Data storage model in which data is divided into logical pools and physical storage spans multiple servers.

3.2 Overall description

This section gives an overview of the whole system.

3.2.1 Product Perspective

This system will consist of two parts: one mobile application and a Cloud Storage. Mobile application will be used to take photo of leaf and will process the photo. Application will show results to user and uploads the results including meta-data to the Cloud Storage.

The mobile application will need to communicate with the camera within the mobile phone to capture the photo, see figure 3.1.GPS device within the mobile phone will also be used to get the current location of the user. Both these functionalities will be embedded in the application, so the user can use application with minimum effort.

Mobile application will also communicate with the Cloud Storage. Cloud Storage could only be accessed by authenticated users.

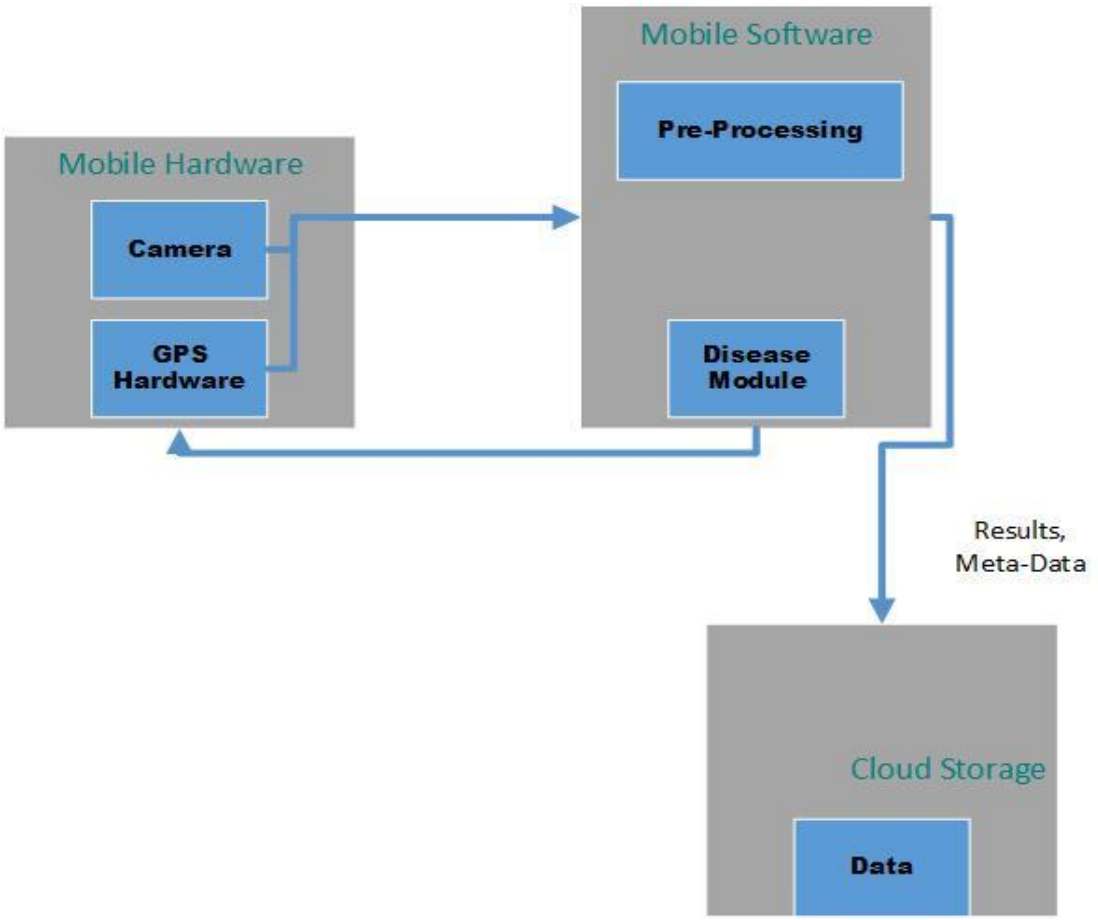


Figure 3.1 Block-Diagram

3.2.2 Product functions

With the mobile application users will be able to take picture of the leaf and see results. It will automatically process picture and save data on to local database of application. For uploading data on Cloud Storage users will have to be registered. Results provided by application will include percentage severity and type of disease. Application will process input image based on some pre-specified criteria.

Cloud Storage will store data provided by mobile application. Only registered users will be able to upload data to cloud storage. This data will be useful for analyzing various diseases across the country. All the data related to certain plant diseases will be available for analysis on one place. But strictly speaking analysis software for data on Cloud storage is out of project scope.

3.2.3 User characteristics

There are two types of users that will interact with the system: normal (unregistered) users of the mobile application, and the registered users who will upload the data on the Cloud Storage. Each of these users has different use of the system, so that they have their own requirements.

Users of the mobile application will only use the application to diagnose their plant's disease. They will not be able to see what others have uploaded on Cloud Storage.

3.2.4 Constraints

The mobile application is constrained by the camera hardware within mobile phone. Since different mobile phones have different camera, the picture taken by application will vary according to camera quality, hence results can vary.

Internet connection is also a constraint for the mobile application. Since Transmission of data between application and Cloud Storage takes place through internet. But internet connection is not crucial for application to be functional. Whenever internet connection will be available, application will upload data otherwise store locally.

Another constraint on mobile application is that picture will always be taken with a white paper in the background. Scaling for calculation of surface area will be mentioned on the paper. Without explicit scaling mentioned on white paper application will not be able to calculate leaf surface area.

3.2.5 Assumptions and dependencies

One assumption about the product is that it will always be used on android based mobile phones with enough camera quality. If the camera quality is low, then the results presented to user may not be accurate or completely inaccurate.

Another assumption about product usage is that user will always place wheat leaf no other plant leaf with a white background. Otherwise, application will not work as intended.

3.3 Specific Requirements

This section contains details of the interfaces and database requirements. This section also describes functional requirements and software system attributes.

3.3.1 External Interface Requirements

The following section gives description of external interfaces to application and their requirements.

3.3.1.1 User Interfaces

A user of the application should be able to run the application on his/her device. When the application will be running, user will be able to take photo of a plant's leaf or select the photo from gallery and press the button process. Then the application will automatically calculate results and display to the user.

3.3.1.2 Hardware Interfaces

Interface to camera and GPS hardware will be handled by the underlying operating system. Our application will request to operating system to use these features. All the low-level details of handling camera and GPS will be hidden from our application.

3.3.1.3 Software Interfaces

The software interface between application and Cloud storage will be handled by application automatically.

3.3.1.4 Communication Interfaces

There are no special communication interfaces requirements. Communication interfaces will be operating system specific.

3.3.2 Software Product Features

Some of the software product features are defined below in the form of Use Cases. Each table contains a single use-case. Fully-dressed format is used for writing use-cases. Following is the list of identified use-cases:

1. Take Photo
2. Select Photo
3. Diagnose Leaf
4. Save Data
5. View Local Data
6. Register User
7. Upload Data
8. Sign in
9. Sign out

1. Take Photo

This is the option to take picture using device camera. User just have to click capture button and picture will be automatically captured and send back to application.

Table 3.2 UC-1 Take Photo

UC-1 Take Photo	
Scope	Leaf Disease Identification System
Level	Subfunction
Primary-Actor	User
Stakeholders	User wants to take photo of leaf to be used as input.
Preconditions	Application is running in device.
Success Guarantee	Photo has been captured.
Main Success Scenario	1. User will place a white paper behind the leaf. 2. User will take photo of leaf using camera option in application. 3. Application will show photo to user for confirmation. 4. User confirms photo as input.
Extensions	3a: User wants to discard photo and take a new one 1. User discards photo. 2. User will take photo of leaf using camera option in application. 3. Application will show photo to user for confirmation.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

2. Select Photo

This is the option to select picture from device gallery. User just have to select the desired picture and it will be automatically sent back to application.

Table 3.3 UC-2 Select Photo

UC-2 Select Photo	
Scope	Leaf Disease Identification System
Level	Subfunction
Primary-Actor	User
Stakeholders	User wants to select photo of leaf from gallery to be used as input.
Preconditions	Application is running in device.
Success Guarantee	Photo has been selected successfully.
Main Success Scenario	1. User will select photo of leaf from gallery. 2. Application will show photo to user for confirmation. 3. User confirms photo as input.
Extensions	3a: User wants to discard photo and take a new one 1. User discards photo. 2. User will select photo from gallery. 3. Application will show photo to user for confirmation.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

3. Diagnose Disease

This is the option that will allow user to diagnose the selected leaf.

Table 3.4 UC-3 Diagnose Disease

UC-3 Diagnose Disease	
Scope	Leaf Disease Identification System
Level	User goal
Primary-Actor	User
Stakeholders	User wants correct diagnose of leaf disease.
Preconditions	Application is running in device.
Success Guarantee	Disease identified in leaf or leaf is healthy.
Main Success Scenario	1. User confirms photo as input. 2. Application will process photo and display results to user.
Extensions	1a. Take photo: Include Take Photo 1b. Select photo: Include Select Photo 2a. Save Data: Include Save Data
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

4. View Local Data

This is the option to allow user to view data (results) saved locally in device storage.

