# INTRODUCTION

* 1. **Background**

The Reason I Have Selected This Topic is because it allows it to be a fully functional android application that will help detect and classify plant diseases. It is also a helpful medium for farmers to take appropriate measures of treatment. Early detection of plant diseases is necessary for effective control. The occurrence of the disease on the plant may result in significant loss in both quality as well as the quantity of agricultural product.

This can produce the negative impact on the countries whose economies are primarily dependent on the agriculture. Hence the detection of the disease in the earlier stages is very important to avoid the loss in terms of quality, quantity and finance.

# Objectives

The main objective of the project is to create an Android Application using TensorFlow Lite provide the best possible solution for that disease. It helps in smart farming to increase crop yield and even reduce the effort of farmer in terms of time and money.

# Purpose, Scope and Applicability

* + 1. **Purpose**

The main goal of this software is to identify the disease present in plant leaves with best possible accuracy. This software will also give the solution to the disease which is present in the plant leaves. It will reduce the effort of farmers and will increase the productivity of the farmers.

# Scope

Farmers are struggling throughout the world to protect their crop from the attack of several harmful microorganisms such as virus, fungus, bacteria, nematodes etc. It is only after the disease affects at a great severity level that symptoms occur and demand immediate control. This will help them to timely predict the probable disease in crops and will also eliminate the dependency on manual tasks.

# Applicability

This project can be used by farmers and other people to take care of their crops with the help of this with the help of new technologies like transfer learning and

image processing. It will detect if any disease is spreading in the crops along with this it will also tell the farmers that what treatment should be given to protect it.

# Organization of Report

# The following chapters of this project consists the technologies used, Requirement analysis, Designing, Implementation, and Testing phase. The first phase was to decide what technologies are required to build the project. So a survey of multiple technologies was conducted to find out the best suitable for the application which includes both client and server side. I have chosen to build my in project python and java. The next phase is requirement analysis which defines the requirement of hardware and software in the project. Next is designing phase where the modules will be built and designing would be done. After this is the implementation phase which includes physical designing and programming of all the functionalities. Final is the testing phase that means testing all functionalities and implementation of the project. At the end the project is finally completed and then checked to see whether the objective is successfully achieved.

# Survey of Technologies

# Java: I am making my application in java using Android Studio.

# XML: Used in android studio.

# Python: It is used to build the machine learning model using TensorFlow.

# PyCharm: I am using this to train my model and it is easy and fastest to use.

# Real time Database (Firebase):I am using this database to store my details of user and for authentication.

# Google Colab: For training and testing dataset.

**3. REQUIREMENTS AND ANALYSIS**

**3.1 Problem Definition**

**3.1.1 Existing System**

Earlier classification and detection of diseases was done by 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. It is necessary to make available relevant information and services to the farming community and private sector through the use of information and communication technologies.

## 3.1.2 Proposed System

We’ll create the system based upon image processing and machine learning with the help of tensor flow library. TensorFlow is a multipurpose machine learning framework. TensorFlow can be used anywhere from training huge models across clusters in the cloud, to running models locally on an embedded system like your phone. 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. The image uploaded by the farmer is processed. Then image-processing techniques are applied to the acquired images to extract useful features that are necessary for further analysis. The disease type is detected and displayed on the screen. Even the treatment in order to protect from further damage is provided.

## Requirements Specification

#### Functional Requirements

* The app should be able to let users login or create a new account.
* It should store information of users in a database. The plant images should be stored either on phone database or cloud database.
* The users should be able to easily use the application without much prior knowledge of technology.
* The user scans the picture or uploads from media which automatically recognizes the disease and shows the output.
* The treatment will be displayed on a button click on the detection page.

#### 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. Following requirements are –

**Reliability -** The software would catch the exceptions during

execution.

**Scalability -** The software would be such that new modules and

functionalities can be added.

**Cost -** Cost of the software would be low.

* 1. **Planning and Scheduling**
     1. **Pert Chart**

|  |  |
| --- | --- |
| **Project fixing** | |
| Start Day : 01 | ID : 01 |
| Finish Day : 15 | Duration : 15 |

|  |  |
| --- | --- |
| **Synopsis** | |
| Start Day :16 | ID: 2 |
| Finish Day :31 | DURATION: 15 days |

|  |  |
| --- | --- |
| **Requirement, Gathering and Feasibility** | |
| Start Day : 32 | ID: 3 |
| Finish Day : 47 | DURATION: 15 days |

|  |  |
| --- | --- |
| **System Analysis and Research** | |
| Start Day:48 | ID : 4 |
| Finish Day :228 | Duration :180 days |

|  |  |
| --- | --- |
| **System Design** | |
| START DATE: 229 | ID: 5 |
| END DATE: 239 | DURATION: 10 days |

|  |  |
| --- | --- |
| **Coding** | |
| START DATE: 240 | ID: 6 |
| END DATE: 275 | DURATION: 35 days |

|  |  |
| --- | --- |
| **Testing** | |
| START DATE: 276 | ID: 7 |
| END DATE: 296 | DURATION: 20 days |

|  |  |
| --- | --- |
| **Deployment** | |
| START DATE: 297 | ID: 8 |
| END DATE: 310 | DURATION: 13days |

* 1. **Software and hardware requirements**

**Software Requirements-**

* **Language:** Python, Java, SQL , XML
* **IDE:** PyCharm, Android Studio
* **Backend**: Real Time Database (Firebase)
* Machine Learning
* TensorFlow Lite

**For Training and Testing –** Google Colab

**Hardware Requirements-**

* 8GB RAM Laptop
* Windows 10
* Android Phone
  1. **Preliminary Product Description**

This Plant Disease Classification Application is used to detect disease through image scanning and pattern recognition. Every user who has an android Phone can use it. The app can help you in classification of diseases occurring in plant leaves, stems, lesion etc. The sooner disease appears should be detected, identified and corresponding measures should be taken to avoid loss.

