ABSTRACT

When plants and crops are affected by pests it affects the economy of the country. Usually farmers observe the plants with bare eyes to check and identify disease. But this method can be time consuming, expensive and inaccurate.

Automatic detection using image processing provides faster and accurate results. This project is associated with an approach for plant leaf disease recognition application, which is based on classification of leaf image, using deep convolutional networks. The developments in computer vision provides the possibility in increasing and to enhance the detection of plant leaf diseases. All the needed steps for using the disease recognition model are explained in the project including deep learning framework to perform the CNN training. This project is an approach in identifying and detecting plant leaf diseases with the help of the deep convolutional neural network which is trained for the database containing plant leaves that were gathered for different plant diseases. The developed model has great simplicity. Diseased and Healthy leaves are in according to other classes, which enables the model to differentiate among different diseased and healthy leaves by using deep CNN.

Keywords: Deep learning, Image processing, Convolutional neural networks, Plant disease

INDEX

1. INTRODUCTION:

1.1 Purpose of System:

Plant leaf disease detection is very difficult in agriculture. Incorrect detection leads to many problems and might result expensive. Plant leaf disease detection involves having huge knowledge about the plant diseases, leaf diseases and various other parameters. Lacking knowledge in any of the parameters may lead to incorrect detection.

However, to overcome the above problem, the project uses the convolutional neural networks technique of deep leaning. High accuracy of detection is what our system offers and it benefits a huge deal to the farmers. It decreases the total economic loss caused due to the incorrect detection. Identification of disease involves loading the image, checking against the model and intimating the user with the name of the disease. The system also provides the output in speech.

1.2 Existing System:

Normally the diseases occurring in the plants usually take a lot of time for the farmers to recognize and also is very error prone. Farmers generally check for the symptoms with the naked eye which can often be undetected. Though there many systems which use ML techniques, there are many accuracy problems which might turn out to be expensive.

Disadvantages:

The existing system is time taking and highly error prone. Symptoms are often undetected with naked eye. Though there are few ML techniques, they are having accuracy problems which might not be correct every time.

1.3 Proposed System:

- The project is an android application which uses deep learning techniques to identify the diseases more accurately and efficiently.
- It generally produces higher accuracy and confidence.
- We used convolution neural network of deep learning model in our project.

Advantages:

1. High accuracy and confidence.
2. Highly reliable.
3. Better than the existing system.

2. SYSTEM REQUIREMENTS:

2.1 Hardware System Configuration:

RAM - 8GB

Hard Disk - 4 GB

Keyboard - Standard Windows Keyboard

Monitor - 1280x800 screen resolution

Android phone with RAM - 2 GB or higher

Internet Connectivity

2.2 Software System Configuration:

Development Platform : Android Studio

Operating System : Windows 7/8/10

Model Development : Google Collab

Database Connectivity : Firebase Database

Android version : Version 7 or higher

2.3 MODULES OF THE SYSTEM

1. Model
2. Application
3. Database

2.3.1 Model:

The dataset taken for the model is divided as training and testing dataset. The training dataset is trained against 32 different plant disease classes. The testing set is, therefore, tested against the trained set and the disease is recognized. If the leaf is not infected with any of the diseases then the result is returned as normal. The model uses Convolutional Neural Networks(CNN) algorithm to classify the labels disease-wise. The accuracy of the model highly depends on the algorithm.

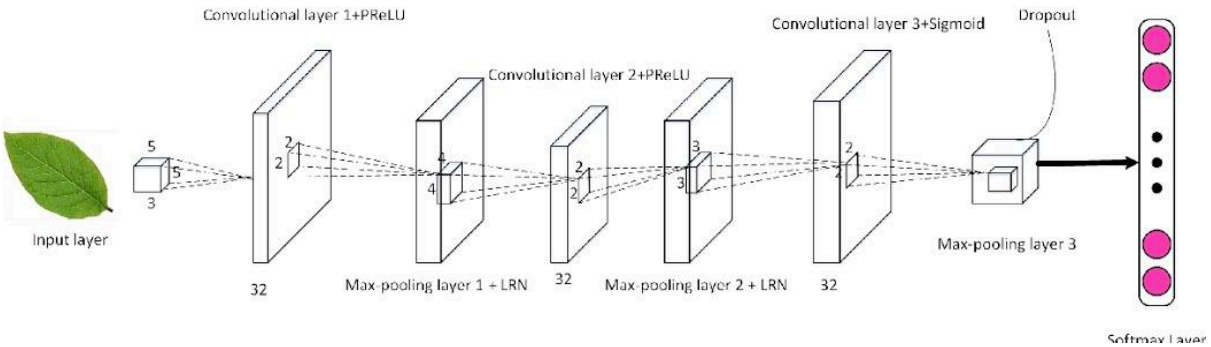


Fig. 2.3.1 CNN skeleton for the model

2.3.2 Application:

To provide the interface for the model we create an Android application. The application is as user friendly as it can be and understandable by all. Firstly, the user of the application must enter his name and mobile number. Secondly, we load the image either from the photos or we can take the photo and choose detect option. Then the control is shifted to the model and the processing takes place and the result is displayed in the application for the user. The voiceover feature in the application is activated once we choose the speak option which makes it even more easy to even farmers who can't read the text in the application.

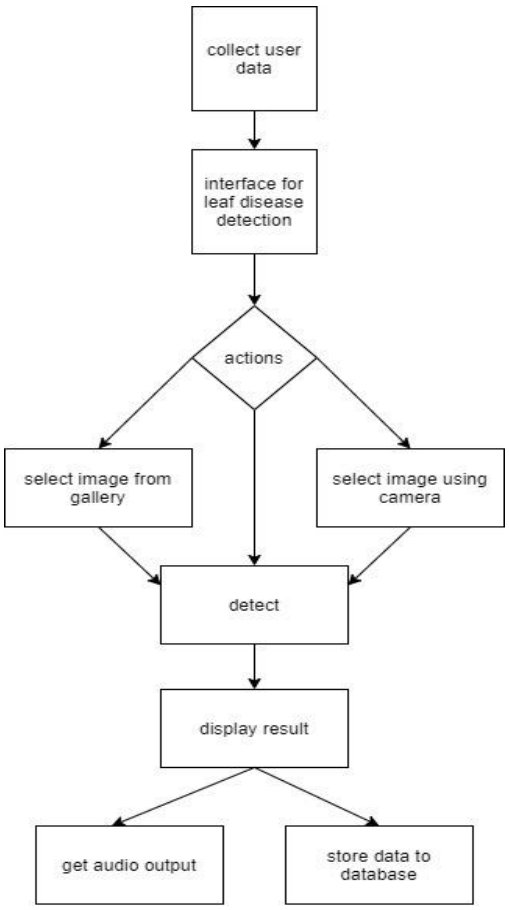


Fig. 2.3.2 Application Flow Diagram

2.3.3 Database:

The database is also connected to the application to store the information about all the users using the application. The Database stores the User data and the results. The database has several statistical tools and can store large data. The data stored can be used for improving the disease detection CNN model and can also be used to find various insights regarding the agricultural practices.

Fig. 2.3.3. Database Flow

3. SYSTEM ANALYSIS:

3.1. FEASIBILITY-STUDY

The Project's feasibility is studied in this stage. During the system evaluation, the feasibility of the proposed system is to be performed and checked. This is needed to make sure that the specified system works as required without any errors. For the feasibility analysis, knowledge regarding the requirements of the system is important.
Three main considerations needed in the analysis are

3.1.1. ECONOMICAL-FEASIBILITY

3.1.2. TECHNICAL-FEASIBILITY

3.1.3. SOCIAL-FEASIBILITY

3.1.1. ECONOMICAL-FEASIBILITY

This phase is carried for testing the financial effect that the project may place on the user or on organization .This is because the funds which can be poured for the research and developing of the project is bounded. Thus the proposed project should be within the specified budget and it can be achieved as most of the required technologies being used in the project are open source and can be availed freely.

3.1.2. TECHNICAL-FEASIBILITY

This stage is needed to test for the technical-feasibility, i.e., the technical specifications of the project. Any project developed should not have high need on the technical resources which are hard to maintain. It results in high demand of resources on the client-side. The proposed project must have a set of resource requirements, as minimum or null. The Project should be able to adapt to changing technologies.

3.1.3. SOCIAL-FEASIBILITY

The aim of this stage is to test the level of achieved acceptance of the project needed to the user. It also consists of the ability of the user to work with the project efficiently. The user should not feel endangered in using the system, but should accept and use it as a need. The Acceptance level of the users mainly dependent on the various methods that were used to enlighten the user regarding the system and to familiarize the project to them. The level in confidence should be increased so that the user is capable of making some feedbacks and are accepted, as the user is the target of the project.

3.2. SOFTWARE ENVIRONMENT

3.2.1 Android-Studio:-

Android-Studio is the Integrated Development Environment referred as (IDE) for Android Application Development. It is launched by Google for Android Operating System ,which is built on the JetBrains Intelli J software and especially designed for Android. Android Studio is available on, Linux based Systems, Windows and macOS.

The Android Software Development Kit (SDK) consists of set of needed development tools. It has documentation, debugger, sample code, handset emulator, libraries etc. The Supported platforms can be computers running MAC OS, Linux or Windows.

Android is an Operating System based on Linux developed for devices including smartphones and tablets and is open sourced. Android was first developed by Open-Handset Alliance.

The Studio can develop applications using either Java or Kotlin programming languages. The Kotlin programming language is very effective and beneficial when compared to android with Java programming language.

3.2.2. Deep-Learning

Deep-learning is one among various classes of machine learning algorithms that uses multi layers to get greater level insights from the provided data. For instance, in image-processing the lower level layers can detect and identify the edges, where as the higher layers helps to detect the data relevant to a human which includes letters or digits or faces etc. Most modern deep learning models use Convolutional Neural-Networks (CNN)s.

In deep-learning, every level extracts and transform the input into a more brief and suitable representation. For an image-recognition application, the input can be matrix of pixels and the representational layer can include abstraction of pixels, detect and encode the edges. The second layer can compose and encode the edges. The next third layer is able to encode the nose and the eyes. The fourth layer can also detect the images containing a face. A Deep-learning process is able to learn to work on which features to be placed optimally on which level by deciding of its own.

3.2.3. Convolutional Neural Networks (CNN)s

The Convolutional Neural-Network consists of the network which can use a mathematical operation known as Convolution. The Convolution is the special form for linear operation. Convolutional networks can be termed as simply neural-networks that can use convolution in place of generally used matrix-multiplication in minimum of one of it's layers. In deep-learning, a Convolutional Neural-Network (CNN) is one among the classes of deep neural-network. They are mostly used for analysis for the visual imagery. They can also be called as space-invariant artificial neural-networks which works with the shared weights architecture. They are also used in language processing, recommender systems, medical image analysis, image classification, image and video recognition and image classification.

The Convolutional Neural-Network contains the input layer, output layer, and other multi-hidden layers. The hidden-layers of the CNN usually include a set of convolutional-layers that are developed with the multiplication/other dot product. It can be represented as either cross-correlation or sliding dot product . This has much importance for the value of indices in matrix, and has influence on how weight can be specified at specific index points.

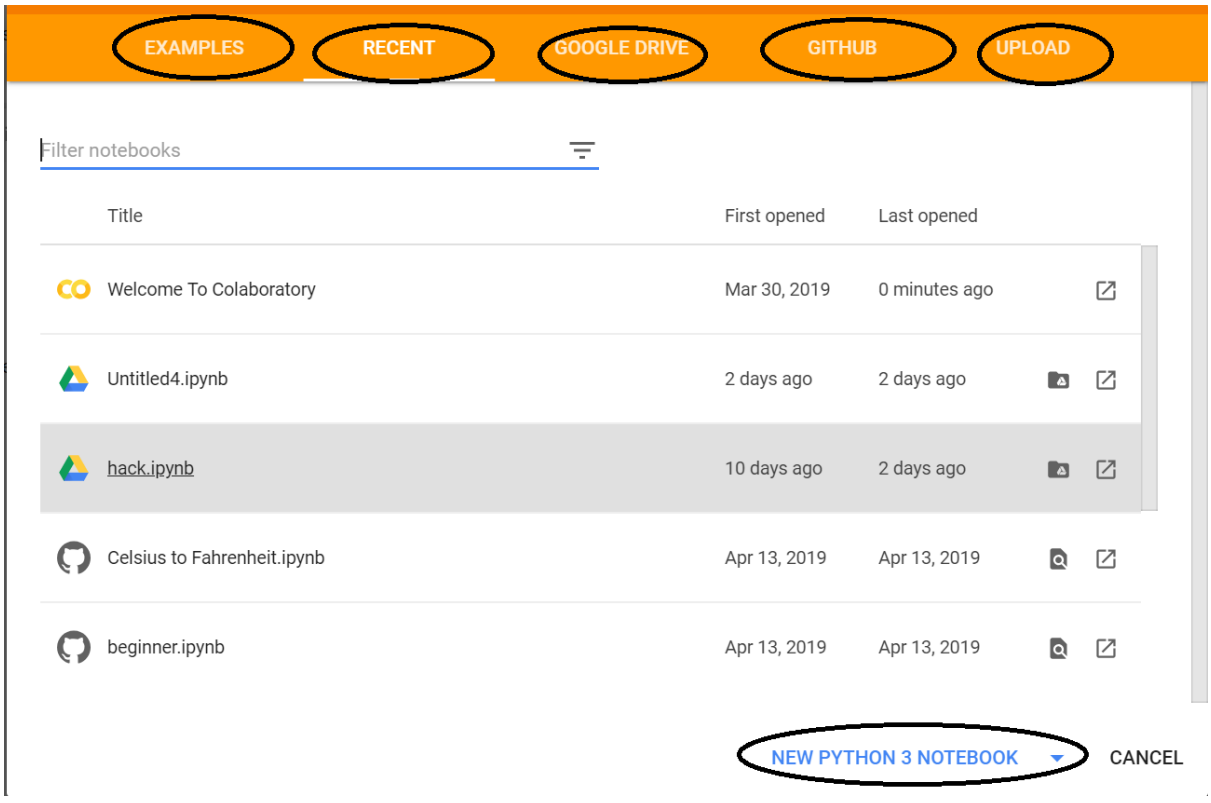
3.2.3 Google Colab

Colaboratory, or "Colab" is one of the products from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is well suited for data analysis, machine learning and education.