Table 3.5 UC-4 View Local Data

UC-4 View Local Data	
Scope	Leaf Disease Identification System
Level	User goal
Primary-Actor	User
Stakeholders	User wants to see all the previous results stored in device storage.
Preconditions	1. Application is running in device. 2. Application has saved previous results.
Success Guarantee	Data is shown on screen to user.
Main Success Scenario	1. User will select the option to view data. 2. Application will show all the data stored in local database of application on screen in a format that is understandable for user.
Extensions	2a: Application does not show any data. 1. User again selects option to view data. 2. Application will show all the data stored in local database of application on screen in a format that is understandable for user.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

5. Save Data

This is the situation where application saves results of the leaf diagnose to device storage.

Table 3.6 UC-5 Save Data

UC-5 Save Data	
Scope	Leaf Disease Identification System
Level	Subfunction
Primary-Actor	User
Stakeholders	User wants to save all the results in device storage.
Preconditions	1. Application is running in device. 2. User has diagnosed the leaf and application has calculated the results.
Success Guarantee	Data is saved to device storage.
Main Success Scenario	1. Application saves results to device storage. 2. User views results in application.
Extensions	1a. Application fails to save results. 1. Application tries again and results are saved.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

6. Register User

This is the option that allows user to register in order to upload data to cloud.

Table 3.7 UC-6 Register User

UC-6 Register User	
Scope	Leaf Disease Identification System
Level	User goal
Primary-Actor	User
Stakeholders	User wants to register to upload data to cloud storage.
Preconditions	1. Application is running in device.
Success Guarantee	User registered successfully.
Main Success Scenario	1. User will select register option. 2. Application will present form for registration details. 3. User fills form with user credentials and submits form. 4. Application will process form data. 5. Application shows user registered successfully.
Extensions	5a: Application shows that form is not completely filled: 1. User fills form with user credentials and submits form. 2. Application will process form data.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

7. Upload Data

This is the option that allows a signed-in user to upload data (results) to cloud storage.

Table 3.8 UC-7 Upload Data

UC-7 Upload Data	
Scope	Leaf Disease Identification System
Level	User goal
Primary-Actor	Registered User
Stakeholders	User wants to upload data to cloud storage.
Preconditions	1. Application is running in device. 2. User is signed-in 3. User has diagnosed the leaf and application has calculated the results.
Success Guarantee	Data uploaded to cloud storage.
Main Success Scenario	1. User will select upload data to cloud option. 2. Application will communicate with the cloud storage service to upload data to cloud.
Extensions	2a: Application fails to upload data due to network un-availability: 1. Application saves data and tries again, when network is available.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

8. Sign in

This is the option that allows user to sign-in in order to upload data to cloud.

Table 3.9 UC-8 Sign in

UC-8 Sign in	
Scope	Leaf Disease Identification System
Level	User goal
Primary-Actor	Registered User
Stakeholders	User wants to sign in to upload data to cloud storage.
Preconditions	1. Application is running in device.
Success Guarantee	Application shows that user is signed in.
Main Success Scenario	1. User will select Sign in option. 2. User will enter his credentials. 3. Application will communicate with the authentication service to check user credentials. 4. Authentication service confirms user credentials, application shows that user has signed in successfully.
Extensions	4a: Authentication service responds with failure message. 1. Application will ask user to try again. 2. User will enter his credentials. 3. Application will communicate with the authentication service to check user credentials. 4. Authentication service confirms user credentials, application shows that user has signed in successfully.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

9. Sign out

This is the option that allows a user to sign-out from application.

Table 3.10 UC-9 Sign out

UC-9 Sign out	
Scope	Leaf Disease Identification System
Level	User goal
Primary-Actor	Registered User
Stakeholders	Wants to sign out.
Preconditions	1. Application is running in device. 2. User is signed-in.
Success Guarantee	Application shows that user is signed out.
Main Success Scenario	1. User selects sign out option. 2. Application shows user that he/she has signed out successfully.
Extensions	2a: Application shows that user is signed-in. 1. User selects sign out option again. 2. Application shows user that he/she has signed out successfully.
Special Requirements	Touch screen user interface
Frequency of Occurrence	Multiple times

3.3.1 Use-case diagram

Following is the use-case diagram.

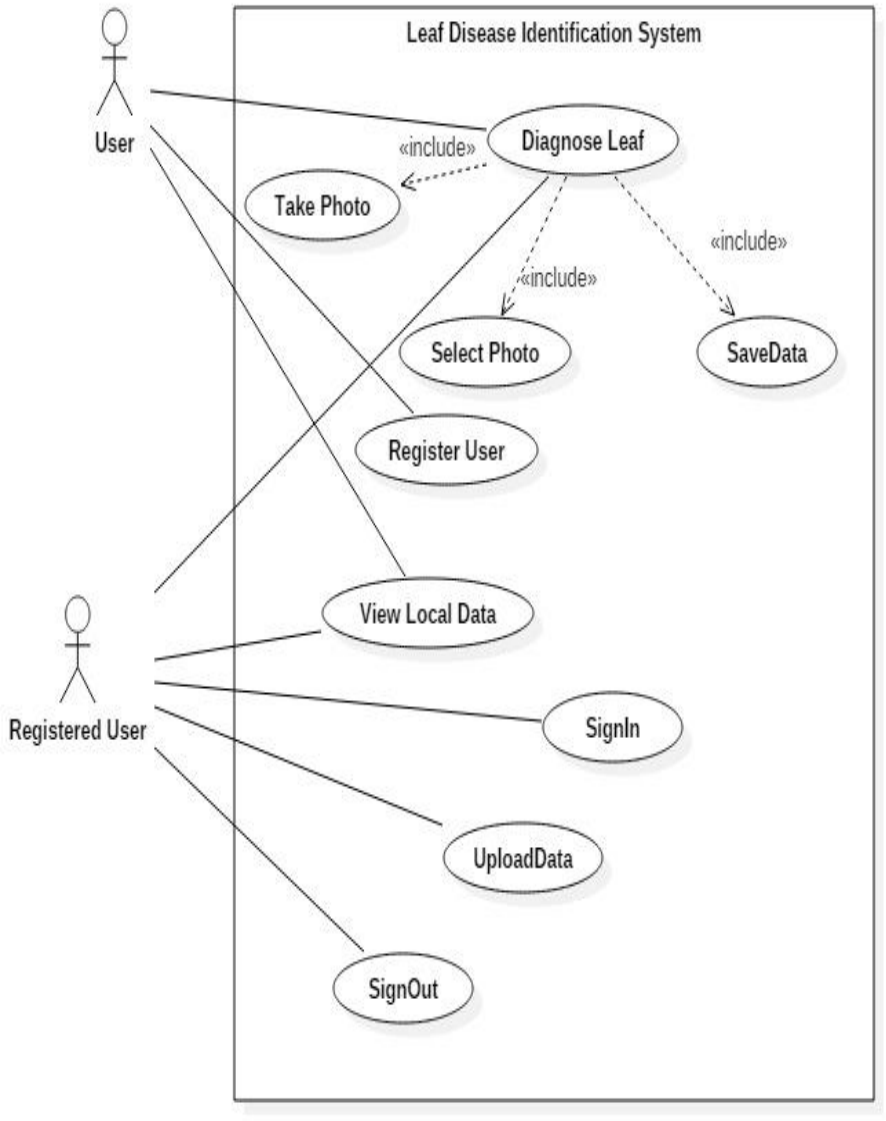


Figure 3.2 Use-Case Diagram

3.3.2 Software System Attributes

This section specifies the required reliability, availability and portability and maintainability.

3.3.2.1 Reliability

The reliability of the system depends on accuracy of diagnose. It should never be able to crash unless or until some system or device failure occurs.

3.3.2.2 Availability

The system is always available for use once installed in device. It will require internet connection for downloading from remote source and must not take much time to launch the application.

3.3.2.3 Portability

Since the system will be designed in Android-Studio, which provides portability to all android operating systems, so this application will support other Android systems as well (v4.4 or higher).

3.3.2.4 Maintainability

The system has the ability to easily adapt to new features and updates. All upgrades can quickly and safely be performed with in minimum time.

3.3.3 Database Requirements

There is a database for the application, that will reside locally in device.

3.3.3.1 ERD

The following figure shows entity relationship diagram for the database.

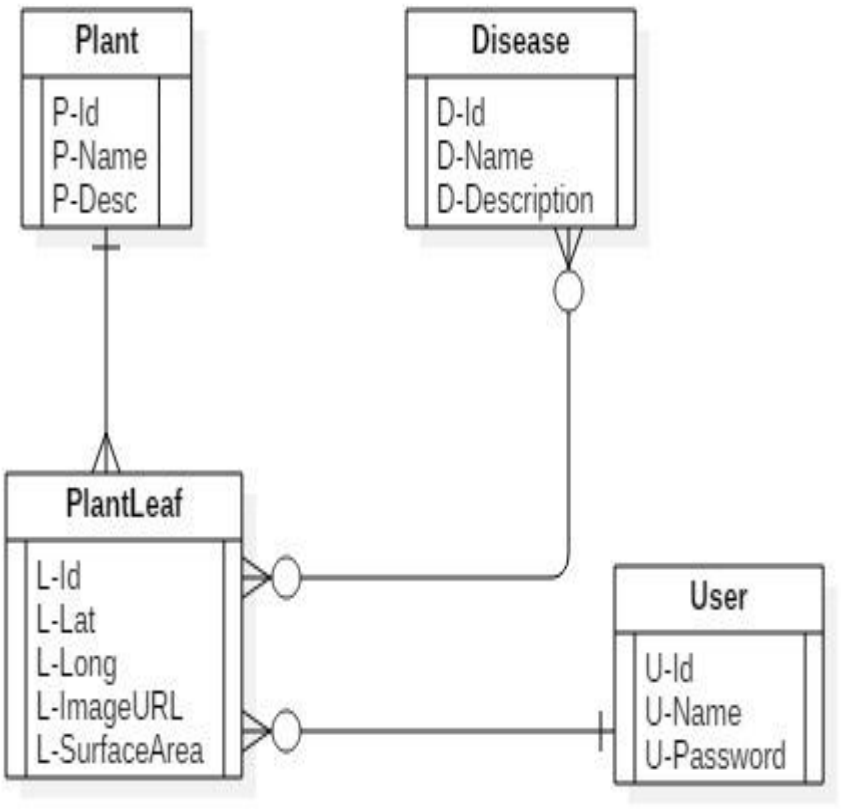


Figure 3.3 ERD

3.3.4 Domain Model

It is a conceptual model of all the topics related to a specific problem. It describes the entities their attributes, roles and relationships of the entities, and the constraints that specify the problem domain.