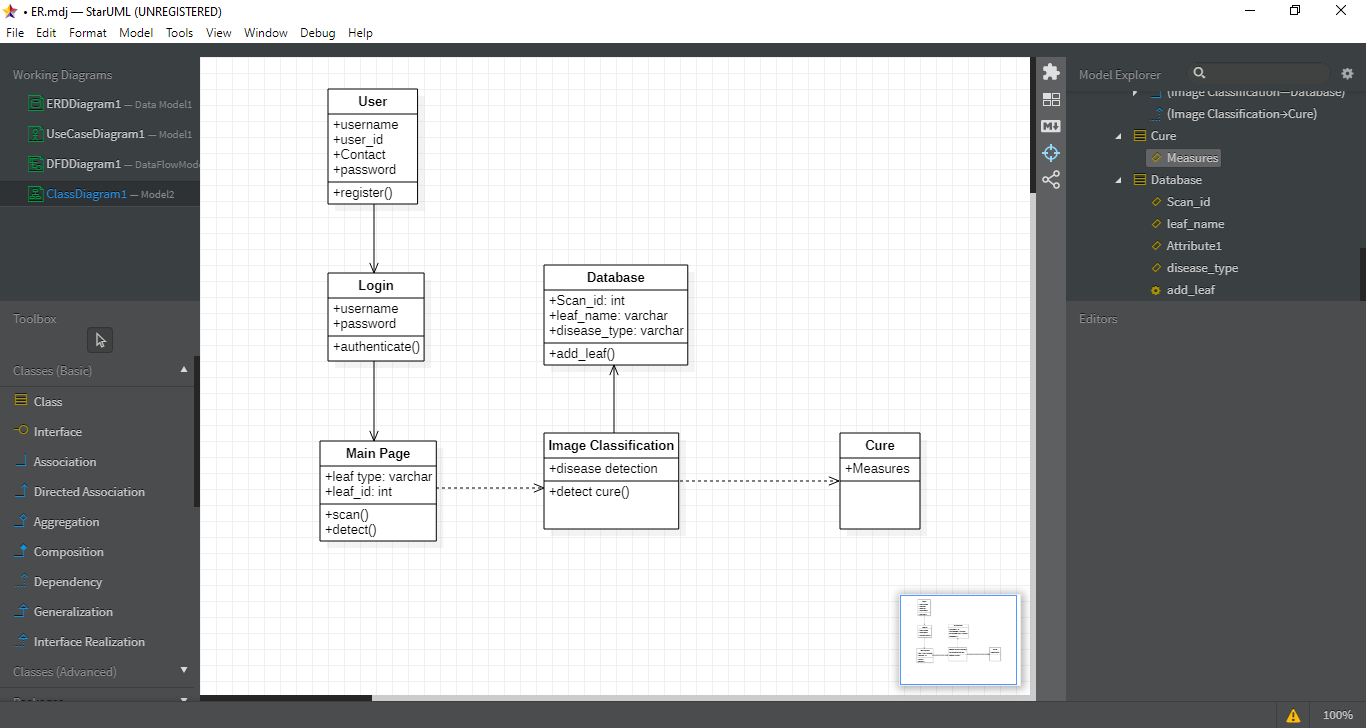
When a user clicks an image, it helps to extract the details from the image and gives more details related to the leaf and if disease detected.

**Module Description:**

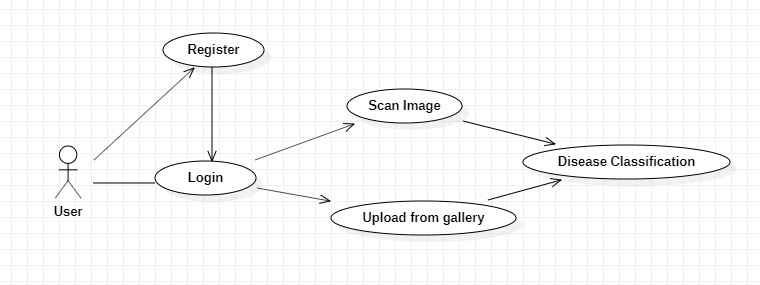
1. **Login/Registration:** This module will allow a user to login or either register themselves with a user-id, username, contact number and password. In case of login validation would be performed from the database before directing to the image scan screen. Whereas in case of sign up the details will be stored in the database for further validation.
2. **Image upload/scan:** Creating a module for scanning or uploading images from gallery and a dataset of pictures of diseased and healthy plant leaves under controlled conditions.
3. **Disease Detection:** Setting up data generators that will read pictures, convert them and feed them (with their labels) to our network and detect whether the given plant has disease or not.
4. **Treatment:** If a disease is detected, them on click of a button on the previous module this module will open and the method to cure the disease will be displayed.
   1. **Conceptual Models**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr No.** | **Event** | **Trigger** | **Source** | **Use Case** | **Response** | **Destination** |
| 1. | Welcome Screen | Log In/Sign Up | User | Option to choose login or signup. | Client directs to the page based on choice | User |
| 2. | Registration  Screen | Sign-Up | User | User details stored in database. | Client gets registered. | User(to login page) |
| 3. | Login Screen | Login Screen | User | User details verified. | Client gets logged in. | User |
| 4. | Image Scanning/Uploading from gallery | Scanning the plant leaf or choosing the image from gallery. | User | Scan the image. | Data is sent to system for disease detection | User |
| 5. | Disease Detected Page | When image scanned successfully and data is passed to the system | System | Provide disease name if detected or else no disease detected . | The disease name is displayed  to the user. | User |
| 6. | Treatment | Direct the user to the page that specifies ways to treat | System | Helps in curing from diseases. | Data is sent to records that the details for future. | Home(A data set created for future reference) |