Colab is a hosted Jupyter notebook service that doesn't require any setup to use, but can provide free access to computing resources including GPUs. It is free to use.

Colab allows you to use and share Jupyter notebooks with others without having to download, install, edit or run anything. Colab notebooks allow you to combine multiple executable code and text into a single document, along with images, HTML and more.

When you create your own Colab notebooks, they are stored in your Google Drive account. With Colab you can harness the full power of popular Python libraries to analyse and visualize data. The code cell below uses NumPy to generate some random data and uses matplotlib to visualize it.



3.2.4. Firebase-Database

The Firebase-Realtime Database allows building of functional applications by making secure connection to the database through client-side application. The Information can be preserved locally, to provide the user a more responsive and effective experience. If the device reconnects, the Realtime-Database can sync with the local data for the changes that occurred.

The Realtime-Database has expression-based, flexible rules language, known as Firebase Realtime-Database Security Rules, which declare on how the information must dependent and when records may be written/read. If the project is included with Firebase-Authentication, we can also outline who can have ability to access to what kind of data, and how can they use it.

The Realtime Database is No-SQL based database and provides various optimality and functional features in comparison to the relational-database. The Realtime-Database API is made to best accept operations that need to be done quickly. Thus, It allows to make a real-time experience that provides the user reducing the lacking of responsiveness. Hence, it is also essential for considering on how the user can have access to the stored data.

4. SYSTEM DESIGN

4.1. INPUT DESIGN

The input design is considered as the link among the model and user. This includes taking user information i.e., the username and mobile number. Then the design enables the user to upload the image either from the existing photos or take a new photo at that instant. Then choosing the detect option gives us the result. The speak option gives us the voiceover. The designing of the input mainly focuses on how to control the quantity of input needed, to control the changes, and avoiding any delay. It is designed so that it can present secure and simplicity of use.

The Input Design should consider the following:

- Data that has to be provided for input?
- Ability to direct the user to provide appropriate input.
- The Methods needed for validation of inputs and the steps needed to follow if any error occur while using the system.

OBJECTIVES:

1. The Design changes of the user data from input to the computer-based system. The design needed to avoid any errors in the process and also shows appropriate direction to get necessary data from the system.

2. The data submitted is tested to verify data validity. The Data may be provided with the use of various screens provided in the system. Suitable alerts are shown such that the user can understand the system.

3. This can be accomplished with developing screens for the data to deal with large extent of data. The main aim of designing such input screens is to build the process of providing the data with seamless and to minimize errors. The screen is made such that the information manipulation if needed to be done.

4.2. OUTPUT DESIGN

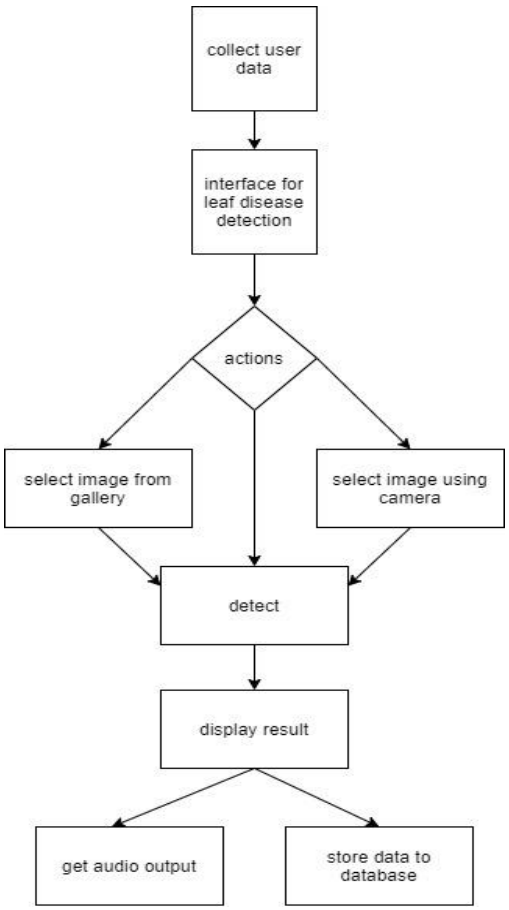
The Design of Output should meet the needs of the user and should present the data clearly. The processing data are provided to the user, and stored in the internal system for further usage. In design, it is specified on how the information is provided for access and can provide a output if needed. Appropriate output design enhances the system's functionality to provide the user decision-making ability.

1. Creating needed reports, documents, or other formats may contain data produced by the system.

2. Select needed techniques and ways for representing information.

3. Output design must continue in well-formed manner, in ordered the appropriate output need made along with making sure every element is made so that user finds the system effective as well as easy to use. By analyzing the output, we can identify that the specific output which is essential to satisfy the requirements.

4.3. SYSTEM ARCHITECTURE



System Architecture

This diagram represents the project's architecture. It focuses on the flow of all activities from one activity to another. The App collects data from user, then detects the disease and stores the data to database. The result is displayed at the end as output to the project.

4.4. UML Concepts

The UML is the most commonly used language for the making of software blue-prints. The UML stands for Unified Modelling Language. It is the language used for

- Specifying
- Visualizing
- Documenting
- Constructing

It contains vocabulary and related guidelines for composing parts for communication. Thus, Modelling language is the one whose guidelines and vocabulary focuses both on conceptual as well as physical representation of the system.

4.4.1Building Blocks of UML:

The UML mainly comprises of three types of building blocks

- Relationships,
- Things and
- Diagrams

Relationships tend to keep the things together.
Things can be termed as the abstractions which are high-quality components of the model.
Diagrams have to group things into collections.

Things in UML:

The various types of things present in UML are:

- Annotational things,
- Grouping things,
- Behavioral things and
- Structural things

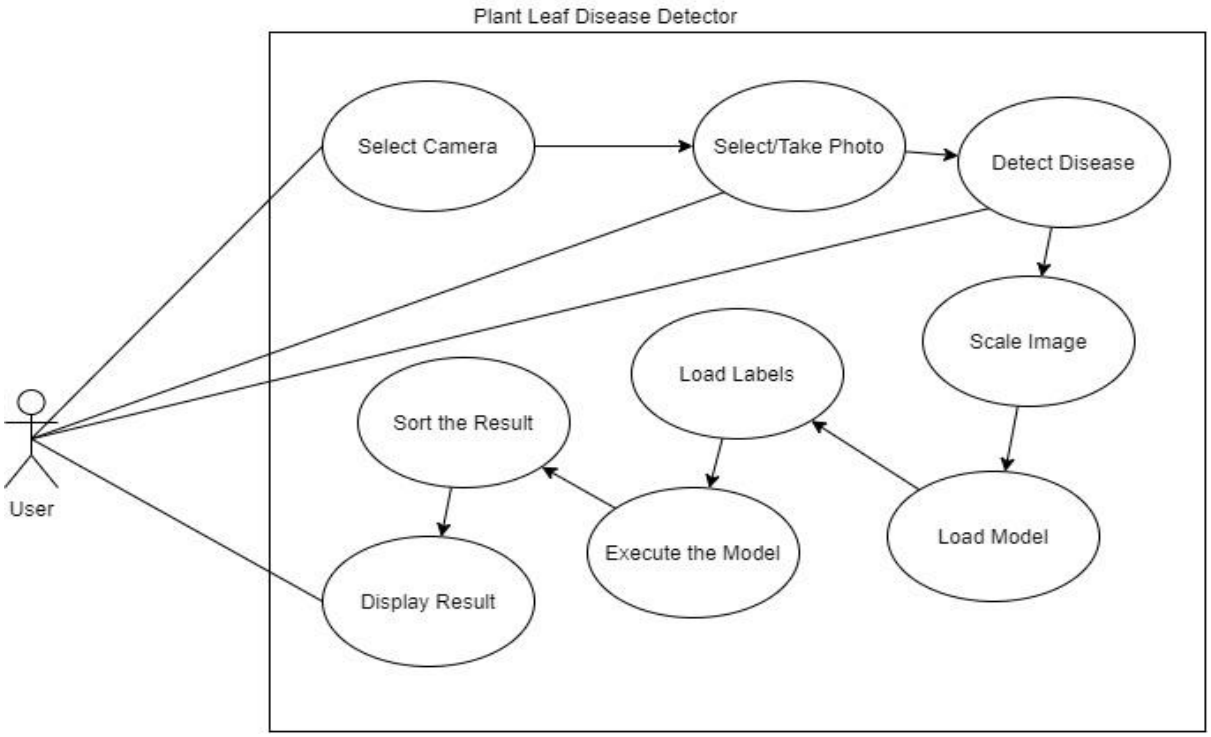
Relationships in UML:

The different types of relationships in UML are:

- Generalization
- Dependency
- Realization
- Association

4.4.3. UML-DIAGRAMS:

1. USE-CASE DIAGRAM:

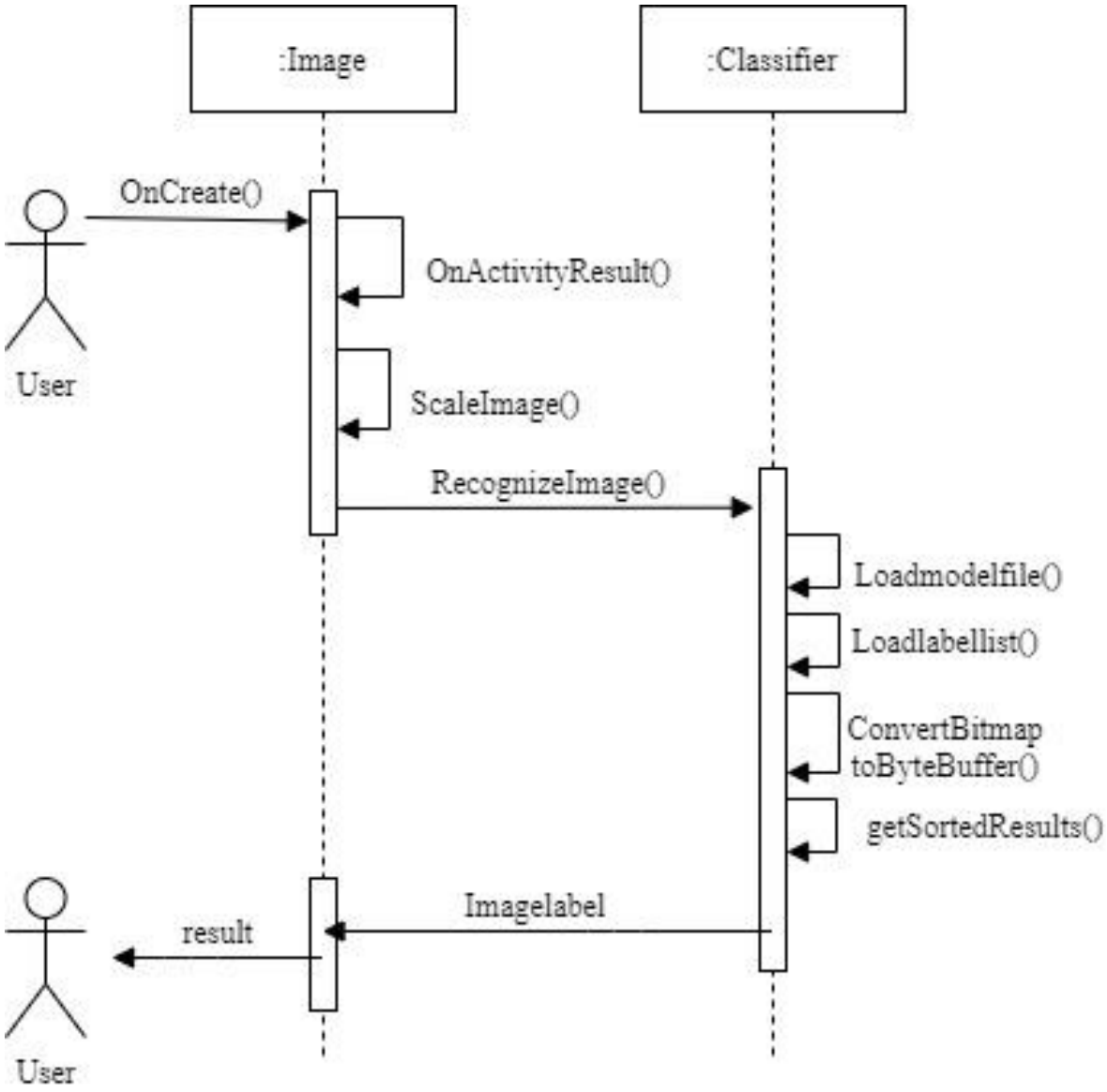


USE-CASE DIAGRAM

This is a Use-Case diagram representing all the major actions in our project. The actions are executed by the user while using the app.