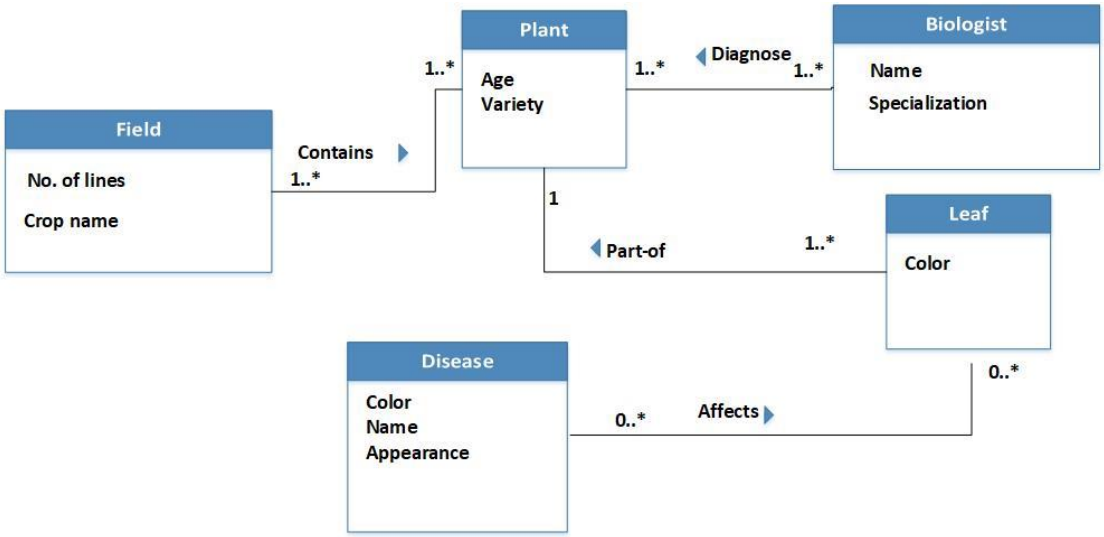


Figure 3.4 Domain-Model

Chapter 4 Software Design Description

4.1 Introduction

The Design Overview section deals with the description of Architecture of the system and product overview.

4.1.1 Overview

The Software Design Description provides the documentation that helps in designing the software. This description will convey the design philosophy and architecture of application. It will also define how the developer intends to implement the application to a function according to the software requirements. This description contains the Architectural design, Sequence diagram and Class diagram.

4.1.2 Purpose

The purpose of the Software Design Description is to provide a detailed description of the design of Leaf Disease Identification System. It will aid in constructing a system that would be efficient. It will also provide necessary information for the designing of software and system to be built.

4.2 Requirements Traceability Matrix

Requirement Traceability Matrix or RTM captures all requirements proposed by the client or development team and their traceability in a single document. In other words, it maps and traces user requirement with test cases. The main purpose of Requirement Traceability Matrix is to see that all test cases are covered so that no functionality should remain missing during the testing.

Table 4.1 Requirements Traceability Matrix

Requirements / Use case	Domain Model	Sequence Diagram	Class Diagram	Test Case
UC-1	Not Required	Figure 4.2	Figure 4.7	Test Case 1
UC-2	Not Required	Figure 4.2	Figure 4.7	Test Case 2
UC-3	Figure 2.3	Figure 4.2	Figure 4.7	Test Case 3
UC-4	Not Required	Figure 4.3	Figure 4.7	Test Case 4
UC-5	Not Required	Figure 4.2	Figure 4.7	Test Case 5
UC-6	Not Required	Figure 4.4	Figure 4.7	Test Case 6
UC-7	Not Required	Figure 4.2	Figure 4.7	Test Case 9
UC-8	Not Required	Figure 4.5	Figure 4.7	Test Case 7
UC-9	Not Required	Figure 4.6	Figure 4.7	Test Case 8

4.3 System Architectural Design:

System Architecture Design is used to represent the components of a system and the interaction between them. Interaction between components of our system is shown in the form of a diagram.

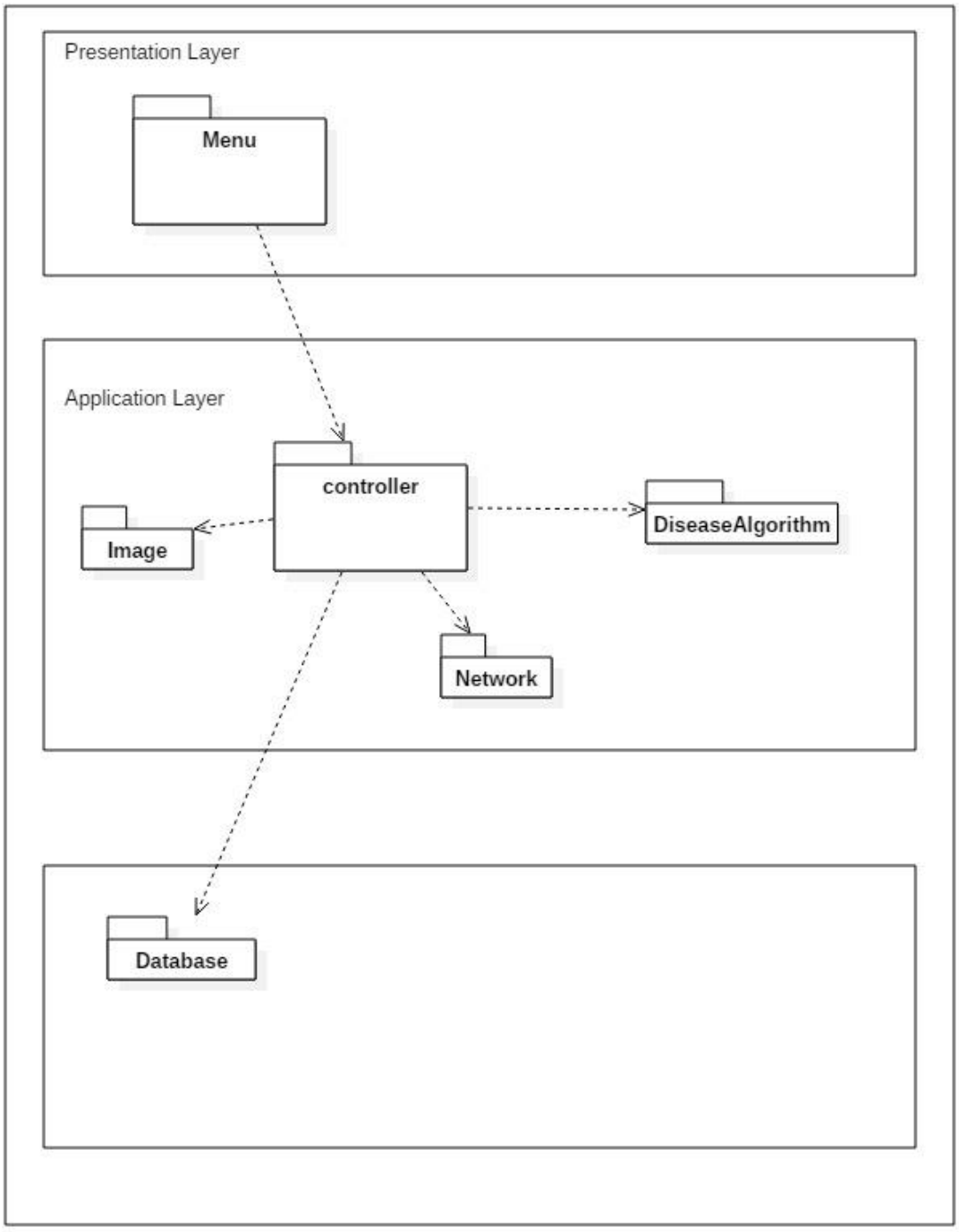


Figure 4.1 Architectural-Design

4.3.1 Chosen System Architecture

Chosen system architecture is 3-tier. There are following three layers;

- Presentation Layer
- Application Layer
- Persistence Layer

4.4 Detailed Description of Components

4.4.1 Menu

Menu is the user interface for displaying set of options at a point in application.

4.4.2 Controller

Controller is the component that is responsible for handling events or actions generated by the presentation layer i.e. user clicks. It will then call the appropriate component in application layer based on the information it received.

4.4.3 Network

Network is the component responsible for handling communication with the cloud side where data of the application is stored. It is responsible for sending data to cloud.

4.4.4 DiseaseAlgorithm

DiseaseAlgorithm is the algorithmic component that will actually process the image to give output to the user.

4.4.5 Image

Image is the component that will be responsible for handling all the operations related to image processing.

4.4.6 Database

Database is a component of persistence layer that is responsible for handling low level communication with the database.