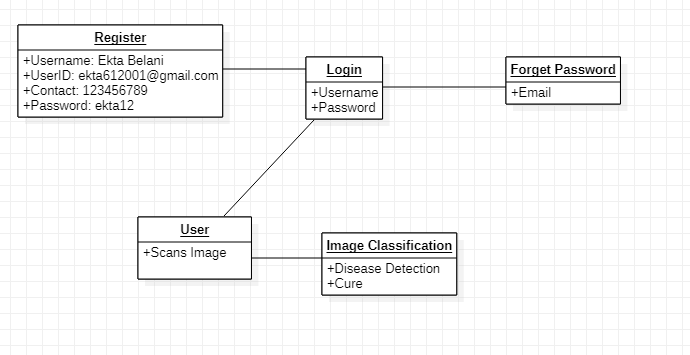
**Event Table**



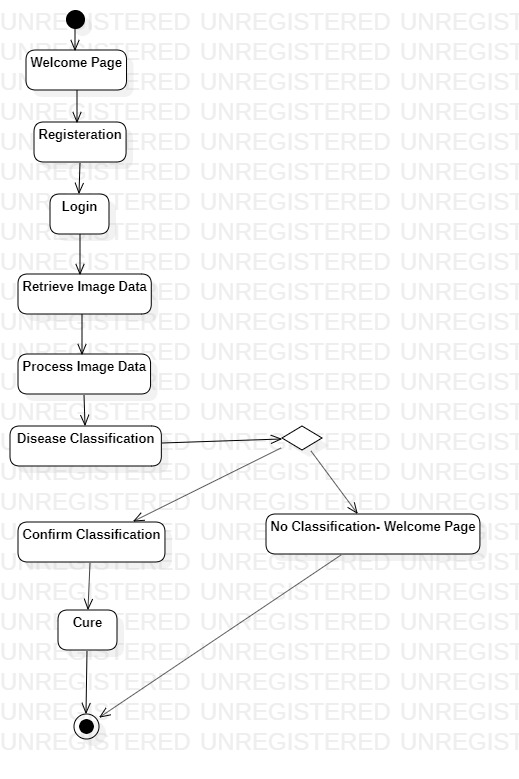
**Class Diagram**

****

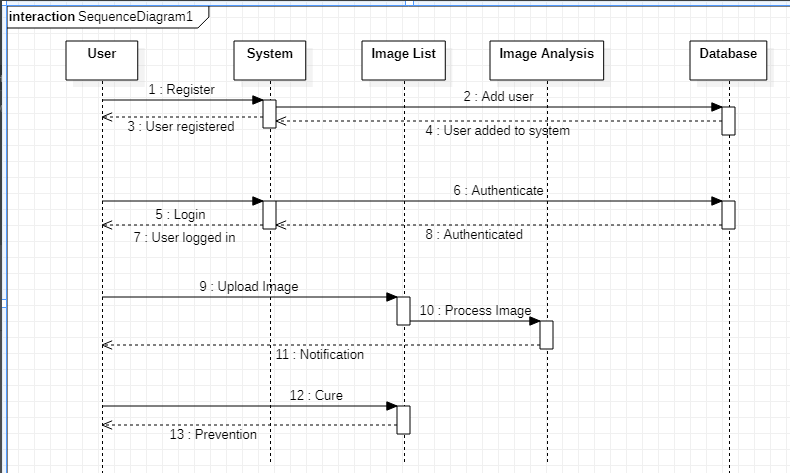
**Use Case Diagram**

****

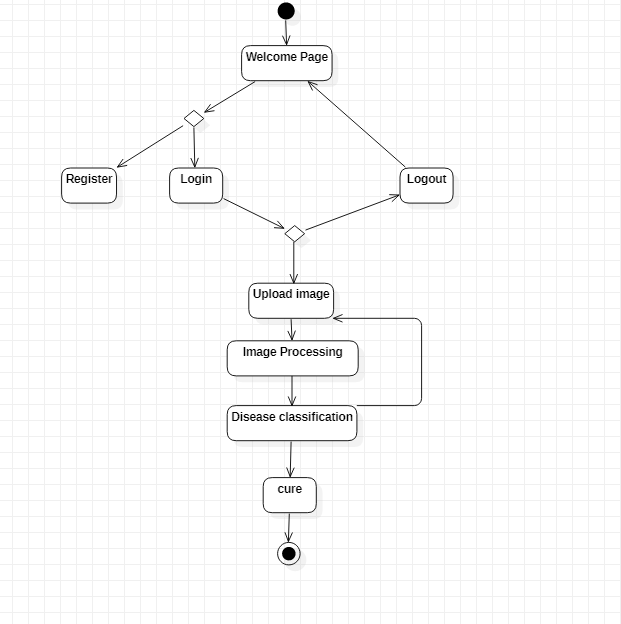
**Object Diagram**

****

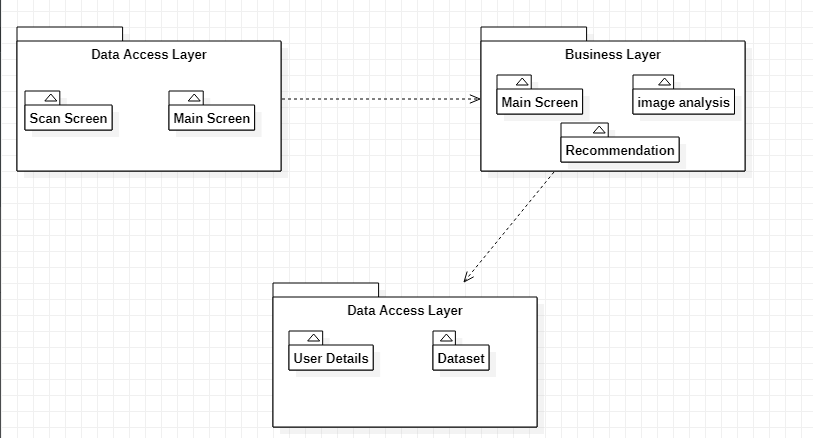
**Activity Diagram**

****

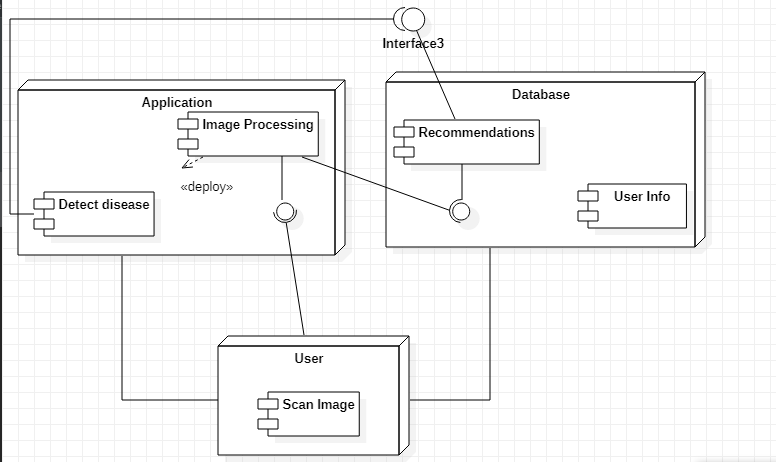
**Sequence Diagram**

****

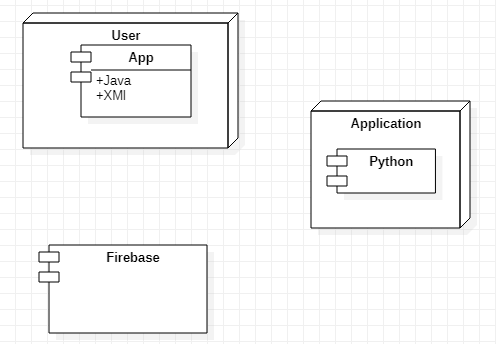
**State Chart Diagram**

****

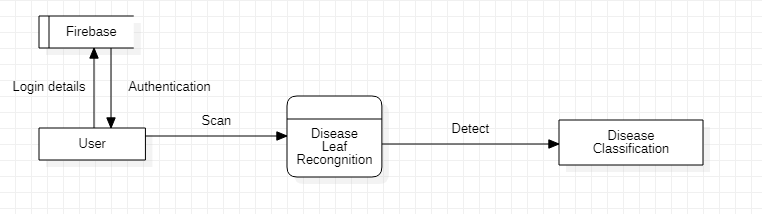
**Package Diagram**

****

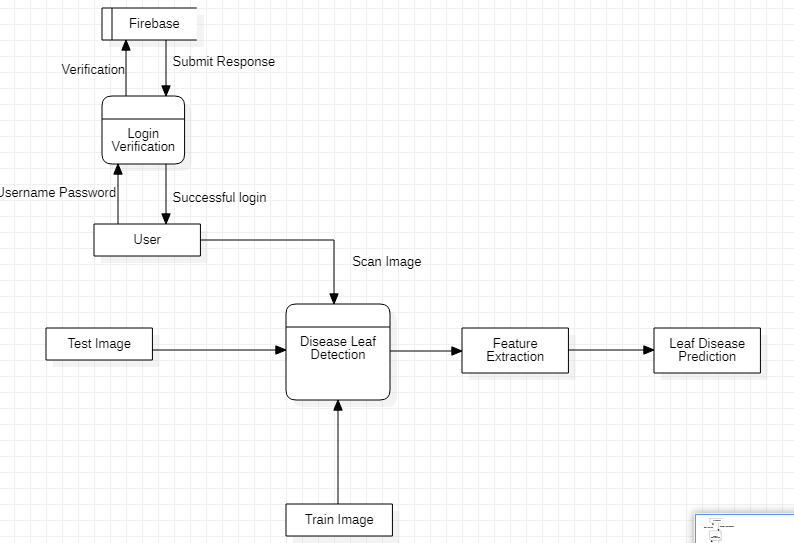
**Component Diagram**

****

**Deployment Diagram**

****

**Data Flow Diagram – Level 0**

****

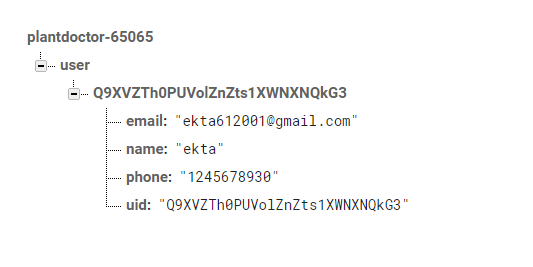
**Data Flow Diagram – Level 1**

**4. SYSTEM DESIGN**

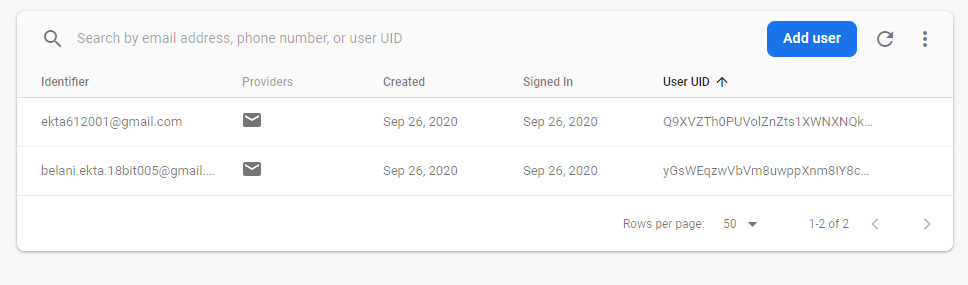
**4.1 Basic Modules**

1. **Login/Registration:** This module will allow a user to login or either register themselves with a user-id, username, contact number and password. In case of login validation would be performed from the database before directing to the image scan screen. Whereas in case of sign up the details will be stored in the database for further validation.