Use-Case diagram consists of the actions represented using the ovals which are performed in the system and the user is represented with the help of actors like symbols in system. The links between user to various actions mean the user can perform them as shown.

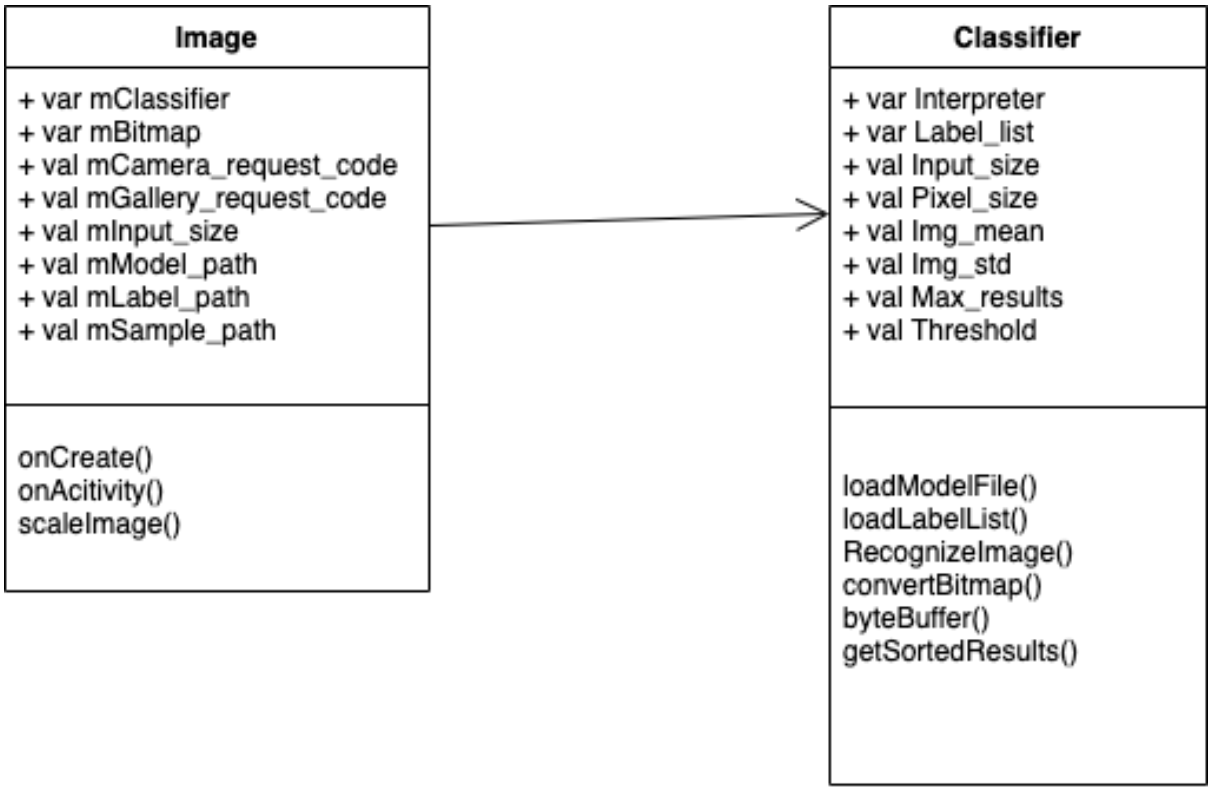
2. SEQUENCE DIAGRAM



SEQUENCE DIAGRAM

This diagram represents various interactions among objects specifically arranged accordingly. This shows the various classes present and messages exchanged between these class objects to carry out the functionalities. Image and classifier are the two classes interacting with various methods. User (actor) is the one who initiates the interaction and gets the result.

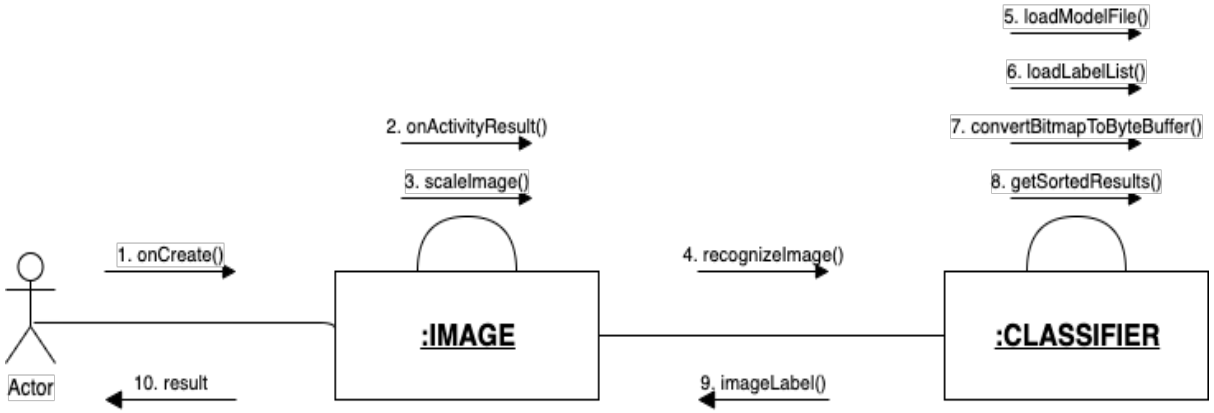
3. CLASS-DIAGRAM:



CLASS DIAGRAM

This diagram shows the class diagram of the project which includes various classes needed for the system and the relationship among them in the system. A Class is generally represented by rectangle box. The top portion contains class name. The second portion contains attributes(values). The bottom portion contains the methods of the class.

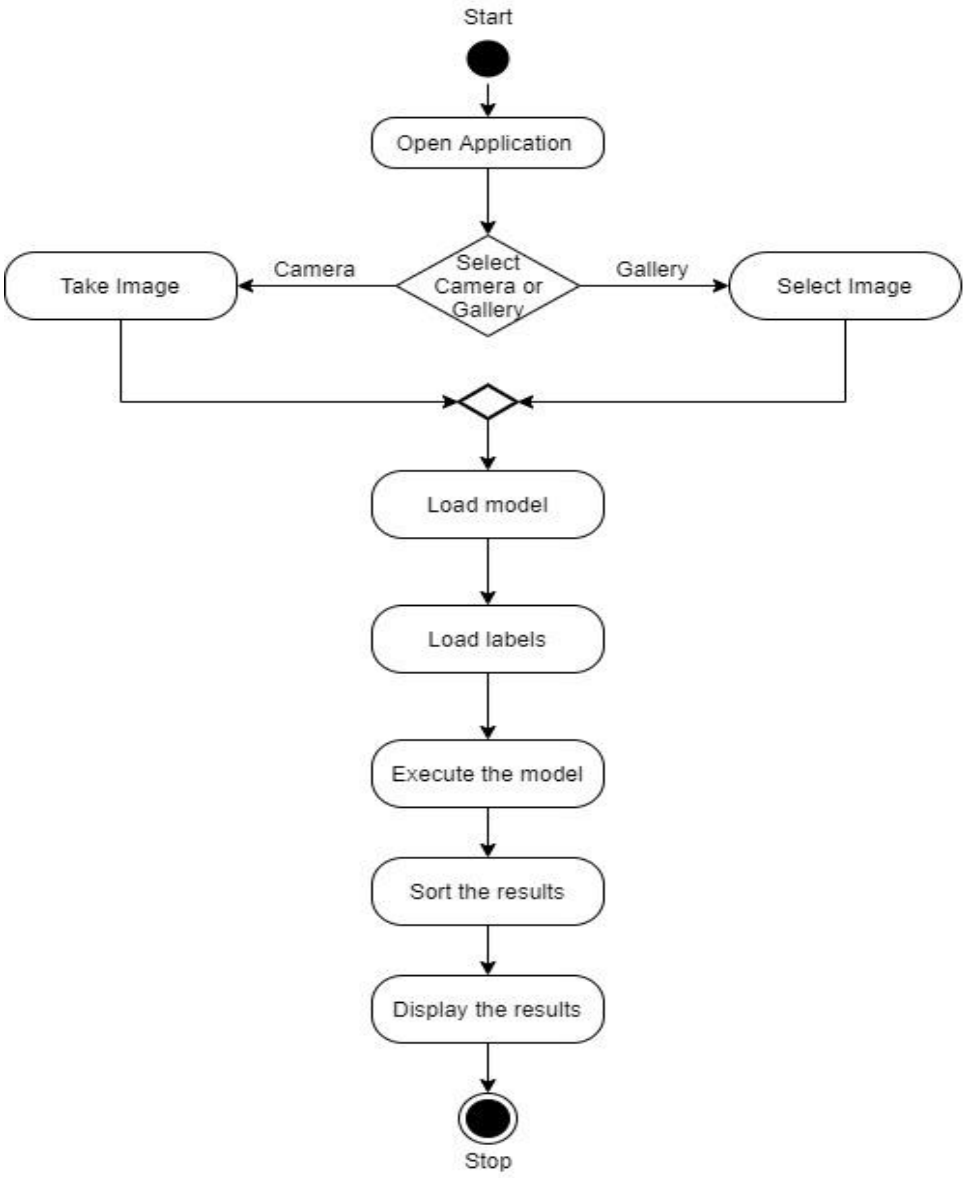
4. COLLABORATION-DIAGRAM:



COLLABORATION DIAGRAM

A collaboration-diagram represents the relationships and interactions among objects. Objects are represented with the help of rectangles with named labels in them. The relationships among different objects are displayed as the connecting lines between the objects. The various messages among various objects can be represented as directed arrows with labels that define the messages.

5. ACTIVITY DIAGRAM:



Activity Diagram

The Activity diagram is a flowchart which represents the flow among various activities. The activity is termed as operation or action that can happen in the system. It illustrates the dynamic view of system. Activity diagrams displays the flow of the control among objects.

5. CODING

MODELLING

```
!pip install -U "tensorflow-gpu==2.0.0rc0"
!pip install -U tensorflow_hub
!pip install -U tensorflow_datasets

im port matplotlib.pylab as plt
im port numpy as np
im port time
im port os
im port tensorflow as tf
im port tensorflow_hub as hub
im port tensorflow_datasets as tfds
tfds.disable_progress_bar()

from tensorflow.keras import layers

#Load data
zip_f=tf.keras.utils.get_file(origin='https://storage.googleapis.com/plantdata/PlantVillage.zip'
,fname='PlantVillage.zip', extract=True)
#Create the training and validation directories
data_directory = os.path.join(os.path.dirname(zip_f), 'PlantVillage')
train_directory = os.path.join(data_directory, 'train')
validation_directory = os.path.join(data_directory, 'validation')

!wget https://github.com/obeshor/Plant-Diseases-Detector/archive/master.zip
!unzip master.zip;
import json
with open('Plant-Diseases-Detector-master/categories.json', 'r') as s:
    category_to_name = json.load(s)
    classes = list(category_to_name.values())
batch_size  = 32 # no of classes
IMG_SHAPE  = 224 # no of im ages
fro m tensorflow.keras. preprocessing.im age im port ImageDataGenerator
```

```
image_generator_train  = Image Data Generator(
                rescale=1./255,
                rotation_range=45,
                width_shift_range=.15,
                height_shift_range=.15,
                horizontal_flip=True,
                zoom_range=0.5
                )

train_data_generator = image_generator_train.flow_from_directory(
                                        batch_size=batch_size,
                                        directory=train_directory,
                                        shuffle=True,
                                        target_size=(IMG_SHAPE,IMG_SHAPE),
                                        class_mode='sparse'
                                        )

Im age_generator_validation = Im age Data Generator ( rescale = 1./255 )

validation_d ata_generator  =  im age_generator_validation.flow_from_dir ectory (batch_size
= batch_s ize,
                                        directory=validation_directory,
                                        target_size=(IMG_SHAPE, IMG_SHAPE),
                                        class_mode='sparse')

URL = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
featuresextractor = hub.Keras Layer ( URL,
                        input_ shape = ( IMG_ SHAPE, IMG_ SHAPE,3 ) )

featuresextractor. train able  =  False

plant_model = tf. ker as .Sequential([
  featuresextractor,
  la yers. Dense(38, activation='softmax')
```

```
])

plant_model.summary()

plant_mo del. compile(
  optim izer= 'ad am',
  loss = tf.losses.SparseCategoricalCrossentropy(),
  met rics = ['accur acy'])
his tory = plant_mod el.fit ( tra in_data_generator,
                epo chs = 3,
                valid ation_data = validation_d ata_generator)

t = time.time()

export_path_sm = "./{}".format(int(t))
print(export_path_sm)
tf.saved_model.save(plant_model, export_path_sm)

reload_sm_keras = tf.keras.models.load_model(
  export_path_sm,
  custom_objects={'KerasLayer': hub.KerasLayer})

reload_sm_keras.summary()

def predict_reload(image):
    probabilities = reload_sm_keras.predict(np.asarray([img]))[0]
    class_idx = np.argmax(probabilities)

    return {classes[class_idx]: probabilities[class_idx]}

import random
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(data_directory, validation_directory, filename))
    img = cv2.resize(img, (IMG_SHAPE, IMG_SHAPE) )
    img = img /255

    return img
for idx, filename in enumerate(random.sample(validation_data_generator.filenames, 2)):
    print("SOURCE: class: %s, file: %s" % (os.path.split(filename)[0], filename))

    img = load_image(filename)
    prediction = predict_reload(img)
    print("PREDICTED: class: %s, confidence: %f" % (list(prediction.keys())[0],
list(prediction.values())[0]))
    plt.imshow(img)
    plt.figure(idx)
    plt.show()
!mkdir "tflite_models"

TFLITE_MODEL = "tflite_models/plant_disease_model.tflite"

con verter  =  tf.lite. TF Lite Converter. from_ saved_ model( export_path_sm )
converted_tfl ite_model = converter. convert ()
open ( TFLITE_MO DEL, "wb" ).write( converted_tf lite_model )
with open('labels.txt', 'w') as f:
  f.write('\n'.join(classes))
from google.colab import files
files.download(TFLITE_MODEL)
files.download('labels.txt')
```