4.5 Sequence Diagram

Sequence diagram depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the system. We identify how the functions are carried out in the system.

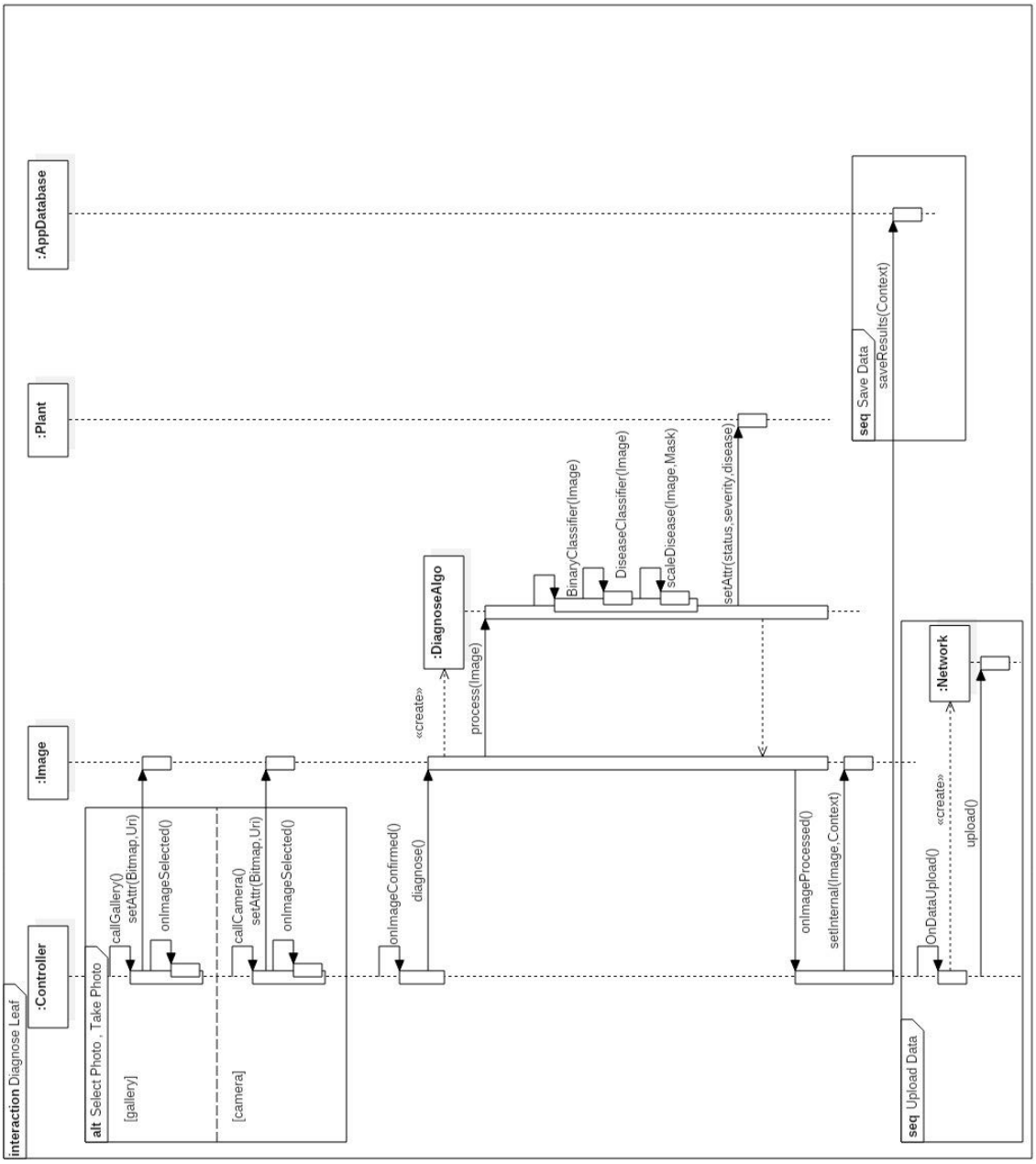


Figure 4.2 Sequence Diagram

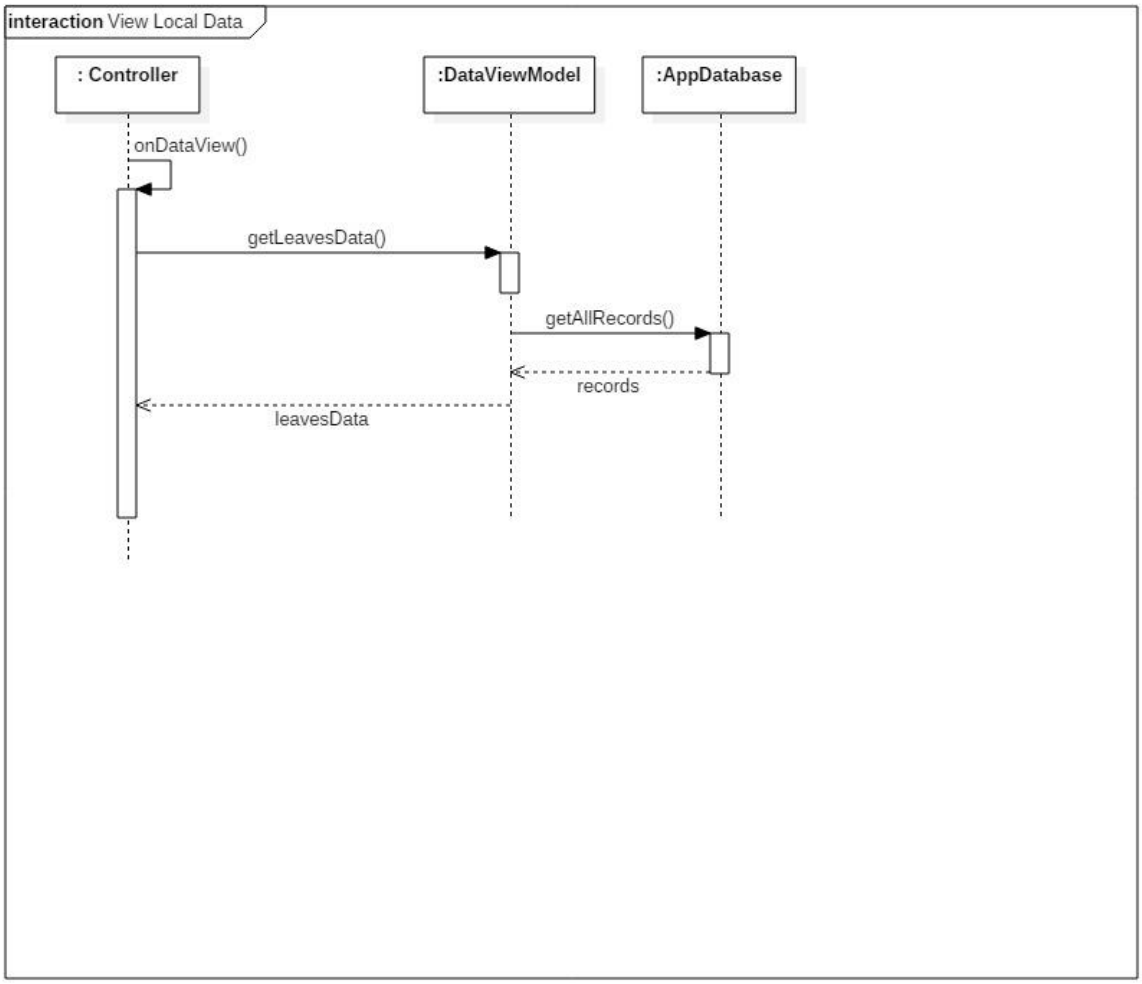


Figure 4.3 Sequence Diagram

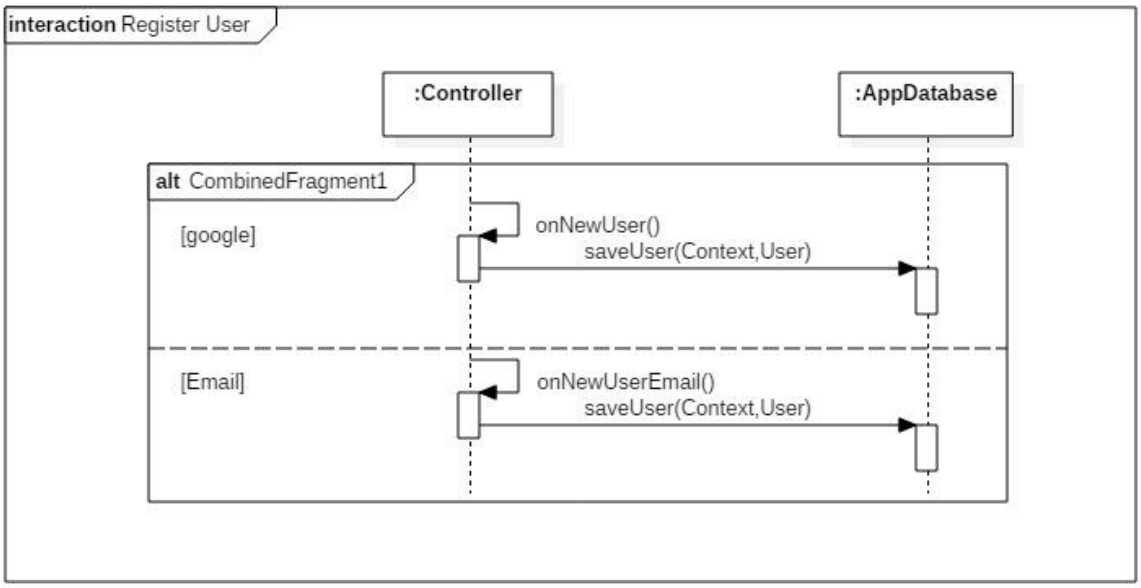


Figure 4.4 Sequence Diagram