2. **Image upload/scan:** Creating a module for scanning or uploading images from gallery and a dataset of pictures of diseased and healthy plant leaves under controlled conditions.
3. **Disease Detection:** Setting up data generators that will read pictures, convert them and feed them (with their labels) to our network and detect whether the given plant has disease or not.
4. **Treatment:** If a disease is detected, them on click of a button on the previous module this module will open and the method to cure the disease will be displayed.
   1. **Data Design**
      1. **Schema design** **and Data Dictionary**

**Registrations-**



**Authentication :**



* + 1. **Data Integrity and Constraints**

{

"rules": {

".read": "now < 1603650600000", // 2020-10-26

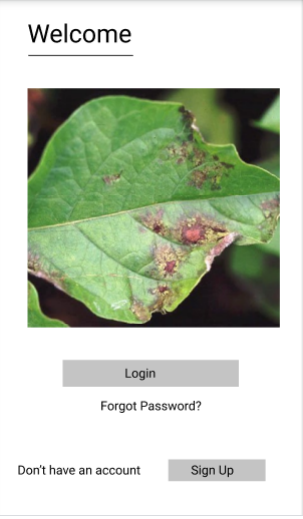
".write": "now < 1603650600000", // 2020-10-26

}

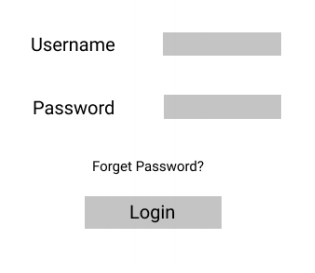
}

|  |  |
| --- | --- |
| **.read** | **Describes if and when data is allowed to be read by users.** |
| **.write** | **Describes if and when data is allowed to be written.** |