AndroidManifest.xml:

```
<? xml version="1.0" en coding="utf-8"?>
< mani fest  xmlns : and roid="http : //schemas.and roid.com/apk/res/and roid"
    package="com.example.leafdiseasedetection">
    <uses-permission and roid : name="and roid.permission.INTERNET" />
    <uses-permission and roid : name="and
roid.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission and roid : name="and
roid.permission.WRITE_EXTERNAL_STORAGE"/>
.    <app lication
        and roid : allowBackup="true"
        and roid : icon="@mipmap/ic_launcher"
        and roid : label="@string/app_name"
        and roid : roundIcon="@mipmap/ic_launcher_round"
        and roid : supportsRtl="true"
        and roid : theme="@style/AppTheme">
        <activity and roid : name=".MainActivity">
            <intent-filter>
                <action and roid : name="and roid.intent.action.MAIN" />
                <category and roid : name="and roid.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity and roid : name=".DetectionActivity"></activity>
    <provider
        and roid : name="and roidx.core.content.FileProvider"
        and roid : authorities="com.example.and roid.fileprovider"
        and roid : exported="false"
        and roid : grantUriPermissions="true">
        <meta-data
            and roid : name="and roid.support.FILE_PROVIDER_PATHS"
            and roid : resource="@xml/file_paths"></meta-data>
    </pro vider>
    </app lication></man ifest>
```

Classifier.kt

```
package com.example.leafdiseasedetection

im port and roid. con tent. res.AssetManager
im port and roid. graphics.Bitmap
im port and roid .util.Log
im port org .tensor flow.lite.Inter preter
im port java .io.FileInputStream
im port java .nio.ByteBuffer
im port java. nio.ByteOrder
im port java.nio.MappedByteBuffer
im port java. nio.channels.FileChannel
im port java.util.*



class Classifier(assetManager: AssetManager, modelPath: String, labelPath: String,
inputSize: Int) {
    private var INTERPRETER: Interpreter
    private var LABEL_LIST: List<String>
    private val INPUT_SIZE: Int = inputSize
    private val PIXEL_SIZE: Int = 3
    private val IMAGE_MEAN = 0
    private val IMAGE_STD = 255.0f
    private val MAX_RESULTS = 3
    private val THRESHOLD = 0.4f
```

```
    data class Recognition(
        var id: String = "",
        var title: String = "",
        var confidence: Float = 0F
    ) {
        override fun toString(): String {
            return "Title = $title, Confidence = $confidence)"
        } }
    init {
        INTERPRETER = Interpreter(loadModelFile(assetManager, modelPath))
        LABEL_LIST = loadLabelList(assetManager, labelPath)
    }

private fun loadModelFile(assetManager: AssetManager, modelPath: String):
MappedByteBuffer {
        val fileDescriptor = assetManager.openFd(modelPath)
        val inputStream = FileInputStream(fileDescriptor.fileDescriptor)
        val fileChannel = inputStream.channel
        val startOffset = fileDescriptor.startOffset
        val declaredLength = fileDescriptor.declaredLength
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,
declared-Length )
    }

    private fun load Label List (asset Manager : Asset Manager , labelPath : S tring )  : List<
String> {
        return assetManager.open(labelPath).bufferedReader().useLines { it.toList() }


    }

    fun recognizeImage(bitmap: Bitmap): List<Recognition> {
        val scaledBitmap = Bit map.create Scaled Bit map ( bit map , INPUT_ SIZE , INPUT_
SIZE , false )
        val byteBuffer = convertBitmapToByteBuffer(scaledBitmap)
```

```
        val result = Array(1) { FloatArray(LABEL_LIST.size) }
        INTERPRETER.run(byteBuffer, result)
        return getSortedResult(result)
    }


    private fun convert Bitmap To Byte Buffer ( bitmap : Bitmap ) : Byte Buffer {
        val byte Buffer = Byte Buffer .allocate Direct ( 4 * INPUT_SIZE * INPUT_SIZE *
PIXEL_SIZE )
        byte Buffer. order ( Byte Order. Native Order () )
        val int Values  = IntArray(INPUT_SIZE * INPUT_SIZE)
        bit map. Get Pixels ( intValues, 0, bit map. width, 0, 0, bit map. width, bit map. height)
        var pixel = 0
        for ( i in 0 until INPUT_SIZE) {
            for (j in 0 until INPUT_SIZE) {
                val `val` = int Values [ pixel ++ ]
                byte Buffer. Put Float ((((  `val`.shr(16)  and 0xFF ) - IMAGE_ MEAN ) /
IMAGE_ STD ))
                byte Buffer. Put Float ((((  `val`.shr ( 8 ) and 0xFF )  - IMAGE_ MEAN ) /
IMAGE_ STD ))
                byte Buffer. Put Float ((((  `val` and 0xFF ) - IMAGE_ MEAN )  / IMAGE_ STD ))
            }
        }
        return byte Buffer
    }


    private fun get Sorted Result ( label Prob Array : Array<FloatArray> ) : List< Recognition
> {
        Log.d("Classifier", "List Size:(%d, %d,
%d)".format(labelProbArray.size,labelProbArray[0].size,LABEL_LIST.size))
        val pq = PriorityQueue(
            MAX_RESULTS,
```

```
    Comparator<Recognition> {
          (_, _, confidence1), (_, _, confidence2)
        -> java.lang.Float.compare(confidence1, confidence2) * -1
    })
  for (i in LABEL_LIST.indices) {
    val confidence = label Prob Array [ 0 ] [ i ]
    if  ( confidence > = THRES HOLD ) {
      pq. add (
        Recognition ( "" + i ,
        if ( LABEL_LIST.size  > i ) LABEL_LIST[ i ] else "Unknown", confidence )
      ) } }



  Log.d("Classifier", "pqsize:(%d)".format(pq.size))
  val recog = ArrayList<Recognition>()
  val recogSize = Math. min(pq. size, MAX_ RESULTS)
  for (i in 0 until recogSize) {
    recog.add(pq.poll())
  }
  return recog
} }
```

User.kt:

```
pack age com. example. leafdiseasedetection
class User(
   val id: St ring , val name: St ring , val phone: St ring , val disease: St ring , val
date : St ring  )
```

activity_main.xml:

```
<? xml ver sion ="1.0" en coding ="utf-8"?>
<android .constraint layout. wid get. Constraint Layout
 xml ns : and roid="http://schemas.android.com/apk/res/android"
   xml ns : app ="http://schemas.android.com/apk/res-auto"
   xml ns : tools ="http://schemas.android.com/tools"
   android : layout_ width="match_parent"
   android : layout_ height="match_parent"
   tools : con text=".Main Activity ">

   < TextView
      and roid : id="@+id/textView1"
      and roid : layout_ width="67dp"
      and roid : layout_ height="30dp"
      and roid : text="Name:"
      and roid : textAppearance="@style/TextAppearance.AppCompat.Medium"
      app:lay out_ constraint Bottom_ toTopOf="@+id/textView2"
      app:lay out_ constraint Horizontal_ bias="0.093"
      app:lay out_ constraint Left_ toLeftOf="parent"
      app:lay out_ constraint Right_ toRightOf="parent"
      app:lay out_ constraint Top_ toTopOf="parent"
      app:lay out_ constraint Vertical_ bias="0.674" />
   < Text View
      and roid : id="@+id/textView2"
      and roid : layout_ width="58dp"
      and roid : layout_ height="34dp"
      and roid : text="Phone:"
      and roid : text Appearance="@style/TextAppearance.AppCompat.Medium"
      app : layout_ constraint Bottom_ toBottomOf="parent"
      app : layout_ constraint Horizontal_ bias="0.089"
      app : layout_ constraint Left_ toLeftOf="parent"
      app : layout_ constraint Right_ toRightOf="parent"
```

```
    app : layout_ constraint Top_ toTopOf="parent"
    app : layout_ constraint Vertical_ bias="0.344" />

< Edit Text
    and roid : id="@+id/editName"
    and roid : lay out_ width="wrap_content"
    and roid : lay out_ height="wrap_content"
    and roid : ems="15"
    and roid : inputType ="textPersonName"
    and roid :hint="Enter your Name"
    app : lay out_ constraint Bottom_ to BottomOf="parent"
    app : lay out_ constraint End_ to EndOf="parent"
    app : lay out_ constraint Horizontal_ bias="0.191"
    app : lay out_ constraint Start_ to EndOf="@+id/textView1"
    app : lay out_ constraint Top_ to TopOf="parent"
    app : lay out_ constraint Vertical_ bias="0.189" />

< Edit Text
   and roid : id="@+id/editPhone"
    and roid : lay out_ width="wrap_content"
    and roid : lay out_ height="wrap_content"
    and roid : ems ="10"
    and roid : hint = "Enter Phone Number"
    and roid : input Type ="phone"
    app : layout_ constraint Bottom_ to BottomOf="parent"
    app : layout_ constraint End_ to EndOf="parent"
    app : layout_ constraint Horizontal_ bias="0.116"
    app : layout_ constraint Start_ to EndOf="@+id/textView2"
    app : layout_ constraint Top_ to BottomOf="@+id/editName"
    app : layout_ constraint Vertical_ bias="0.115" />

< Button
    and roid : id ="@+id/btnContinue"
    and roid : lay out_ width ="101dp"
```

```
    and roid : layout_ height ="43dp"
    and roid : text = "Continue"
    app : layout_ constraint Bottom_ to BottomOf="parent"
    app : layout_ constraint End_ to EndOf="parent"
    app : layout_ constraint Horizontal_ bias="0.498"
    app : layout_ constraint Start_ to StartOf="parent"
    app : layout_ constraint Top_ to TopOf="parent"
    app : layout_ constraint Vertical_ bias="0.642" />

  < TextView
    and roid : id ="@+id/textView"
    and roid : lay out_ width ="wrap_content"
    and roid : lay out_ height ="wrap_content"
    and roid : text ="Enter your details"
    and roid : text Appearance ="@style/TextAppearance.AppCompat.Large"
    app : lay out_ constraint Bottom_ to BottomOf ="parent"
    app : lay out_ constraint End_ toEndOf ="parent"
    app : lay out_ constraint Start_ toStartOf ="parent"
    app : lay out_ constraint Top_ to TopOf ="parent"
    app : lay out_ constraint Vertical_ bias ="0.087" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.kt

```
Pack age com. example. leafdiseasedetection

im port and roid.annotation.SuppressLint
im port and roid. con tent. Intent
im port and roid. con tent.pm.ActivityInfo
im port and roidx.appcompat.app. App Compat Activity
im port and roid .os. Bundle
im port and roid .widget.Button
im port and roid .wid get.EditText
im port and roid .wid get.Toast

class MainActivity : App Compat Activity () {
    private lateinit var name : Edit Text
   private lateinit var phone : Edit Text
   private lateinit var btn con : Button
     @SuppressLint("SourceLockedOrientationActivity")
     over ride fun onCreate(savedInstanceState: Bundle?) {
      super. On Create( saved Instance State )
        requestedOrientation = Activity Info .SCREEN_ ORIENTATION _PORTRAIT
      set Content View ( R.layout. activity_main)
        name = find View By Id( R. id. editName )
        phone = find View By Id (  R. id.editPhone )
       btncon = find View By Id ( R. id.btnContinue )
      btncon.set On Click Listener {
         saveData()
      } }
   private fun saveData()
   {
     val nametxt=name.text.toString()
      val phonetxt=phone.text.toString()
      if(nametxt.isEmpty()||phonetxt.isEmpty()||phonetxt.length!=10) {
```

```
        Toast.makeText(applicationContext, "Enter valid data",
To ast. LENGTH_LONG). show ()
      return
      }
      val in tent= In tent ( this, DetectionActivity:: class .java)
        in tent. put Extra("Name",nametxt)
        in tent. put Extra("Phone",phonetxt)
        startActivity(intent)
   }
}
```

activity_detection. xml:

```
<? xml-version="1.0" en coding="utf-8"?>
< and roidx. constraint lay out. widget. Constraint Lay out
   xml ns : and roid = "http://schemas.and roid.com/apk/res/and roid"
xml ns : app ="http://schemas.and roid.com/apk/res-auto"
xml ns : tools ="http://schemas.and roid.com/tools"
and roid : lay out_ width="match_parent"
and roid : lay out_ height="match_parent"
tools : context =".DetectionActivity" >