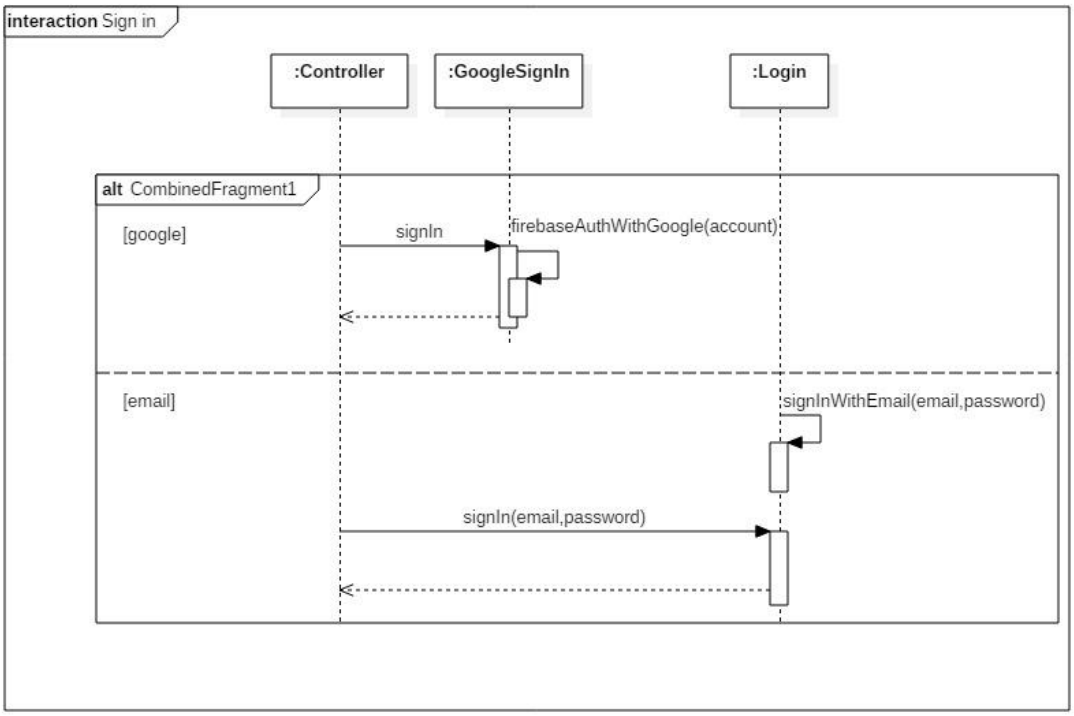


Figure 4.5 Sequence Diagram

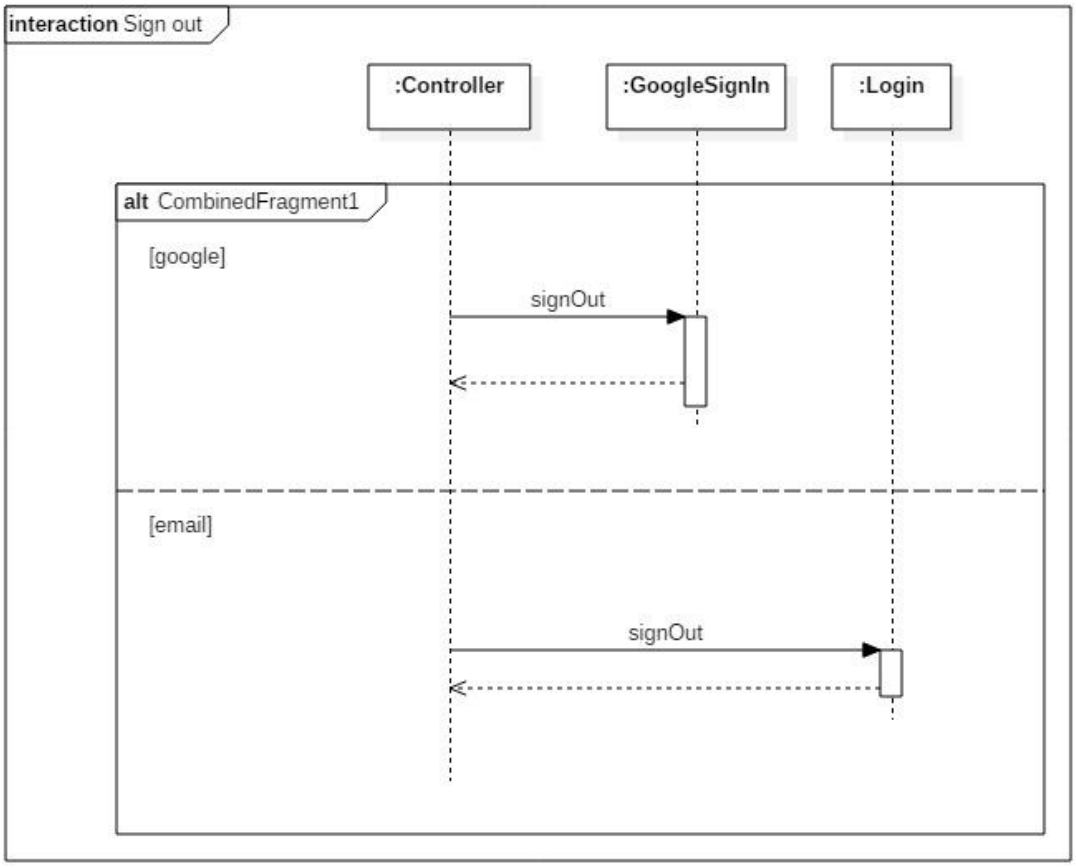


Figure 4.6 Sequence Diagram

4.6 Class Diagram

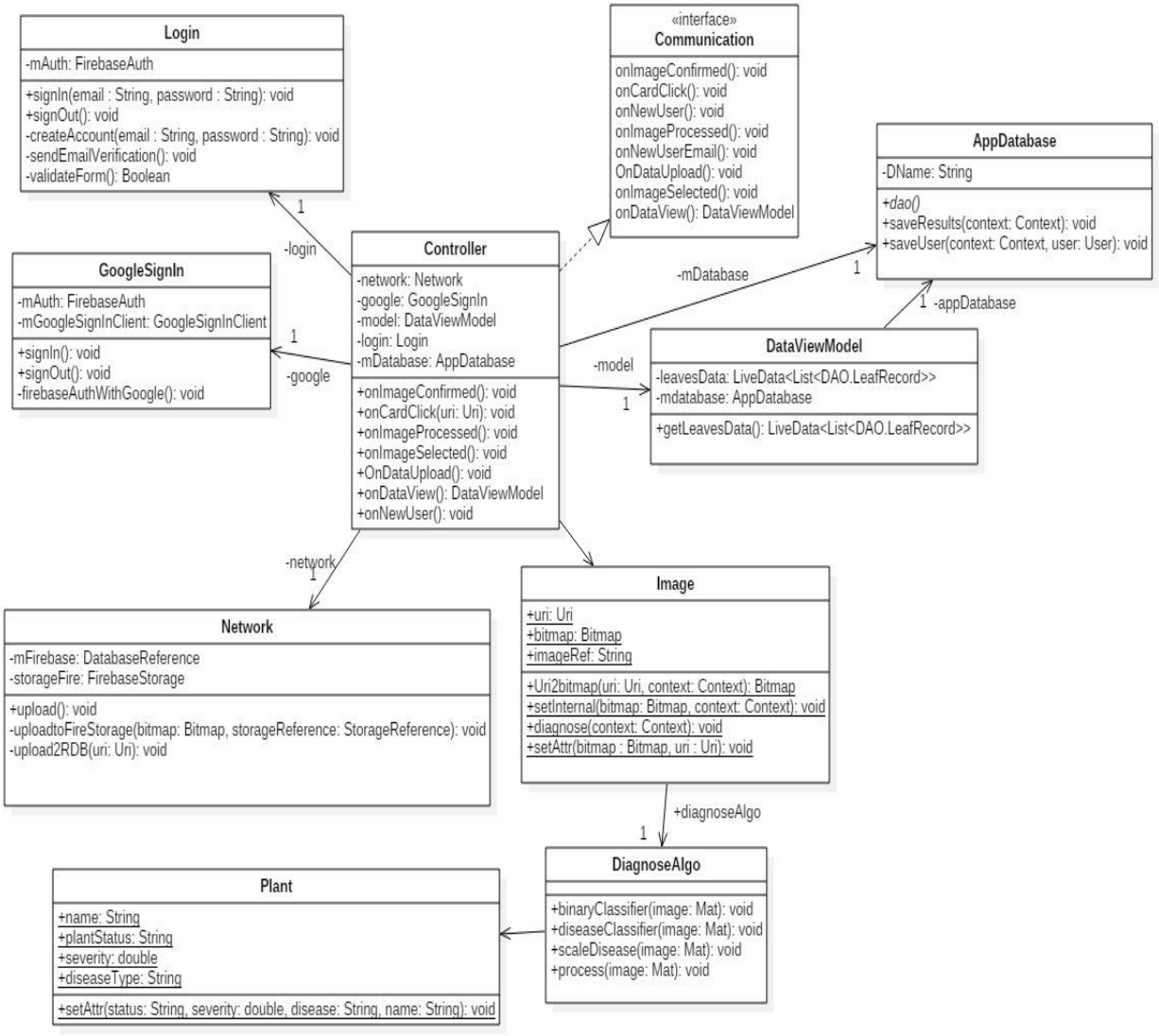


Figure 4.7 Class-Diagram

4.7 User Interface Design

User interface provides interface between the software product and its users. In this section user interface of the application is discussed.