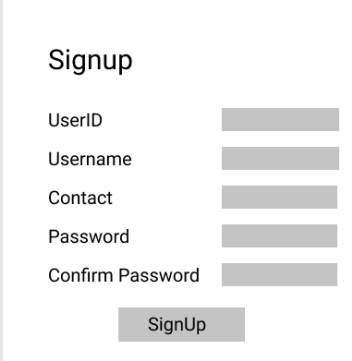
* 1. **UI Design**

****

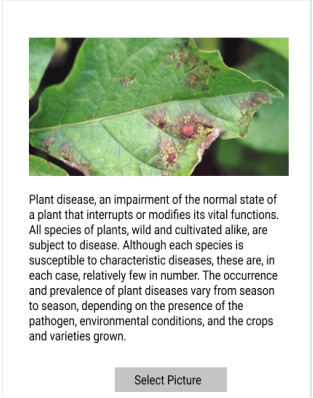
**Figure 1: Home Page**

****

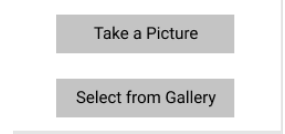
**Figure 2:Login Page**

****

**Figure 3:Sign Up**

****

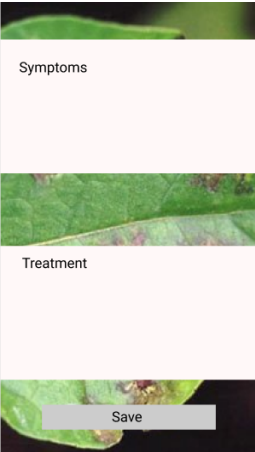
**Figure 4: Image Upload/Scan**

****

**Opens on click of Select Picture Button**

****

**Figure 5: Disease Detection**

****

**Figure 6: Symptoms and Treatment**

* 1. **Security Issues**

1. **Password Encryption:**

hash\_config {  
  algorithm: SCRYPT,  
  base64\_signer\_key: b/VgnvmisM3efz44DHREhgBnaQk1zjHwdzMxZRU2u34WT08g++RaUBHt2PIeZ9EdWIdyzCFOTia3PHcXX2Eq7A==,  
  base64\_salt\_separator: Bw==,  
  rounds: 8,  
  mem\_cost: 14,  
}

* 1. **Test Cases Design**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Condition** | **Input Selected** | **Expected** | **Actual** |
| **User**  **Login** | Username=””  Password=”” | Username and Password Fields are empty. | Username and Password Fields are empty. |
| Username=”ekta612001@gmail.com”  Password=”ekta612001” | Verify and then redirect to the homepage. | Verify and then redirect to the homepage. |
| **Forgot**  **Password?** | Email=”” | Please enter email | Please enter email |
| Email=”abc” | Please enter valid email. | Please enter valid email. |
| Email=”belani612001@gmail.com” | Email has been sent to your email-id. | Email has been sent to your email-id. |
| **Scan** | Textbox=Image | Image Displayed | Image Displayed |
| **Detect disease** | Image | Display disease detected. | Display disease detected. |

**5. IMPLEMENTATION AND TESTING**

**5.1 Implementation Approach**

* **Introduction**

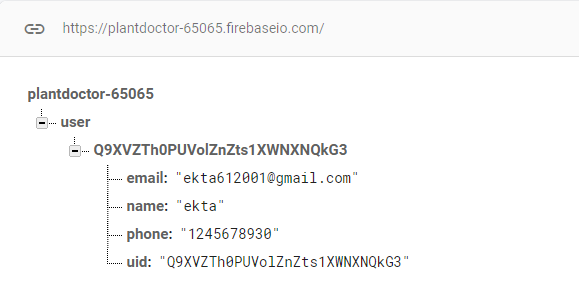
The implementation of the application starts with a welcome page which is the first page and provides a user options to either login or register just by one click. Upon successful login, it asks for permissions to use camera .Then the user will be redirected to the scanning page where they can scan the image. They can also add an image from gallery. Upon successful image upload detection of disease can be done by clicking the detect button and further cure can be seen. From the navigation panel, you can logout easily.

* **Input and Output Design Implementation**

To start my Project Implementation Plan, first, I had to find what modules do I need and which all software’s are required to install, what will the database schema look like, what dependencies are required, and what will the front UI of the application look like. Once everything is decided, I moved on to making the front end design of the application. This was followed by connecting the front end to the back end(firbase) and to connecting the application to the database and make them work in a flow.

* **Database Implementation**

This project is supported by firebase. Firebase provides login and user authentication functionality. Simple email authentication is enabled inside the firebase console in case the user forgets the password. It stores the details of a user – email, name, phone, uid.



* **Code Module**

This shows what Technology we are using for my Front-End and Back-End and which Database. So, I am using Java, XML and Python for Front-End and firebase for Back-End i.e.-for database. So, in my code firstly user will register or login and then scanning screen will be displayed. So the main functionality of my project will work i.e. The user can then scan or upload an image to detect the disease. **.**

* **System Implementation**

My project consists of multiple modules. Firstly, the welcome screen is visible after logging into the system the user is presented with a scanning screen .The user can scan or upload multiple images. In the same page, the user can see the disease detected. The user can next go to the cure tab where he/she can check for the cure of the detected disease. The users can then logout of the system.

* **Project Summary**

After logging into the system the user is presented with the scanning screen where the user can scan or upload an image from gallery which is then processed. On the same page the user can see the detected disease name by clicking a button. The user can then even see for the cure for the respective disease detected. The user can then logout of the system.

**5.2 Coding Details and Code Efficiency**

**Model:**

from tensorflow.compat.v1 import ConfigProto

from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()

config.gpu\_options.per\_process\_gpu\_memory\_fraction = 0.5

config.gpu\_options.allow\_growth = True

session = InteractiveSession(config=config)

from tensorflow.keras.layers import Input, Lambda, Dense, Flatten

from tensorflow.keras.models import Model

from tensorflow.keras.applications.inception\_v3 import InceptionV3

#from keras.applications.vgg16 import VGG16

from tensorflow.keras.applications.inception\_v3 import preprocess\_input

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator,load\_img

from tensorflow.keras.models import Sequential

import numpy as np

from glob import glob

#import matplotlib.pyplot as plt

! pip install -q kaggle

! mkdir ~/.kaggle

! cp kaggle.json ~/.kaggle/

! chmod 600 ~/.kaggle/kaggle.json

! kaggle datasets list

! kaggle datasets download -d kalpeshkhandelwal/plantvillagek

!unzip /content/plantvillagek.zip

# re-size all the images to this

IMAGE\_SIZE = [224, 224]

train\_path = '/content/PlantVillage/train'

valid\_path = '/content/PlantVillage/val'

inception = InceptionV3(input\_shape=IMAGE\_SIZE + [3], weights='imagenet', include\_top=False)

# don't train existing weights

for layer in inception.layers:

    layer.trainable = False

 # useful for getting number of output classes

folders = glob('/content/PlantVillage/train/\*')

# our layers - you can add more if you want

x = Flatten()(inception.output)

prediction = Dense(len(folders), activation='softmax')(x)

# create a model object

model = Model(inputs=inception.input, outputs=prediction)

# view the structure of the model

model.summary()

# tell the model what cost and optimization method to use

model.compile(

  loss='categorical\_crossentropy',

  optimizer='adam',

  metrics=['accuracy']

)

# Use the Image Data Generator to import the images from the dataset

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255, # Make sure you provide the same target size as initialied for the image size

training\_set = train\_datagen.flow\_from\_directory('/content/PlantVillage/train',

                                                 target\_size = (224, 224),

                                                 batch\_size = 32,

                                                 class\_mode = 'categorical')

test\_set = test\_datagen.flow\_from\_directory('/content/PlantVillage/val',

                                            target\_size = (224, 224),

                                            batch\_size = 32,

                                            class\_mode = 'categorical')