   < Button
      and roid : id ="@+id/mCameraButton"
      and roid : lay out_ width ="wrap_content"
      and roid : lay out_ height ="wrap_content"
      and roid : tex t="@string/buttonTakePhoto"
      app : lay out_ constraintBottom_ toTopOf ="@+id/mPhotoImageView"
      app : lay out_ constraint End_ to EndOf ="parent"
      app : lay out_ constraint Horizontal_ bias ="0.79"
      app : lay out _ constraint Start _ to StartOf = "parent"
      app : lay out _ constraint Top _ to TopOf = "parent"
```

```
   app : lay out _ constraint Vertical _ bias = "0.863" / >
< Button
   and roid : id = "@+id/mGalleryButton"
   and roid : lay out _ width = "wrap _ content"
   and roid : lay out _ height = "wrap _ content"
   and roid : text = "@string/buttonSelectPhoto"
   app : lay out _ constraint Bottom _ toTopOf = "@+id/mPhotoImageView"
   app : lay out _ constraint End _ toStartOf = "@+id/mCameraButton"
   app : lay out _ constraint Horizontal _ bias = "0.418"
   app : lay out _ constraint Start _ toStartOf = "parent"
   app : lay out _ constraint Top _ toTopOf = "parent"
   app : lay out _ constraint Vertical _ bias = "0.863" / >

< Image View
   and roid : id = "@+id/mPhotoImageView"
   and roid : lay out _ width = "350dp"
   and roid : lay out _ height = "400dp"
   and roid : contentDescription = "@string/descriptionImage"
   app : lay out _ constraint Bottom _ toTopOf = "@+id/mResultTextView"
   app : lay out _ constraint End _ toEndOf = "parent"
   app : lay out _ constraint Horizontal _ bias = "0.491"
   app : lay out _ constraint Start _ toStartOf = "parent"
   app : lay out _ constraint Top _ toTopOf = "parent"
   app : lay out _ constraint Vertical _ bias = "0.679"
   app : lay out _ constraint Vertical _ chainStyle = "packed"
   app : srcCompat = "@and roid : color/darker _ gray" / >

< Button
   and roid : id = "@+id/mDetectButton"
   and roid : lay out _ width = "101dp"
   and roid : lay out _ height = "60dp"
   and roid : lay out _ margin Top  =  "12dp"
   and roid : text  =  "@string/buttonDiagnose"
   app : lay out _ constraintBottom _ toTopOf  =  "@+id/mResultTextView"
```

```
    app : lay out _ constraint End _ toStartOf = "@+id/btn _ speak"
    app : lay out _ constraint Horizontal _ bias = "0.604"
    app : lay out _ constraint Start _ toStartOf = "parent"
    app : lay out _ constraint Top _ toBottomOf = "@+id/mPhotoImageView"
    app : lay out _ constraint Vertical _ bias = "1.0" />

<TextView
    and roid : id = "@+id/mResultTextView"
    and roid : lay out _ width = "318dp"
    and roid : lay out _ height = "109dp"
    and roid : text = "@string/defaultImage"
    and roid : text Alignment = "center"
    and roid : textAppearance = "@style/TextAppearance.AppCompat.Medium"
    and roid : textStyle = "bold"
    app : lay out _ constraint Bottom _ to BottomOf = "parent"
    app : lay out _ constraint End _ to EndOf = "parent"
    app : lay out _ constraint Horizontal _ bias = "0.494"
    app : lay out _ constraint Start _ to StartOf = "parent"
    app : lay out _ constraint Top _ to TopOf = "parent"
    app : lay out _ constraint Vertical _ bias = "0.974" />

<Image Button
    and roid : id = "@+id/btn _ speak"
    and roid : lay out _ width = "59dp"
    and roid : lay out _ height = "56dp"
    app : lay out _ constraint Bottom _ to BottomOf = "parent"
    app : lay out _ constraint End _ to EndOf = "parent"
    app : lay out _ constraint Horizontal _ bias = "0.809"
    app : lay out _ constraint Start _ to StartOf = "parent"
    app : lay out _ constraint Top _ to TopOf = "parent"
    app : lay out _ constraint Vertical _ bias = "0.8"
    app : src Compat ="@drawable/speak"
    and roid : contentDescription ="@string/enables_speech"/>
```

</and roidx. constraintlayout. widget. ConstraintLayout>

DetectionActivity.kt

pack age com. example.leafdiseasedetection

im port and roid. annotation.SuppressLint

im port and roid. app.Activity

im port and roid. content.Intent

im port and roid. content. pm. ActivityInfo

im port and roid. graphics. Bit map

im port and roid. graphics. Bit map Factory

im port and roid. graphics. Matrix

im port and roid. net.Uri

im port and roid .os.Build

im port and roid .os.Bundle

im port and roid .os.Environment

im port and roid .provider.MediaStore

im port and roid .speech.tts.TextToSpeech

im port and roid .view.Gravity

im port and roid .widget.ImageButton

im port and roid .widget.ImageView

im port and roid .widget.Toast

im port and roidx .annotation.RequiresApi

im port and roidx .appcompat.app.AppCompatActivity

im port and roidx. core.content.FileProvider

im port com. google. firebase .database.Firebase Database

im port com. google. firebase. storage. Firebase Storage

im port com. google. firebase. storage. Storage Reference

im port kotlinx. and roid. synthetic. main.activity_detection.*

im port java.io. File

im port java.io. IOException

im port java.text. Simple Date Format

im port java. util.*

```
@Suppress("DEPRECATION")

class DetectionActivity : App Compat Activity () {

    private late in it var mClassifier: Classifier
    private late in it var mBitmap: Bitmap
    private late in it var mTTS: TextToSpeech
    private late in it var btnSpeak: ImageButton
    private late in it var txtSpeak: String
    private late in it var photo: ImageView
    private late in it var photoURI:Uri
    private late in it var id:String
    private late in it var disease:String
    private var ch=0
    private late in it var currentPhotoPath: String
    private late in it var mStorageRef:StorageReference
    private val mCameraRequestCode = 0
    private val mGalleryRequestCode = 2
    private val mInputSize = 224
    private val mModelPath = "plant_disease_model.tflite"
    private val mLabelPath = "plant_labels.txt"
    private val mSamplePath = "soybean.JPG"
    private var nametxt:String="default"
    private var phonetxt:String=""

    @SuppressLint("SourceLockedOrientationActivity", "SimpleDateFormat", "SetTextI18n")
    @Requires Api( Build. VERSION_ CODES.O)
```

```
override fun onCreate(savedInstanceState: Bundle?)
{
    super. on Create (saved Instance State)
    requestedOrientation  = Activity Info .SCREEN_ ORIENTATION_PORTRAIT
    set Content View ( R. lay out. activity_detection)
    nametxt= intent.getStringExtra("Name")
    phonetxt=intent.getStringExtra("Phone")
    this.btnSpeak =findViewById(R.id.btn_speak)
    btnSpeak.isEnabled=false
   photo=findViewById(R.id.mPhotoImageView)
    mStorageRef = FirebaseStorage.getInstance().reference
    mTTS= TextToSpeech(this,TextToSpeech.OnInitListener { mTTS.language =
Locale.US })
    mClassifier = Classifier(assets, mModelPath, mLabelPath, mInputSize)
    resources.assets.open(mSamplePath).use {
      mBitmap = BitmapFactory.decodeStream(it)
      mBitmap = Bitmap.createScaledBitmap(mBitmap, mInputSize, mInputSize, true)
      mPhotoImageView.setImageBitmap(mBitmap)
    }

    mCameraButton.setOnClickListener {
     dispatchTakePictureIntent()
    }

    mGalleryButton.setOnClickListener {
      val callGalleryIntent = Intent(Intent.ACTION_PICK)
      callGalleryIntent.type = "image/*"
      startActivityForResult(callGalleryIntent, mGalleryRequestCode)
    }
```

```
    mDetectButton.setOnClickListener {
        val results = mClassifier.recognizeImage(mBitmap).firstOrNull()
        val c = Calendar.getInstance()
        val dateFormat = SimpleDateFormat("dd-MM-yyyy h:mm:ss a")
        val date = dateFormat.format(c.time)
        val displayText = "Dear User, $nametxt\n The Result is \n"
        mResultTextView.text =
            displayText + results?.title + "\n Confidence:" + results?.confidence
        txtSpeak = displayText + results?.title
        btnSpeak.isEnabled = true
        mDetectButton.isEnabled = false

        if (ch != 0) {
            val dref = FirebaseDatabase.getInstance().getReference("User")
            id = dref.push().key.toString()
            disease=results?.title.toString()
                val user = User(id, nametxt, phonetxt, disease, date)
                dref.child(id).setValue(user)
            uploader()
        }
    }

    btnSpeak.setOnClickListener {
        val pitch = 1.0f
        val speed = 0.8f
        mTTS.setPitch(pitch)
        mTTS.setSpeechRate(speed)
        mTTS.speak(txtSpeak, TextToSpeech.QUEUE_FLUSH, null)
    }
}
```

```
    private fun uploader() {
        val ref = mStorageRef.child("Images/").child("$id   $disease")
        ref.putFile(photoURI)
    }

    @SuppressLint("SimpleDateFormat")
    @Throws(IOException::class)

    private fun createImageFile(): File {
        val time stamp : String= Simple Date Format ("yyyyMMdd_HHmmss"). format (Date())
        val storage Dir : File = get External Files Dir ( Environment. DIRECTORY_
PICTURES)
        return File. Create Temp File (
            "JPEG_${timeStamp}_",
            " .jpg ",
            storage Dir
        ) . apply {
                currentPhotoPath = absolutePath
        }
    }

    private fun dis patch Take Picture Intent () {
        Intent ( Media Store. ACTION_ IMAGE_ CAPTURE).also { take Picture Intent  ->
            take Picture Intent. resolve Activity  ( package Manager ) ? .also {
                val photoFile: File=
                    createImageFile()
                photoFile. also {
                    photo URI =  File Provider. get Uri For File (
                        this ," com. example. android. file provider ", it
                    )
                 take Picture Intent. put Extra ( Media Store .EXTRA_ OUTPUT , photo URI)
                    start Activity For Result ( take Picture Intent, mCameraRequestCode)
                } } } }
```

```
   @SuppressLint("SetTextI18n")
   over ride fun on Activity Result ( request Code : Int, result Code : Int, data : Intent ? ) {
      super.on Activity Result ( request Code , result Code , data)
      if ( request Code == mCameraRequestCode){
          if(resultCode == Activity. RESULT_ OK) {
            ch=1
               mBitmap = MediaStore.Images.Media.getBitmap(this.contentResolver,
photoURI)
            mBitmap = scaleImage(mBitmap)
            val toast = Toast.makeText(this, ("Image crop to: w= ${mBitmap.width} h=
${mBitmap.height}"), Toast. LENGTH_ LONG )
            to ast. Set Gravity ( Gravity. BOTTOM, 0, 20)
            to ast . show ()
            mPhotoImageView.setImageBitmap(mBitmap)
            mResultTextView.text= "Your photo image set now."
            mDetectButton.isEnabled=true
          } else {
            Toast.makeText(this, "Camera cancel..", To ast. LENGTH_ LONG ). show()
          }
      } else if (requestCode == mGalleryRequestCode) {
          if ( data ! = null ) {
             photoURI= data .data
            ch=1
            try {
               mBitmap = Media Store. Images. Media. get Bit map ( this.contentResolver,
photoURI)
            } catch ( e : IO Exception ) {
               e. print Stack Trace ()
            }
            println("Success!!!")
            mBitmap = scaleImage(mBitmap)
            mPhotoImageView.setImageBitmap(mBitmap)
```