4.7.1 Description of the User Interface

In application, user can interact with application by using touch screen interface of Android device. When user first starts the application, a navigation menu will appear that will allow user to go through various screens in application. Select photo screen will allow user to select or take photo and process it. Results screen shows user the results and allow to upload on cloud if user is signed in. Storage screen shows user the data in local database of application. Google Sign in and Email Password screens will allow user to register, sign in and sign out.

4.7.2 Screen Images

Following are few screen images of application.

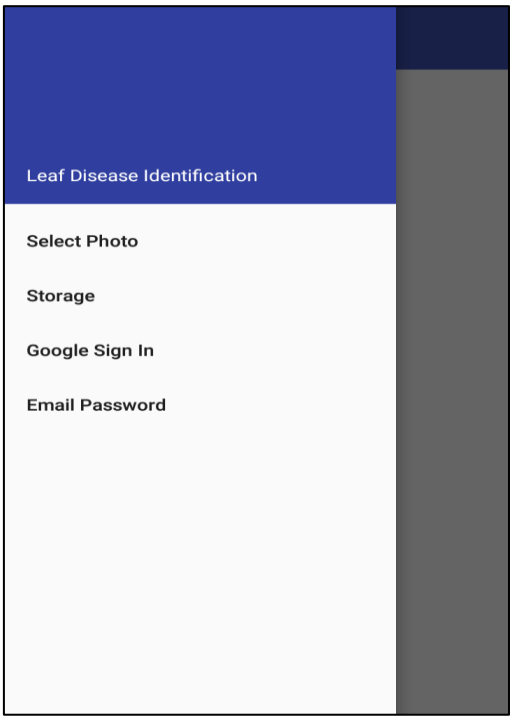


Figure 4.8 Navigation-Screen

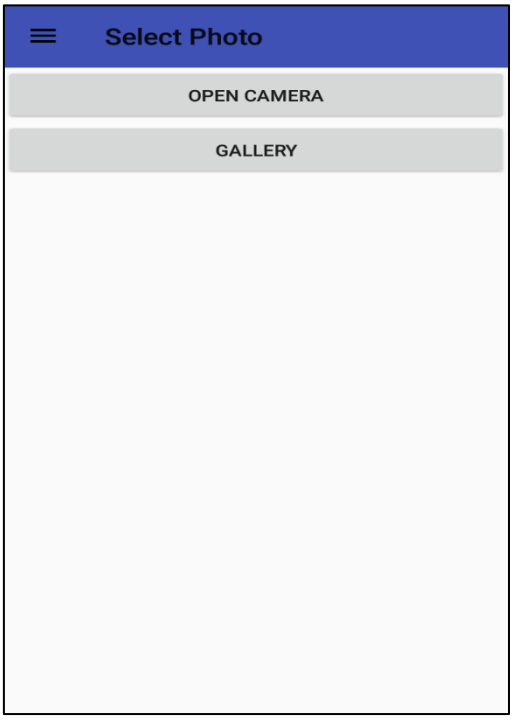


Figure 4.9 Select-Photo-Screen

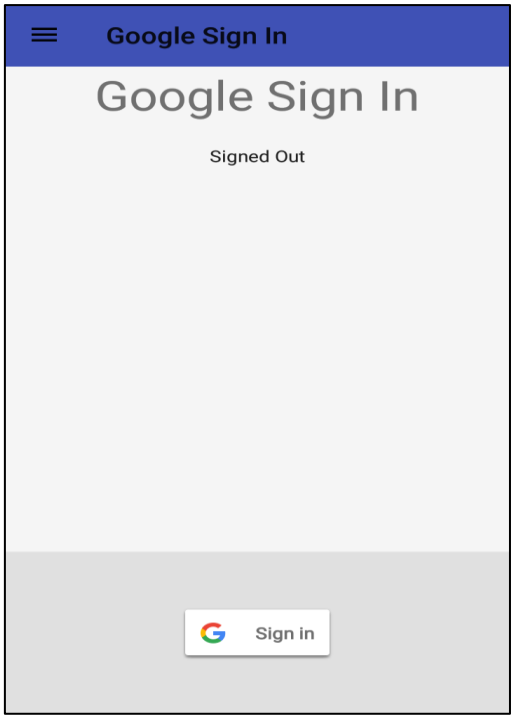


Figure 4.10 Google-Sign-in-Screen

Figure 4.11 Email-Screen

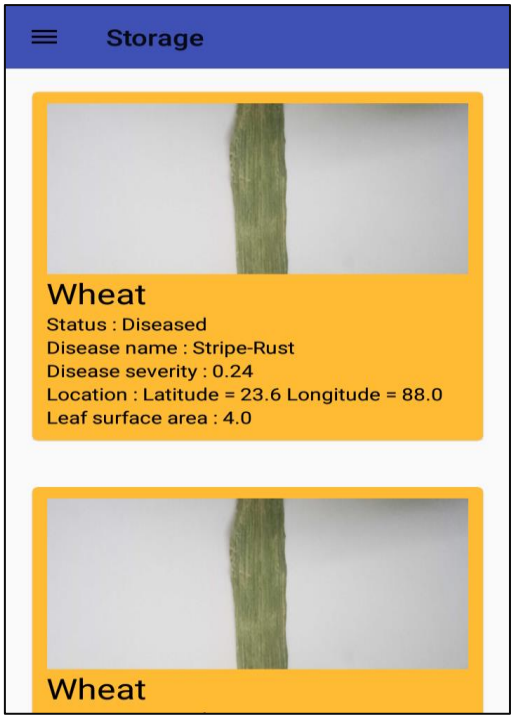


Figure 4.12 Storage-Screen

Figure 4.13 Results-Screen

Chapter 5 Software Implementation

5.1 Introduction

In this chapter, the framework and language selection for the project is provided. It also includes screenshots for the implementation of the application.

5.1.1 Framework Selection

The framework selected for this project is Android Development Kit. Android development kit support development of applications that run on variety of Android devices. Android development is free to use kit includes an IDE i.e. Android Studio. It supports development and testing of application. Android studio uses Gradle as its build to convert source and resource files into Android application package.

For image processing I used OpenCV (Open source Computer Vision) package, it includes many well-known image processing algorithms already implemented and has many other basic image manipulation functions such as for color conversion, rotation, smoothing etc.

5.1.2 OpenCV Functions

1. **cvtColor** (src, dst, code)
   src = input image
   dst = output image
   code = specific code for conversion e.g. RGB2GRAY convert input in RGB to GRAY output.
   This function converts an input image from one color space to another.
2. **GaussianBlur (src, dst, ksize, sigmaX, sigmaY)**
   src = input image
   dst = output image
   ksize = Gaussian kernel size should be positive and odd.
   sigmaX = Gaussian kernel standard deviation in X direction.
   sigmaY = Gaussian kernel standard deviation in Y direction, if sigmaY is zero, it is set to be equal to sigmaX.
   This function convolves the source image with the specified Gaussian kernel.
3. **Canny (src, dst, minVal, maxVal)**
   src = input image
   dst = output image
   minVal = minimum value for gradient intensity
   maxVal = maximum value for gradient intensity
   This function detects edges using Canny edge detector criteria.
4. **HoughCircles (src, dst, code, dp, minDst, param1, param2, minRadius, maxRadius)**
   src = input image
   dst = Output vector of found circles. Each vector is encoded as a 3-element floating-point vector (x, y, radius).
   code = method for detection

dp = inverse ratio of the accumulator resolution to the image resolution.
minDst = minimum distance between the centers of the detected circles.
param1 = first method-specific parameter
param2 = second method-specific parameter
minRadius = Minimum circle radius.
maxRadius = Maximum circle radius.
This function detects circles in image.

5. **inRange (src, lower_bound, upper_yellow, output)**
   src = input image in HSV
   lower_bound = lower limit for HSV color
   upper_bound = upper limit for HSV color
   output = binary matrix
   This function takes input image and lower and upper bounds of color to find in image and returns a matrix as output. If the matrix has ones it means pixels of desired color exist in image otherwise desired color do not exist.

5.1.3 Language Selection

The implementation language selected for this project is Java, because it is the default language for android development and it is widely used in android development. Java is an object-oriented language and automatically handles garbage collection.

5.1.4 Operating System

Application will run on Android Operating Systems of version KitKat v4.4.2 or higher.

5.1.5 Algorithms

There are 3 basic algorithms for this application.

5.1.1.1 Binary Classifier

This algorithm will classify the input image of leaf as diseased or healthy. The following are the steps of algorithms in order:

1. Take image as input.
2. Apply Gaussian blur on image to remove noise.
3. Convert to HSV color space.
4. Find diseased color pixels on the basis of minimum and maximum HSV value of color. OpenCV inRange function is used here.
5. If the inRange function found pixels with diseased color then it will return a binary-matrix with most entries having value of 1 showing disease pixels, otherwise a binary-matrix with all entries having value of 0.
   The following figures show working of steps in order.