                                   shear\_range = 0.2,

                                   zoom\_range = 0.2,

                                   horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale = 1./255)

# fit the model

# Run the cell. It will take some time to execute

r = model.fit\_generator(

  training\_set,

  validation\_data=test\_set,

  epochs=10,

  steps\_per\_epoch=len(training\_set),

  validation\_steps=len(test\_set)

)

import matplotlib.pyplot as plt

# plot the loss

plt.plot(r.history['loss'], label='train loss')

plt.plot(r.history['val\_loss'], label='val loss')

plt.legend()

plt.show()

plt.savefig('LossVal\_loss')

# plot the accuracy

plt.plot(r.history['accuracy'], label='train acc')

plt.plot(r.history['val\_accuracy'], label='val acc')

plt.legend()

plt.show()

plt.savefig('AccVal\_acc')

# save it as a h5 file

from tensorflow.keras.models import load\_model

model.save('model\_inception.h5')

y\_pred = model.predict(test\_set)

y\_pred

import numpy as np

y\_pred = np.argmax(y\_pred, axis=1)

y\_pred

from tensorflow.keras.models import load\_model

from tensorflow.keras.preprocessing import image

from keras.applications.resnet50 import ResNet50

# load and evaluate a saved model

from numpy import loadtxt

from keras.models import load\_model

# load model

model = load\_model('model\_inception.h5')

# summarize model.

model.summary()

img\_data

img=image.load\_img('/t1.jpg',target\_size=(224,224))

x=image.img\_to\_array(img)

x

x.shape

x=x/255

#import numpy as np

#x=np.expand\_dims(x,axis=0)

#img\_data=preprocess\_input(x)

#img\_data.shape

model.predict(img\_data)

a=np.argmax(model.predict(img\_data), axis=1)

a==1

import tensorflow as tf

tf.\_\_version\_\_

import numpy as np

from google.colab import files

from keras.preprocessing import image

img=image.load\_img('/t1.jpg',target\_size=(224,224))

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

images = np.vstack([x])

classes = model.predict(images, batch\_size=10)

print(classes)

import cv2

# Utility

import itertools

import random

from collections import Counter

from glob import iglob

def load\_image(filename):

    img = cv2.imread(os.path.join(train\_path,valid\_path,filename))

    img = cv2.resize(img, (IMAGE\_SIZE[0], IMAGE\_SIZE[1]) )

    img = img /255

    return img

def predict(image):

    probabilities = model.predict(np.asarray([img]))[0]

    class\_idx = np.argmax(probabilities)

    return {classes[class\_idx]: probabilities[class\_idx]}

import os

for idx, filename in enumerate(random.sample(training\_set.filenames, 5)):

    print("SOURCE: class: %s, file: %s" % (os.path.split(filename)[0], filename))

    img = load\_image(filename)

    prediction = predict(img)

    print("PREDICTED: class: %s, confidence: %f" % (list(prediction.keys())[0], list(prediction.values())[0]))

    plt.imshow(img)

    plt.figure(idx)

    plt.show()

import tensorflow as tf

!mkdir "tflite\_models"

TFLITE\_MODEL = "tflite\_models/plant\_disease\_model.tflite"

# Get the concrete function from the Keras model.

converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

tflite\_model = converter.convert()

# Save the model.

with open('model.tflite', 'wb') as f:

  f.write(tflite\_model)

**5.2.1 Code Efficiency**

Python is used for training models and for code efficiency. A machine learning model is a file trained to recognize certain patterns. It is trained over a set of data called training dataset. It uses an algorithm to learn from those data.

**5.3 Testing Approach**

In my project I have applied several testing methods such as Unit Testing, Integration Testing, System Testing, White Box testing, Black Box Testing and Regression testing. Regression Testing is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

**5.3.1 Unit Testing**

In unit testing you test each module of the project separately. In this application each module such as registration, login, scanning is tested individually to see if it is possible to make it to the next scene.

**5.3.2 Integration Testing**

Integration Testing means testing more than one Module at a time. Taking the same example, the registration and forget password had to be checked if they worked together e.g. if registering and forgetting password, reflected changes in the database.

**5.3.3 System Testing**

System testing is conducted on a complete integrated system to evaluate the system’s compliance with its specified requirements. System testing takes, as its input, all of the integrated components that have passed integration testing. The application has successfully passed the system testing as per the objectives defined and has been tested on multiple devices

.

**5.4 Modification and Improvements**

After testing and handling all errors, exceptions and bugs, errors were solved by troubleshooting, exceptions were handled by try catch block at the correct places avoiding redundancy of code from index files. Bugs can never be removed completely but most of them are solved and remaining is under development.