```
            mDetectButton.isEnabled=true
          } }
        else {
          To ast. make Text ( this, "Unrecognized request code", To ast. LENGTH_ LONG ).
show ()
      }
  }

  private fun scaleImage(bit map: Bit map?): Bitmap {
    val orignalWidth = bit map !!. width
    val originalHeight = bit map. height
    val scaleWidth = mInputSize.toFloat() / orignalWidth
    val scaleHeight = mInputSize.toFloat() / originalHeight
    val mat = Matrix ()
    mat .postScale(scaleWidth, scaleHeight)
    return Bit map .create Bit map (bitmap, 0 , 0 , orignalWidth, originalHeight, mat, true)
  }}
```

6. OUTPUT SCREENS

Fig. 6.1. User interface

Upon opening the app, the above interface appears. The user has to provide a name and phone number in the specified fields. The user then clicks on the continue button to use the app.

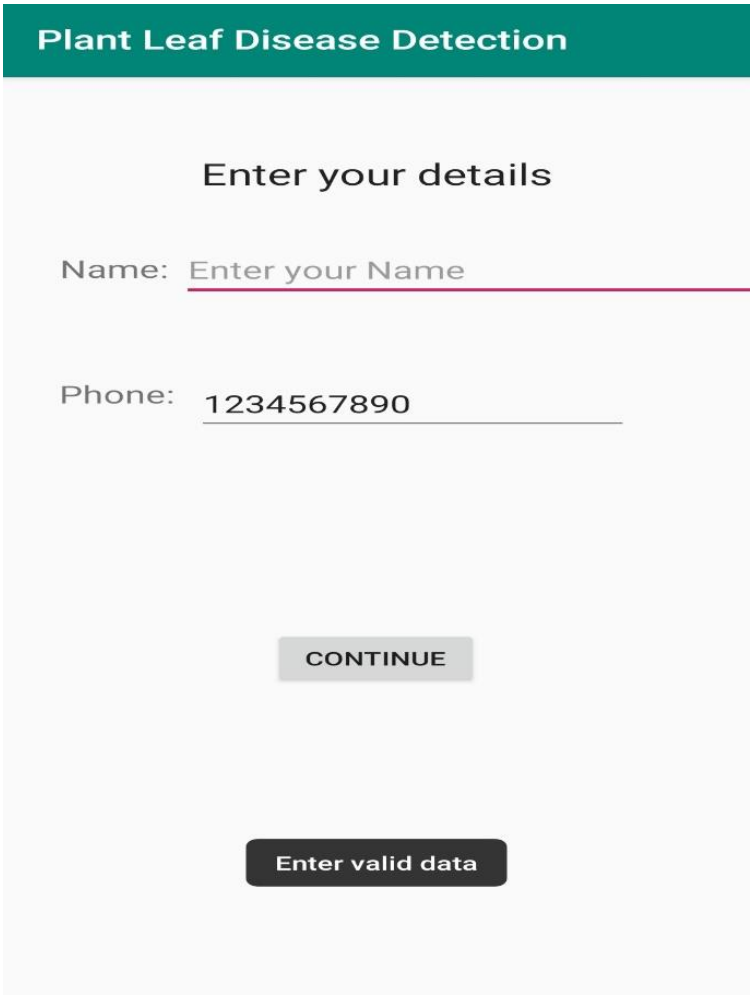


Fig. 6.2. Invalid data

The user has to provide a name and phone number. If either of them is invalid, the user will be prompted to give valid data. Given data is invalid if either of the fields is empty or if the phone number isn't of 10 digits.

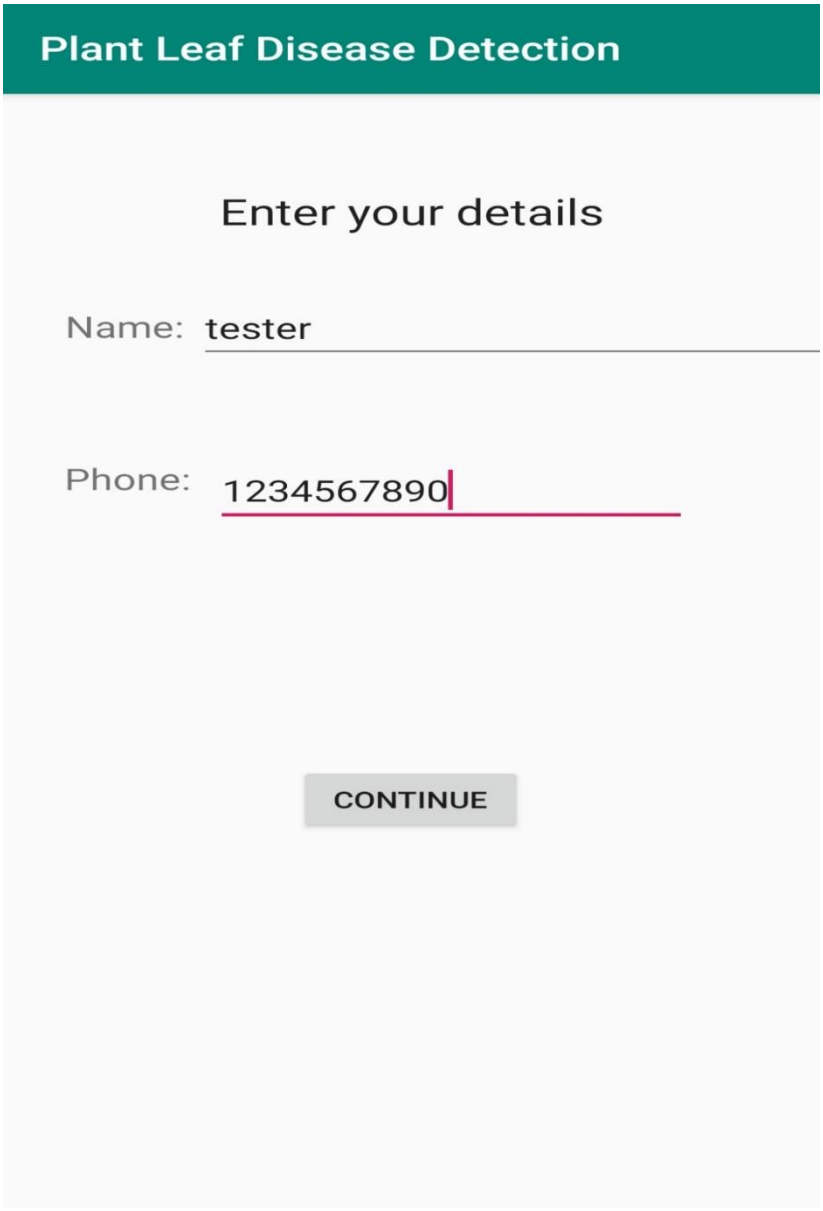


Fig. 6.3. Valid user data

The user enters data, then clicks the continue button. Upon validating the data, the user can continue using the app.

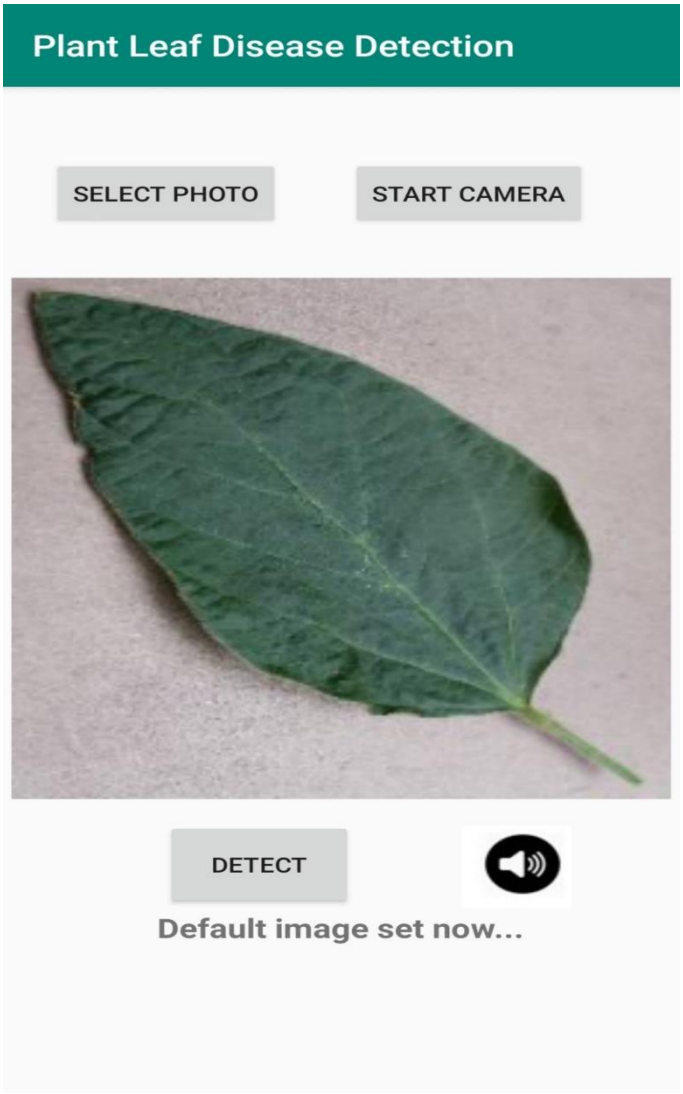


Fig. 6.4. Disease detection interface

The Above interface provides options for the user such as

- Using images from gallery
- Using images from camera

The interface also has a button which calls the detection algorithm.

A button for the text to speech conversion is also available.

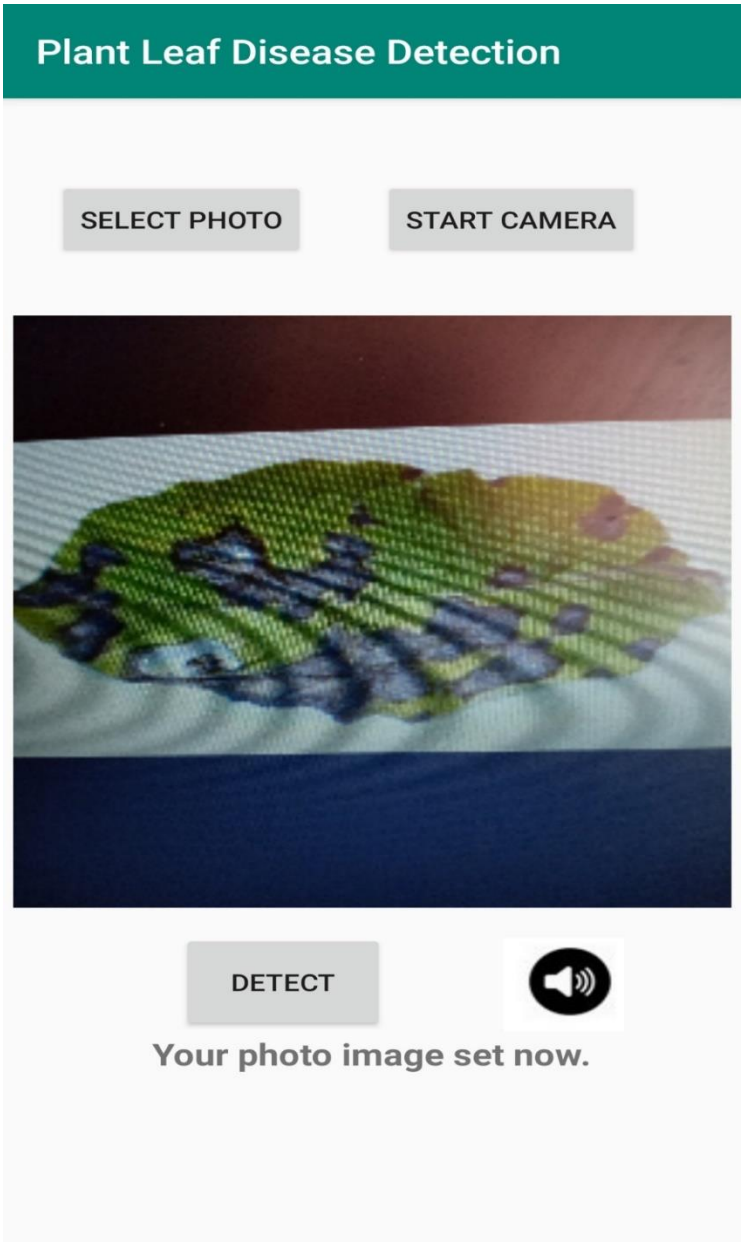


Fig. 6.5. Upload an image

The user can either use gallery or camera to upload the image to the app.

The image is resized and then is passed to the image viewer which displays the image on the interface.

The User can now use the app to detect the disease by clicking on the detect button.

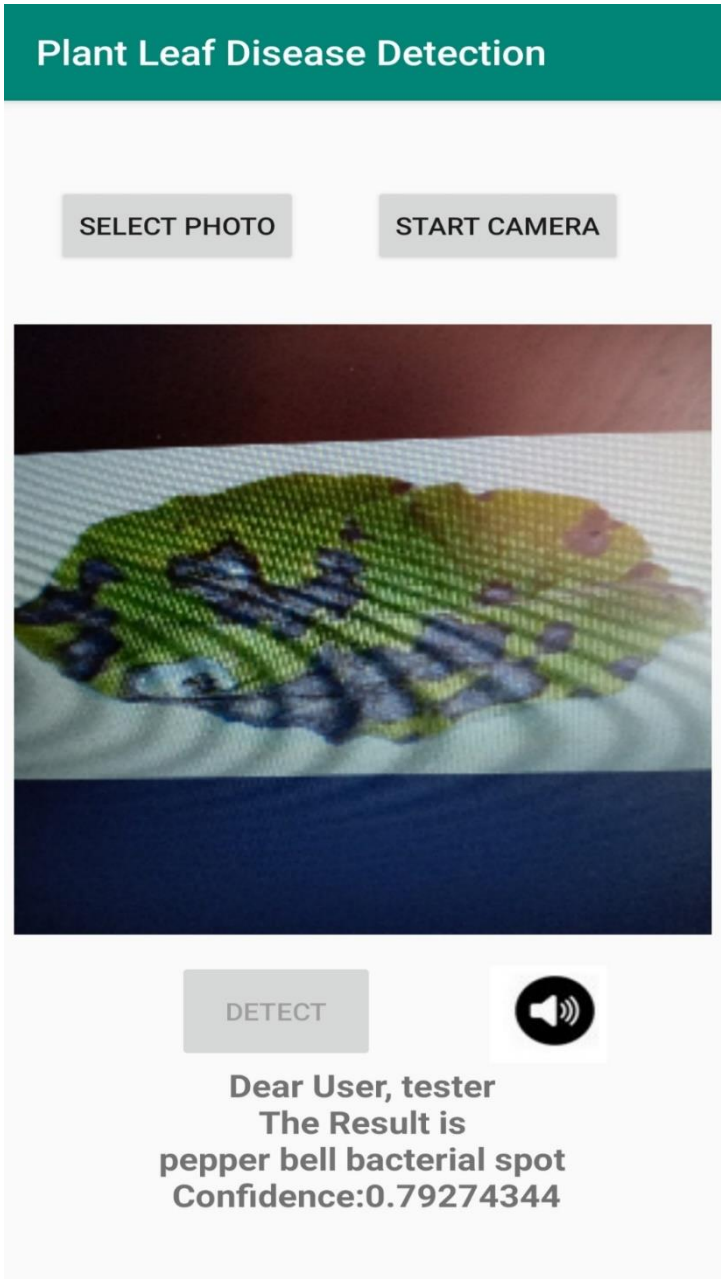


Fig. 6.6. Detecting the disease

On clicking the detect button, the result is displayed at the button of the interface which includes

- The result of detection.
- The confidence of detection.

The User can click on the speak button to have an audio output of the result.

The uploaded image, the result and the user data are uploaded to the database.

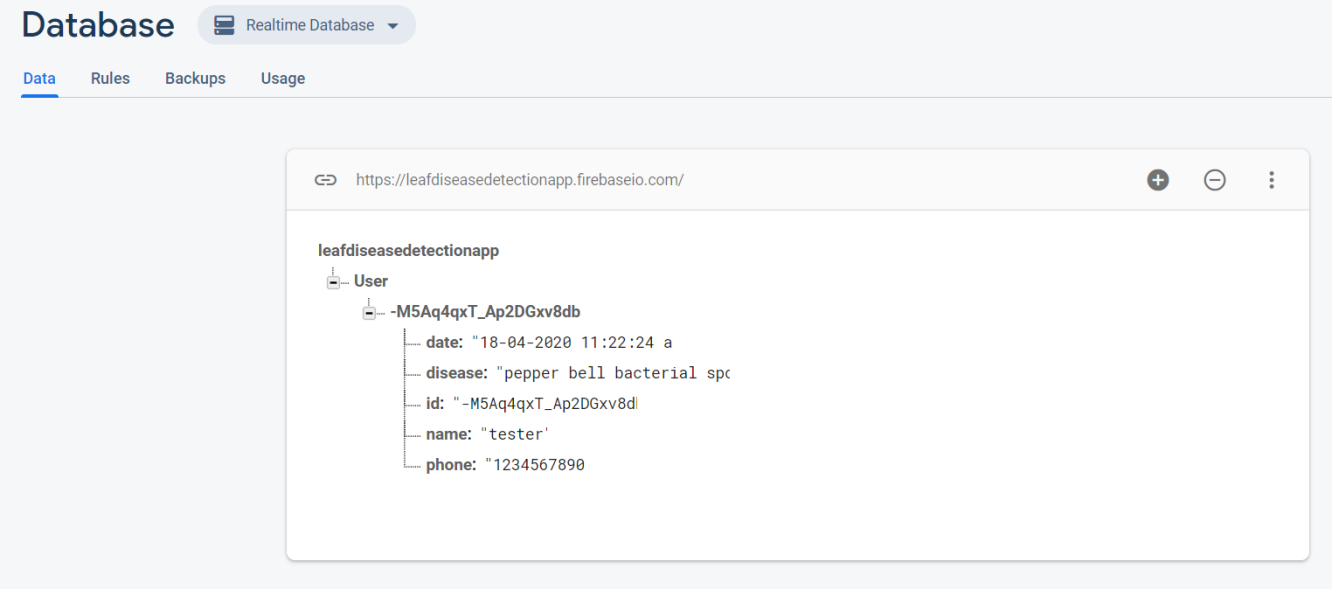


Fig. 6.7. Firebase database

The User data and detected result are stored in the database for future references. These references include the very important information such as:

- Date of usage
- Disease detected
- Id of the user
- Name of the user
- Phone number of the user

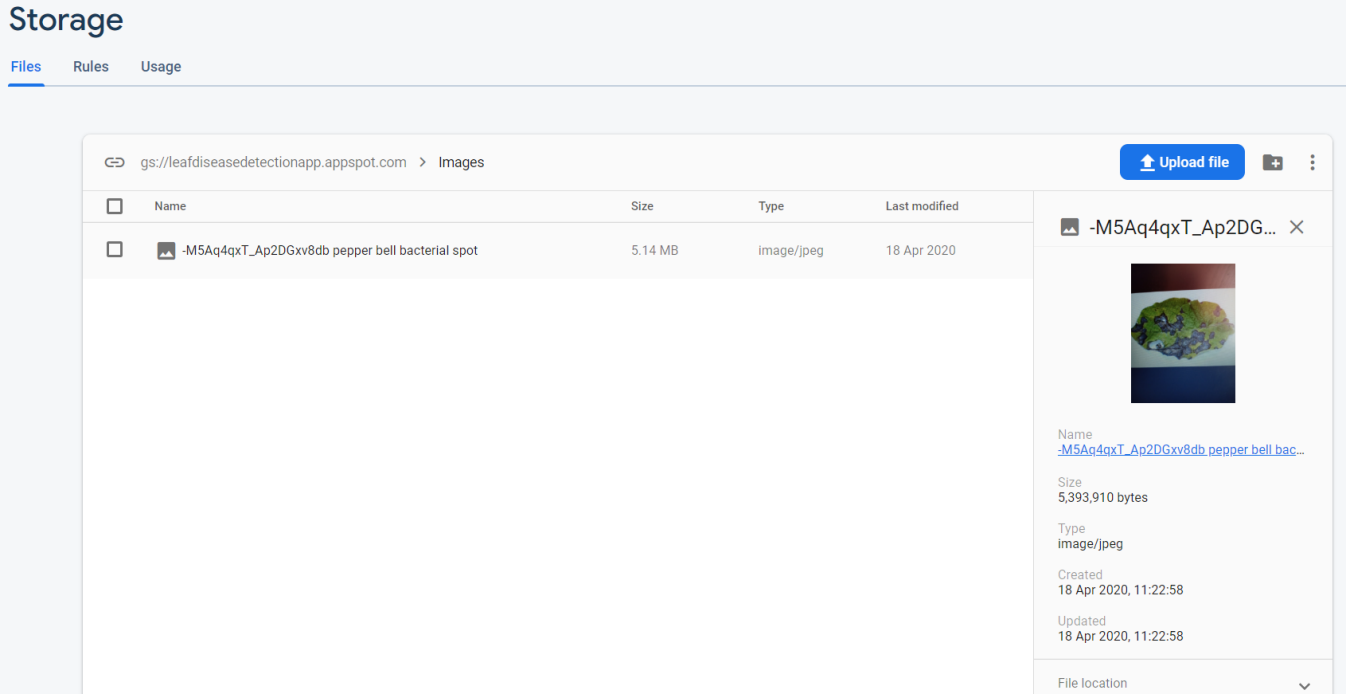


Fig. 6.8. Firebase storage

The uploaded image is stored in the Firebase storage which can be used for future improvements which contains important information regarding the image such as:

- Size of the image
- Type of the image
- Date, time of creation
- Date, time of updation (if any)

7. SYSTEM-TESTING

The main purpose of any testing is finding errors. It is the procedure of finding all possible error in the product or system. This offers to test the functioning of different assemblies, sub-assemblies, components, and/or a completed product. The procedure of executing software with the motive in making sure that the system meets the user needs. There are different kinds of test for the system. Each test kind focuses on a specific kind of requirement.

7.1. TYPES OF TESTS:

1. Unit testing:

This includes designing of test cases which can verify that the program logic is able to function correctly, and the inputs must produce necessary outputs. The decision branches and the code flow must be verified. This is needed for testing of each software unit. This is to be carried at the end of each unit by unit before integrating them. It depends on knowledge about internal working. It does performs the tests at each basic level. This finds that each possible combination of process must perform accurately as per specifications.

2. Integration testing:

These tests were made to check various software integrated components that explain if they run as one complete unit. This is an event specific and is much more related with the outputs. Integration tests explains that even though all units have individual satisfaction, the combination of units must also be efficient. This testing is aimed to expose the defects that form by combining the units.

3. Functional test:

These tests must give implementations that are needed to be test the system documentation, technical requirements, business and the user manuals.

This testing depends on :

- Outputs : The classes of possible outcomes should be generated .
- Functions : necessary functions should be implemented.
- Invalid Inputs : The classes of invalid data should be rejected.
- Valid Inputs : The classes of valid data should be accepted.

Preparation of these tests depends on key functions, requirements, and test cases. Prior to the testing is finished, other necessary tests are to be identified, total value of tests is described.

4. System Test:

This testing helps to ensure that entire system is according to the specified requirements provided. This tests against the various test cases to get the results. System testing is tests on the various configurations. It is also based on various processes, flows and descriptions, focusing on integration points and the results.

5. White Box Testing:

It is the testing in which the tester has some prior information on the inner logic, structure and language, and its purpose. This is useful in testing areas that aren't normally tested by black box testing.

6. Black Box Testing:

This is the testing without any prior information about the inner structure or workings or language under testing. It can also be written from a document, like that of requirements or specification document. In this, the software being tested is like that of a unknown black box .

7.2. Unit Testing:

It is generally performed with the help of code and in testing phase of the software's life cycle, but it is common for unit testing and coding to be performed as different stages.

- The testing of fields is to be carried and the functional tests are to be written clearly.
- Testing the strategy and their approaches.

Objectives:

• User gets the specified output

• The screens, along with the messages and their responses should not be overlooked.

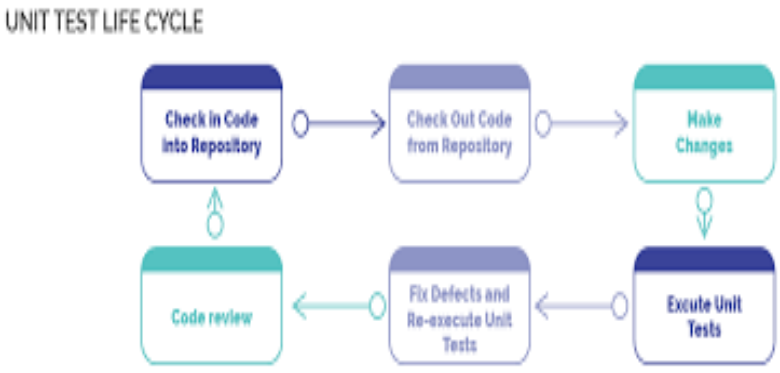
• All entries should function properly

The Features which should be validated are:

• Working with the database