Figure 5.1 Step-1

Figure 5.2 Step-2

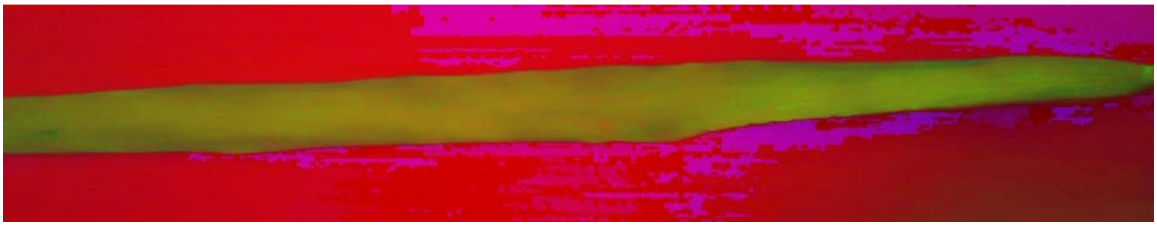


Figure 5.3 Step-3

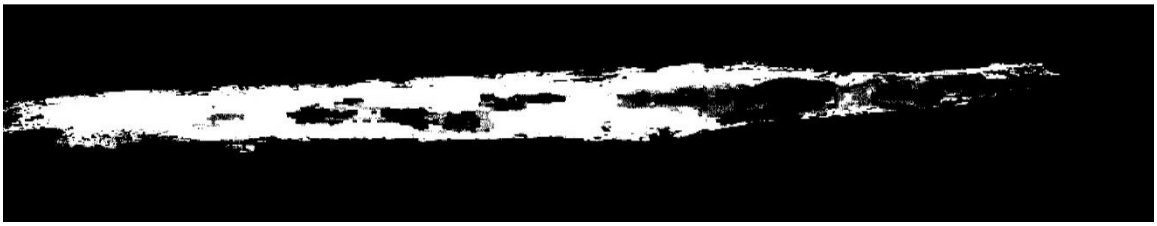


Figure 5.4 Step-4,5

5.1.1.2 Disease Classifier

This algorithm will classify the input image of leaf as a specific type of disease. The following are the steps of algorithms in order:

1. Take image as input.
2. Convert image to gray-scale.
3. Apply Gaussian blur on image to remove noise.
4. Detect whether image of leaf has yellow lines or dark circles. Here Hough Circles function of OpenCV is used to detect circles.
5. If image has yellow lines “Stripe rust” is returned as output else if image has dark red circles “Leaf rust” is returned as output.
   The following figures show the steps of algorithm in order:

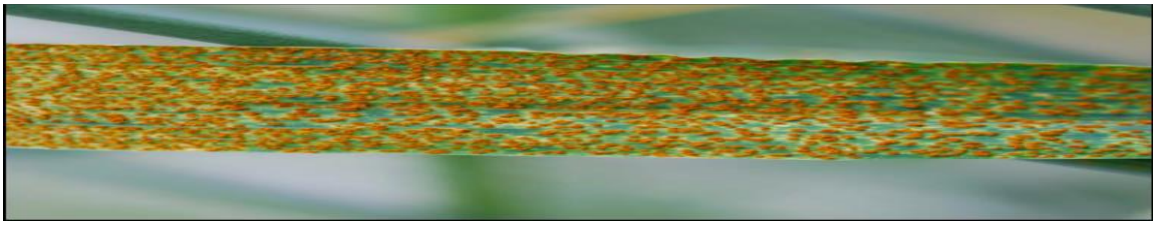


Figure 5.5 Step-1

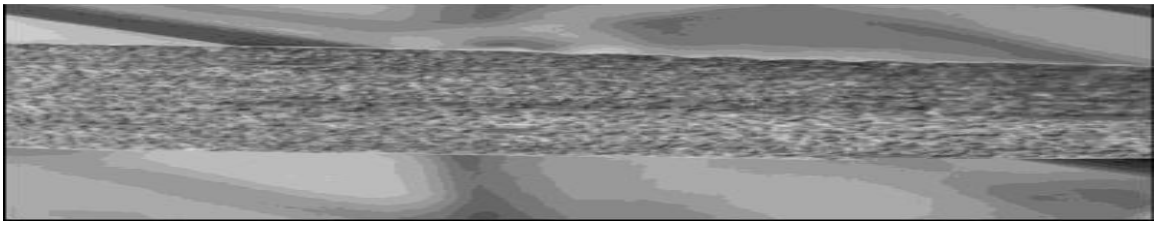


Figure 5.6 Step-2

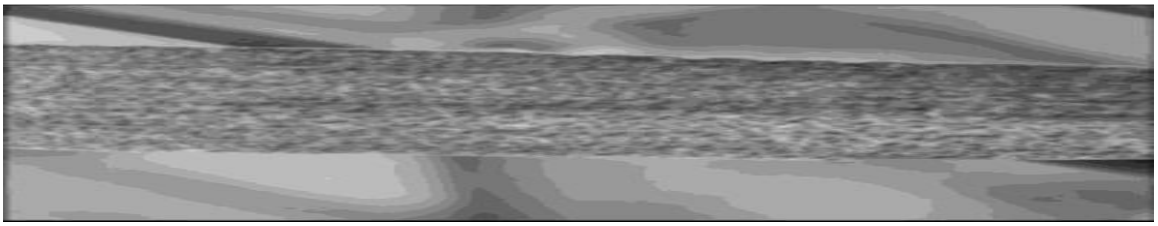


Figure 5.7 Step-3

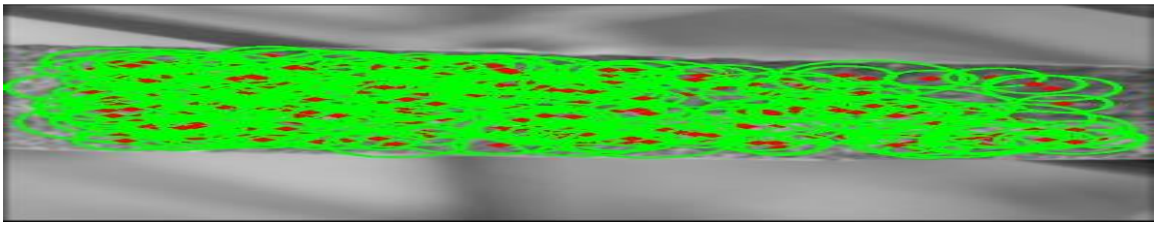


Figure 5.8 Step-4,5

5.1.1.3 Severity Calculator

This algorithm will calculate severity of disease. The following are the steps of algorithms in order:

1. Take image as input.
2. Convert image to gray-scale.
3. Detect edges of leaf using Canny edge detector in OpenCV.
4. Determine bounding box of leaf using OpenCV functions.
5. Calculate approximate total pixels of leaf image. Here image will be gray scale.
6. Calculate approximate total pixels having disease color. Here image will be binary image.
7. (No. of disease pixels)/ (No. of total pixels) ×100 formula is used to calculate severity.
8. Severity in floating point is returned as output.

The following figures show the steps of algorithm in order:

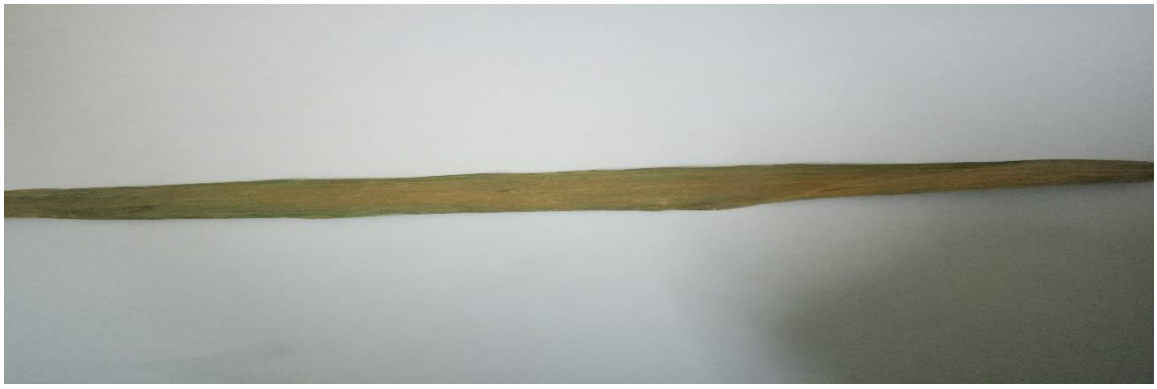


Figure 5.9 Step-1

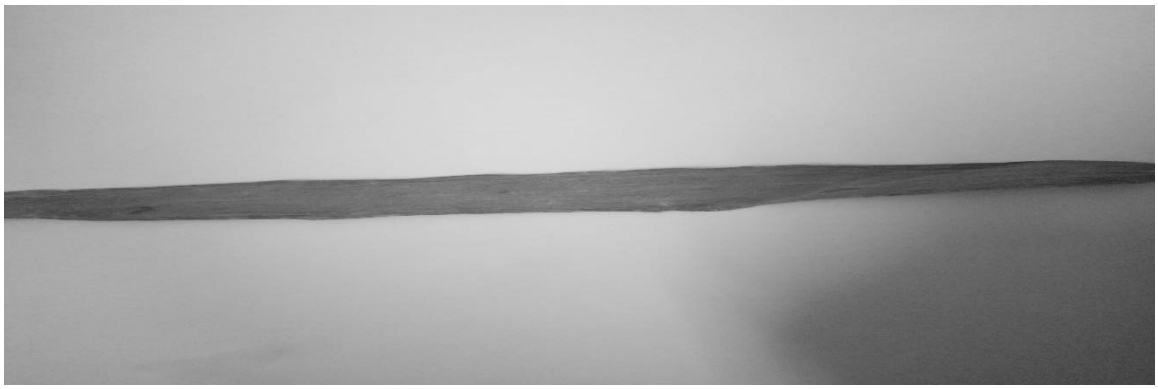


Figure 5.10 Step-2

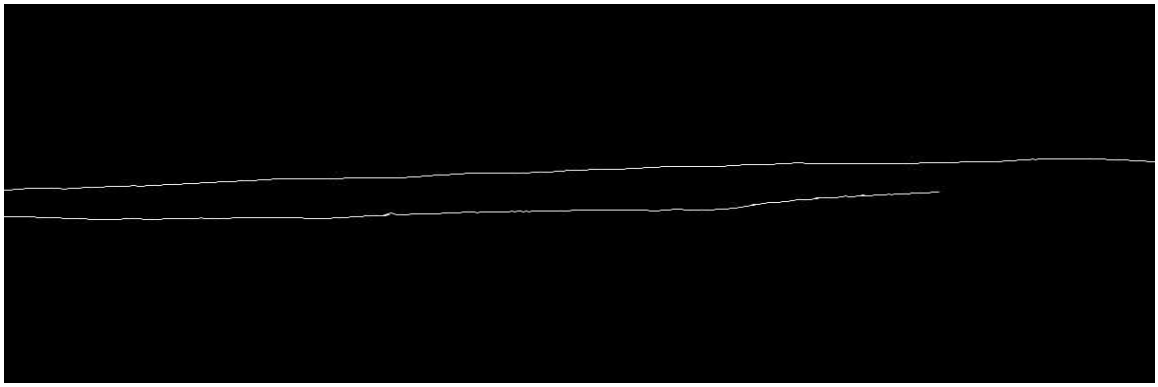


Figure 5.11 Step-3

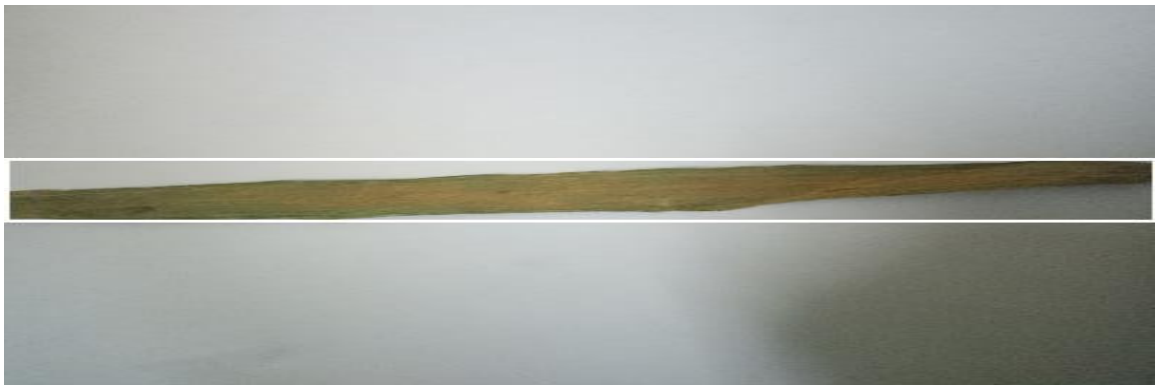


Figure 5.12 Step-4

Chapter 6 Software Testing and Conclusion

6.1 Introduction

Software Testing is the process of evaluating a system or its components with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

6.1.1 System Overview

The product defines an Android based "Leaf Disease Identification System". The purpose of this software is to create a system that will able users to diagnose their plants and also it will help to gather data of various crop diseases in the country, so resistant varieties of crops could be developed.

6.1.2 Test Approach

Testing approach used would be User acceptance testing (UAT) also called end user testing. It consists of a process of verifying that a solution works for the user. Users of the system perform test in line with what occur in real life scenarios.

6.1.3 Testing Objectives

For checking whether the requirement in SRS (Software Requirements Specification) are fulfilled or not we have to make test on these cases.

UAT has following objectives.

i. User Acceptance test make test case against the requirements.
ii. User Acceptance test check actual function, input, expected result, actual result, procedure to make test case, pass/fail status against each test case.

6.2 Test Plan

6.2.1 Features to be tested

Features to be tested are all according to user prospective. For example

- Take picture from camera
- Select picture from gallery
- Save Data
- View Data
- Diagnose leaf
- Upload Data
- Sign in and Sign out
- Register User

6.2.2 Testing Tools and Environment

Since this is UAT testing (testing by the user) so there is no specific tools and environment is required. All a user need is an Android device with application installed.

6.3 Test Cases

6.3.1 Test case for Take Photo

Table 6.1 Test-Case1

Test Case 1: Take Photo	
Purpose	Take Photo
Setup	1. Open application 2. Select camera option
Instructions	1. Take picture with camera.
Expected Result	Photo is captured.
Observed Result	Photo captured successfully.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.2 Test case for Select Photo

Table 6.2 Test-Case2

Test Case 2: Select Photo	
Purpose	Select Photo
Setup	1. Open application and select gallery option
Instructions	1. Choose picture from gallery.
Expected Result	Photo is selected.
Observed Result	Photo selected successfully.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.3 Test case for Diagnose Disease

Table 6.3 Test-Case3

Test Case 3: Diagnose Disease	
Purpose	Diagnose Disease
Setup	1. Open application 2. Take/Select photo for diagnose.
Instructions	1. Select diagnose option.
Expected Result	Application shows disease type and severity.
Observed Result	Application showed disease type and severity.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.4 Test case for View Local Data

Table 6.4 Test-Case4

Test Case 4: View Local Data	
Purpose	View data stored locally in device.
Setup	1. Open application.
Instructions	1. Select view Data option.
Expected Result	Application shows data in a proper format.
Observed Result	Application showed data in local database.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.5 Test case for Save Data

Table 6.5 Test-Case5

Test Case 5: Save Data	
Purpose	Store data locally in device.
Setup	1. Open application.
Instructions	1. Select/Take photo of leaf. 2. Select Diagnose leaf.
Expected Result	Application saves data of leaf recently diagnosed.
Observed Result	Application saved disease type and severity of leaf recently diagnosed.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.6 Test case for Register User

Table 6.6 Test-Case6

Test Case 6: Register User	
Purpose	Register user
Setup	1. Open application. 2. Select register option. 3. Application will present registration form.
Instructions	1. Enters credentials. 2. Submit form.
Expected Result	Application shows user registered successfully.
Observed Result	Application showed user registered successfully.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.7 Test case for Sign in

Table 6.7 Test-Case7

Test Case 7: Sign in	
Purpose	Sign in user
Setup	1. Open application. 2. Select sign in option. 3. Application shows sign in form.
Instructions	1. Enter user credentials. 2. Submit form
Expected Result	Application shows user sign in successfully.
Observed Result	Application showed user signed in successfully.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.8 Test case for Sign out

Table 6.8 Test-Case8

Test Case 8: Sign out	
Purpose	Sign out user
Setup	1. Open application.
Instructions	1. Select sign out option.
Expected Result	Application shows user sign out successfully.
Observed Result	Application showed user signed out successfully.
Frequency	Pass:2 Fail:0
Verdict	Passed

6.3.9 Test case for Upload Data

Table 6.9 Test-Case9

Test Case 9: Upload Data	
Purpose	Upload data to cloud
Setup	1. Open application. 2. Sign in to application.
Instructions	1. Select/Take photo of leaf. 2. Diagnose leaf. 3. Select upload data option.
Expected Result	Data uploaded on cloud.
Observed Result	Application showed data uploaded on cloud successfully.
Frequency	Pass:2 Fail:1
Verdict	Passed

6.4 Conclusion

This product defines an Android based leaf disease identification system for wheat crop known as leaf rust and stripe rust. The main functionality of application is to diagnose whether leaf is heathy or diseased and if diseased its approximate severity and disease type will also be returned. The results of application are satisfactory to some extent which can be refined by having more data available for making better algorithm for disease diagnose.

6.5 Applications

Applications are following:

1. Disease detection
2. Compute leaf size
3. Data collection for research

6.6 Future Enhancements

Application Enhancement and Future work can be done by:

1. Making app diagnose stem rust, which requires taking pictures from different angles and combine them to diagnose disease.
2. Developing algorithms for more plant diseases and integrate them in application.
3. Machine learning algorithms can be applied and the results can be compared.

References

- C. Larman, APPLYING UML AND PATTERNS an Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd ed., Massachusetts: Pearson Education, 2005
- Roger S. Pressman, Software Engineering - A Practitioner's Approach, McGraw Hill, 7th Edition, 2010
- IEEE Software Engineering Standards Committee," IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specification", October 20 1998.
- Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, 2nd ed., Prentice-Hall, Inc,2002
- Gary Bradski, Adrian Kaehler, Learning OpenCV, 1st ed., O'Reilly Media, Inc., 2008
- Mubarak Shah, Fundamentals of Computer Vision, 1st ed., University of Central Florida,1997
- https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html (Jan 05,2019-10:51 PM)
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html (Jan 05,2019-10:51 PM)
- https://cropwatch.unl.edu/plantdisease/wheat/leaf-rust (Jan 05,2019-10:51 PM)
- http://agriculture.vic.gov.au/agriculture/pests-diseases-and-weeds/plant-diseases/grains-pulses-and-cereals/stripe-rust-of-wheat (Jan 05,2019-10:51 PM)