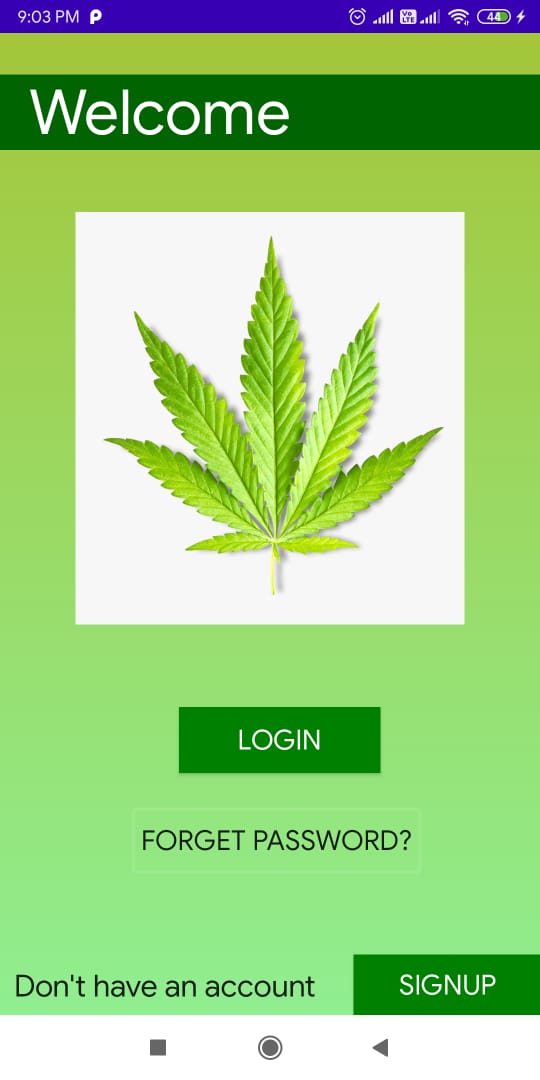
**6. RESULTS AND DISCUSSIONS**

**6.1 TEST REPORTS**

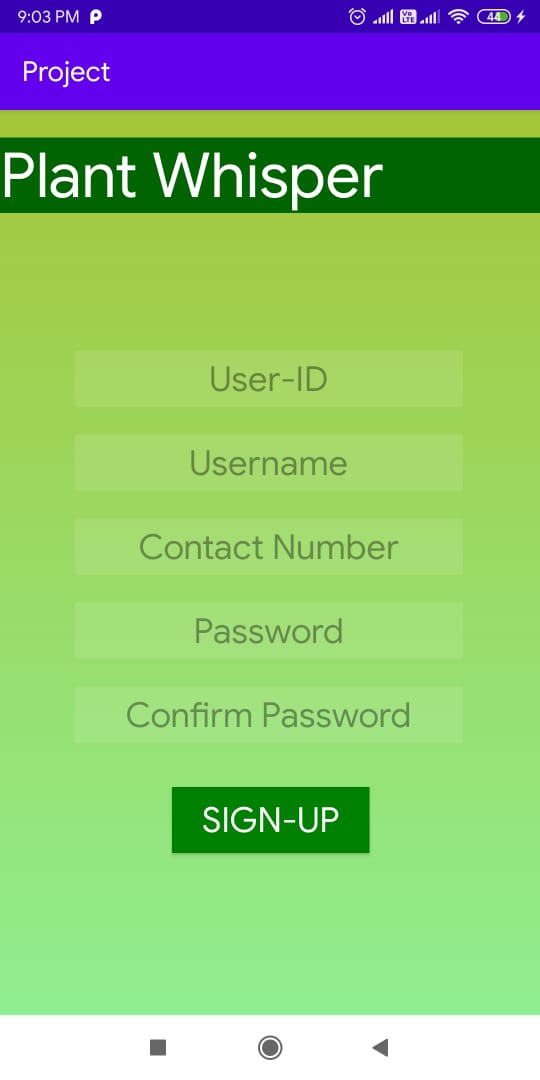
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Condition** | **Input** | **Expected Result** | **Actual Result** |
| **Registration** | Email, Phone number, Password | Enter all details  Validate entered details  Add user to Realtime database. | Enter all details  Validate entered details  Add user to Realtime database. |
| **Log In** | Email  Password | Enter all details  Validate entered details  If existing user, redirect to homepage. | Enter all details  Validate entered details  If existing user, redirect to homepage. |
| **Scan Image** | Image | The image should be scanned or uploaded and processed to detect the disease. | The image should be scanned or uploaded and processed to detect the disease. |
| **Detect Disease** | Button Press | User should click the button to detect the disease. | User should click the button to detect the disease. |
| **Get Cause and Cure** | Button Press | User clicks the button to get the cure and cause. | User clicks the button to get the cure and cause. |
| **Live Chat** | Button Press | Live chat opens. | Live chat opens. |

**Figure 26: Test Report**

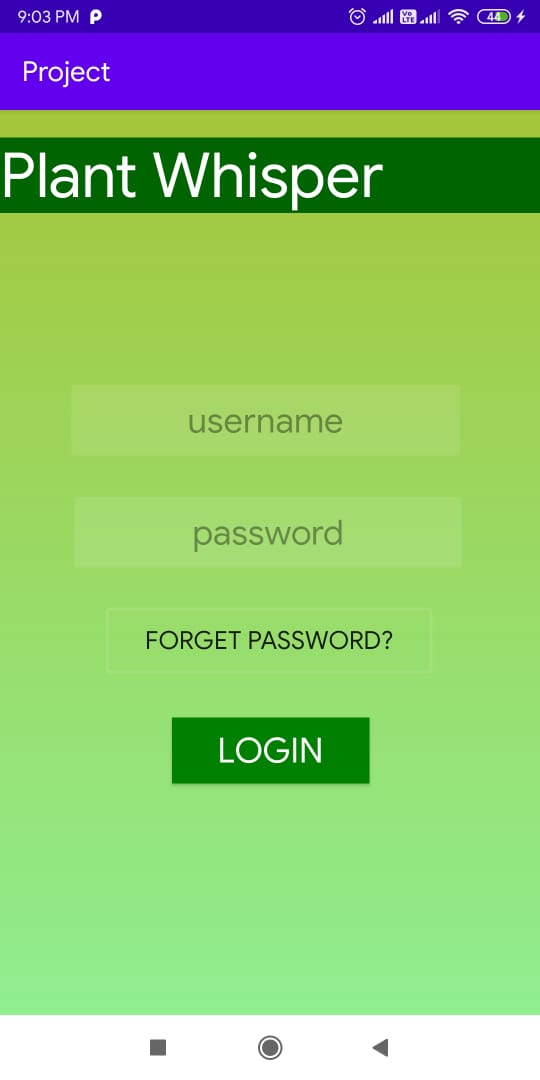
**6.2 USER DOCUMENTATION AND SCREENSHOTS**