• To check that all the data available are according to requirements and of the correct format

• The transformation and outputs should be produced.



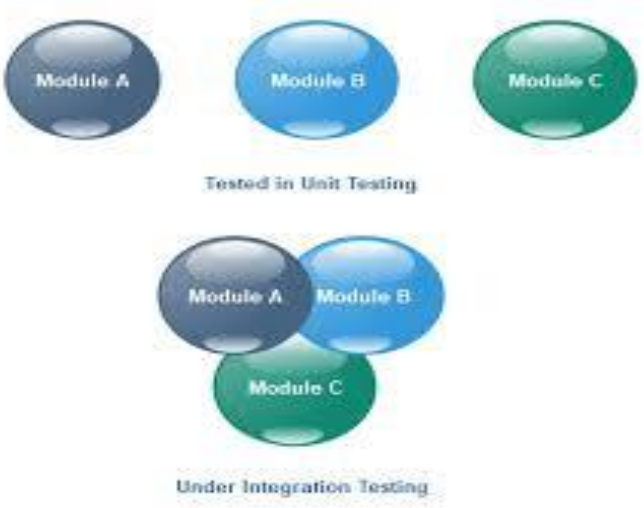
7.3. Integration Testing:

In this testing, the units/modules are combined for integration testing. The sole aim of this testing is to check the reliability, performance, and functionality among the different integrated modules.

Integration Strategies are as below:

- Big - Bang Integration
- Bottom Up Integration
- Hybrid Integration
- Top Down Integration

This testing is the testing of multiple software components which are integrated to detect errors that are caused by the various interface problems. The aim of this testing is to verify that all the components or software applications function properly.



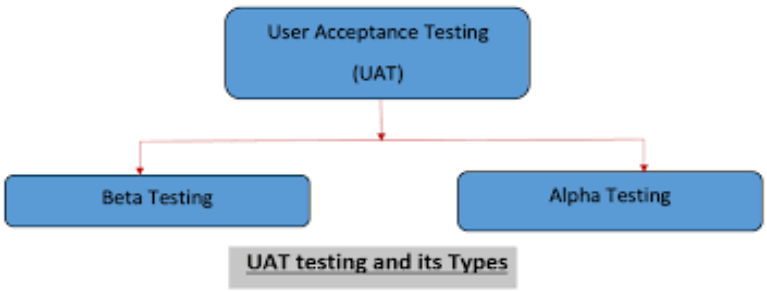
7.4. Acceptance Testing

It is an important stage for any project and needs the use by the user. It also verifies that the system should meet the various functional requirements provided. This testing is done to find that the software system has satisfied the requirements. The aim of this testing is to determine the accordance with the requirements and to find if it has satisfied the conditions for delivering to users at the end.

The various kinds of acceptance testing are:

- Alpha Testing,
- Beta Testing,
- Business acceptance Testing and
- User acceptance Testing

This testing is carried out in various phases. Firstly, the tests are implemented, then if all the results are appropriate enough then more complex phases are executed.



UAT testing and its Types

7.5 Test Cases

S No	Test case ID	Input	Output (Expected)	Output (Actual)	Test Case Pass or fail
1	LDDAT1	Enter invalid name and phone number, then click the continue button.	Prompt user to enter valid data	Enter valid data	Pass
2	LDDAT2	Enter valid data, then click the continue button.	Display detection interface.	Detection interface	Pass
3	LDDAT3	Select image from gallery.	Display image in the view.	Displays image.	Pass
4	LDDAT4	Select image using camera.	Display image in the view.	Displays image.	Pass
5	LDDAT5	Click the detect button	The result is displayed.	Displays result.	Pass
6	LDDAT6	Click the speak button	The result is spoken out	Audio output.	Pass
7	LDDAT7	Click the detect button	The user data is uploaded to database	User data is uploaded.	Pass
8	LDDAT8	Click the detect button	The image is uploaded to storage.	Image is uploaded.	Pass

8. CONCLUSION

Plant diseases are to be dealt quickly and efficiently. In order to do so, detection of disease plays a very important role. This project can efficiently analyze and give proper result to the user. This helps to take necessary actions in time. The interface is easy to understand and use. The Disease is detected based on pre-trained model which can be improved on later stages. The CNN model is efficient to detect the disease. The User has the ease of choosing the image from gallery or take a photo using the camera. The displayed result can be spoken out in the app using the specified button in it. This improves the experience of the project. The data is then stored in the Firebase database and can be accessed for analysis and can help to form statistical data.

9. FUTURE ENHANCEMENT

- We intend to improve the model by increasing the dataset used in the modelling. The Project as of now can detected only few of the diseases.
- The User Interface can be made more appealing and easier to use.
- The User and result data in the database can be used to get statistics regarding the diseases.
- The Project has only audio output in English. This can be extended to use other languages.

10. BIBLIOGRAPHY

https://www.frontiersin.org/articles/10.3389/fpls.2019.00941/full

https://developer.android.com/docs

https://firebase.google.com/docs/database/android/start

https://colab.research.google.com

https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4

https://www.tutorialspoint.com/android

APPENDIX - A

ANDROID STUDIO

Android is Open source operating system. It is developed for devices such as smartphones and tablet.

Android was initiated by the Open Handset Alliance. It is based on Linux.

The Android studio with the Kotlin programming language is very effective in many ways when compared to that of android with Java programming language.

1. **Less code is combined with greater readability**. Allows spending less time writing your code and to work on understanding the code.

2. **Mature language and improved environment**. Kotlin has developed continuously, both not only as a language but also as a whole ecosystem. It is seamlessly integrated in Android Studio and can be actively used for developing Android applications.

3. **Kotlin support in Android Jetpack and other libraries**. KTX extensions add various Kotlin language features, like coroutines, extension functions and named parameters, to the available Android libraries.

4. **Interoperability with Java**. You can use both Kotlin and the Java programming language in applications without needing to change all the code to Kotlin.

5. **Support for multiplatform development**. Ability to use Kotlin for developing not only Android but for iOS, backend, and web applications.

6. **Code safety**. Less code and more readability leads to less errors. The Kotlin compiler can detect these remaining errors, making the code safe.

7. **Easy learning**. Kotlin is easy to learn, especially to Java developers.

8. **Big community**. Kotlin has support and contributions from the community, which is growing.

.

DEEP LEARNING:

Deep-learning is one among various classes of machine-learning algorithms that uses multi layers to get greater level insights from the provided data. For instance, in image-processing the lower level layers can detect and identify the edges, where as the higher layers helps to detect the data relevant to a human which includes letters or digits or faces etc. Most modern deep learning models use Convolutional Neural-Networks (CNN)s.

In deep-learning, every level extracts and transform the input into a more brief and suitable representation. For an image-recognition application, the input can be matrix of pixels and the representational layer can include abstraction of pixels, detect and encode the edges. The second layer can compose and encode the edges. The next third layer is able to encode the nose and the eyes. The fourth layer can also detect the images containing a face. A Deep-learning process is able to learn to work on which features to be placed optimally on which level by deciding of its own knowledge.