****

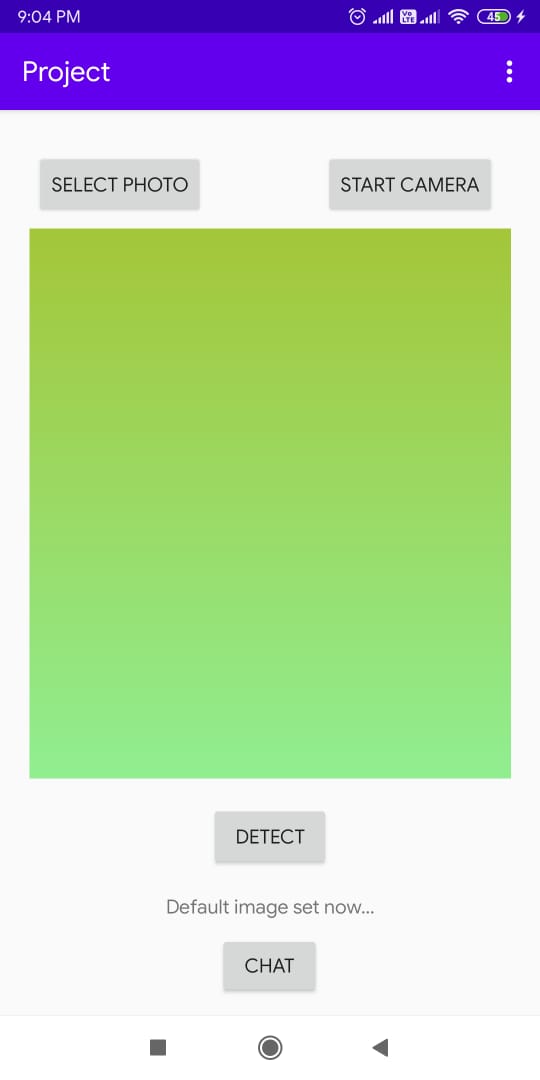
**Figure 27: Welcome Page**

****

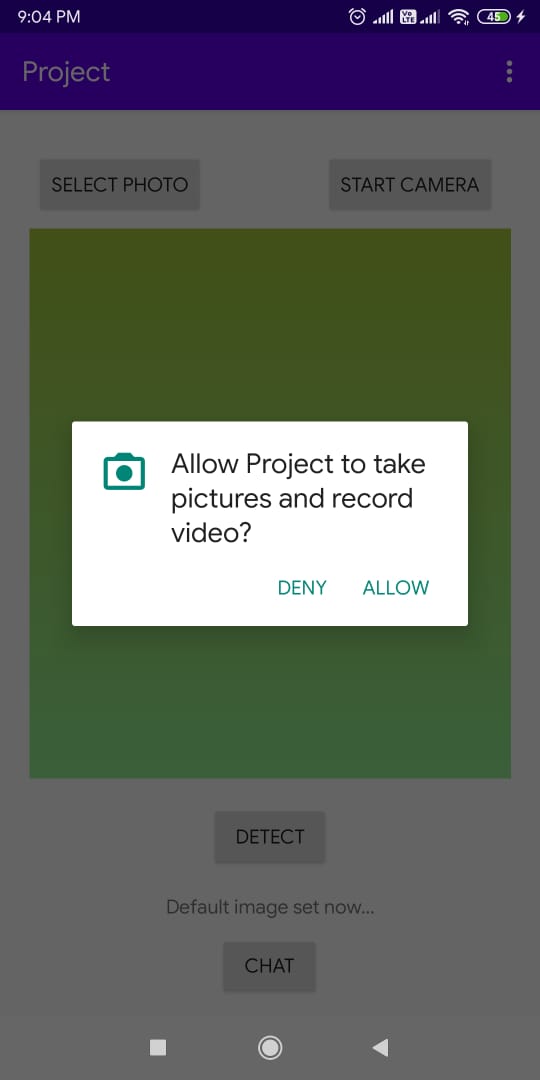
**Figure 28: Sign Up Page**

****

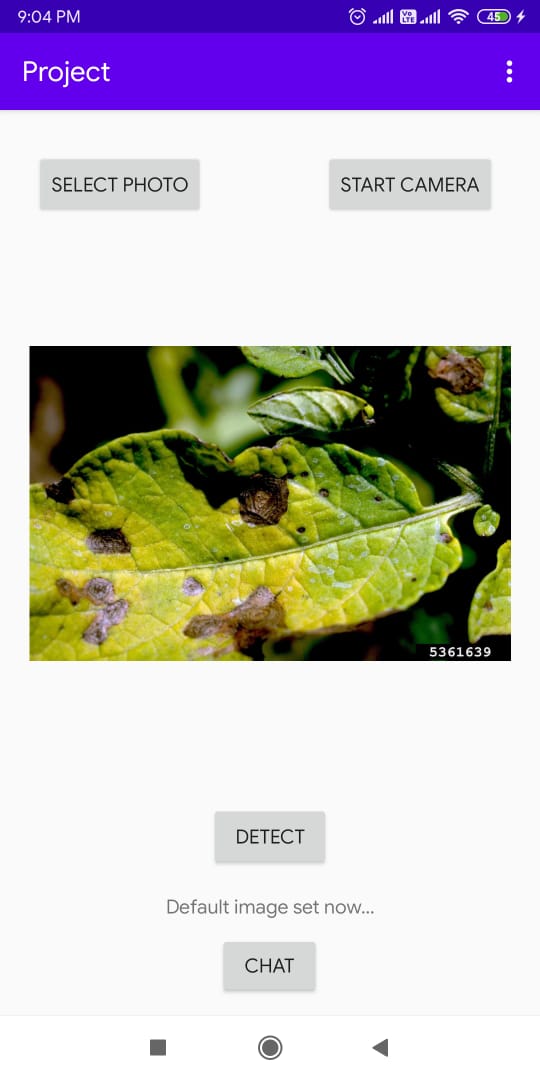
**Figure 29: SignIn Page**

****

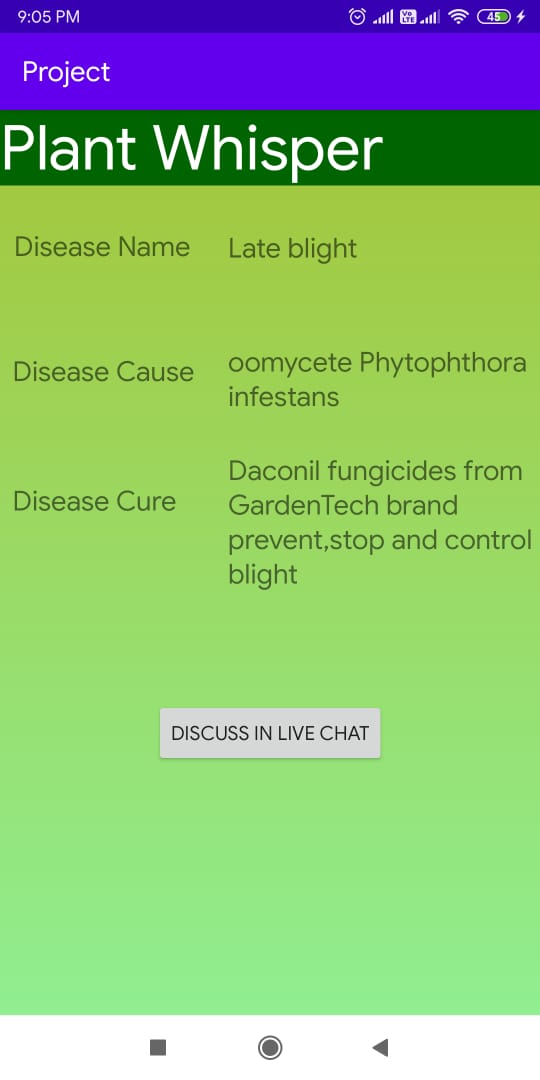
**Figure 30: Home Screen (Scanning Module)**

****

**Figure 31: Grant Permission**

****

**Figure 32: Disease Detection**

****

**Figure 33: Disease Cause and Cure**

****

**Figure 34: Live Chat**