The Deep-Learning consists of large of layers from which data can be transformed. These systems does contain a suitable credit-assignment-path (CAP) depth. Deep learning architectures can also be constructed with the help of a greedy methodology of layer-by-layer method. It helps in differentiating these abstracts and to pick the necessary features to improve the efficiency. In the supervised learning, the methods of deep learning eliminate further steps, by transforming the data into brief abstract forms along with components, and can help to achieve staggered structures that can eliminate the redundancy while representation.

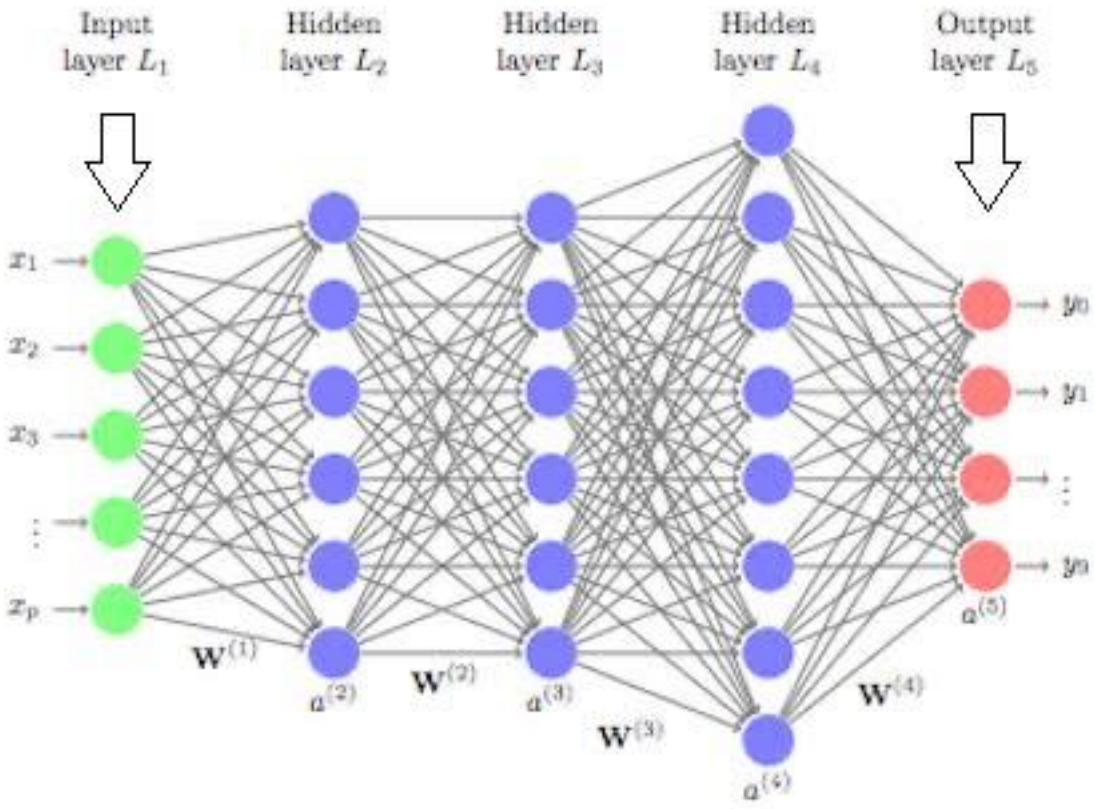


Fig. 3.2.2 Deep Learning Structure

CONVOLUTIONAL NEURAL NETWORKS (CNN)s

The Convolutional Neural-Network consists of the network which can use a mathematical operation known as Convolution. The Convolution is the special form for linear operation. Convolutional networks can be termed as simply neural-networks that can use convolution in place of generally used matrix-multiplication in minimum of one of it's layers. In deep-learning, a Convolutional Neural-Network (CNN) is one among the classes of deep neural-network. They are mostly used for analysis for the visual imagery. They can also be called as space-invariant artificial neural-networks which works with the shared weights architecture. They are also used in language processing, recommender systems, medical image analysis, image classification, image and video recognition and image classification.

The Convolutional Neural-Network contains the input layer, output layer, and other multi-hidden layers. The hidden-layers of the CNN usually include a set of convolutional-layers that are developed with the multiplication/other dot product. It can be represented as either cross-correlation or sliding dot product . This has much importance for the value of indices in matrix, and has influence on how weight can be specified at specific index points.

When programming the CNN, the input data is in the form of tensor along with its shape that has (image depth) x (number of images) x (image height) x (image width). After passing through the convolutional layer, the image transforms from abstracted to a feature map along with shape that has (feature map width) x (number of images) x (feature map channels) x (feature map height). A convolutional layer must have the following attributes:

- Convolutional kernels with height and width.
- The depth of Convolution filter should be equal to the number channels of the input feature map.
- The number of input and output channels.

Convolutional layers should transform the input and transfer the result to the subsequent layer. It is like the reaction of the neuron for a stimulus in the visual cortex of brain. Each one of the convolutional neuron must process data. Even if it can be used to distinguish and learn differences and classify the information, thus, it can be impractical to apply it to images. The convolution operation can solve the above problem as it can reduce the number of parameters, making the network to deepen along with other parameters. Irrespective of the image size, region of tiling of size 5 x 5, all of them with the equal shared-weights, only needs about 25

trainable parameters. This can resolve the exploding gradients or vanishing problem of training multi-layer networks with the help of backpropagation.

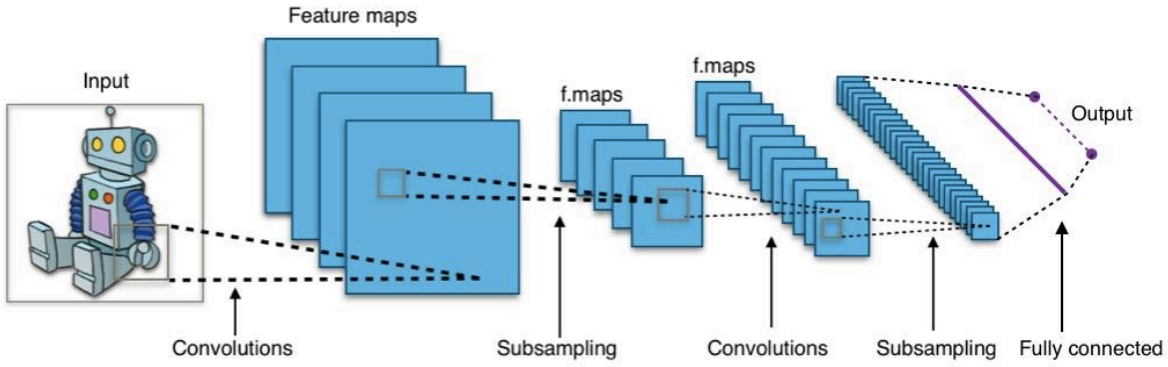


Fig. 3.2.3 CNN Model

GOOGLE COLAB

Colaboratory, or “Colab” is one of the products from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is well suited for data analysis, machine learning and education. Colab is a hosted Jupyter notebook service that doesn’t require any setup to use, but can provide free access to computing resources including GPUs. It is free to use.

Colab allows you to use and share Jupyter notebooks with others without having to download, install, edit or run anything. Colab notebooks allow you to combine multiple executable code and text into a single document, along with images, HTML and more.

When you create your own Colab notebooks, they are stored in your Google Drive account. With Colab you can harness the full power of popular Python libraries to analyse and visualize data. The code cell below uses NumPy to generate some random data and uses matplotlib to visualize it.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

FIREBASE DATABASE:

The Firebase-Realtime Database allows building of functional applications by making secure connection to the database through client-side application. The Information can be preserved locally, to provide the user a more responsive and effective experience. If the device reconnects, the Realtime-Database can sync with the local data for the changes that occurred.

The Realtime-Database has expression-based, flexible rules language, known as Firebase Realtime-Database Security Rules, which declare on how the information must dependent and when records may be written/read. If the project is included with Firebase-Authentication, we can also outline who can have ability to access to what kind of data, and how can they use it.

The Realtime Database is No-SQL based database and provides various optimality and functional features in comparison to the relational-database. The Realtime-Database API is made to best accept operations that need to be done quickly. Thus, It allows to make a real-time experience that provides the user reducing the lacking of responsiveness. Hence, it is also essential for considering on how the user can have access to the stored data.

APPENDIX – B

UML

The UML is the most commonly used language for the making of software blue-prints. The UML stands for Unified Modelling Language. It is the language used for

- Specifying
- Visualizing
- Documenting
- Constructing

It contains vocabulary and related guidelines for composing parts for communication. Thus, Modelling language is the one whose guidelines and vocabulary focuses both on conceptual as well as physical representation of the system.

4.4.1Building Blocks of UML:

The UML mainly comprises of three types of building blocks

- Relationships,
- Things and
- Diagrams

Relationships tend to keep the things together.
Things can be termed as the abstractions which are high-quality components of the model.
Diagrams have to group things into collections.

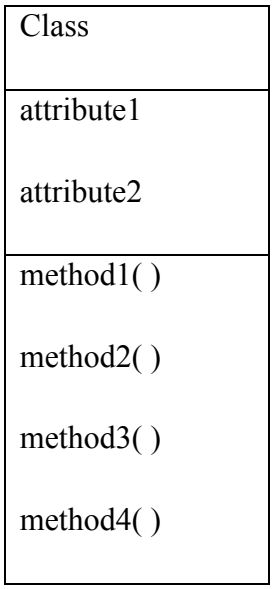
Things in UML:

The various types of things present in UML are:

- Annotational things,
- Grouping things,
- Behavioral things and
- Structural things

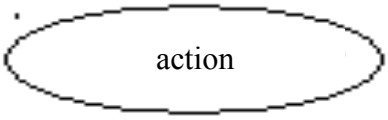
Structural things:

A **Class** is the blueprint of objects that have the same relationships, operations, attributes and semantics.



Class

A U**se-Case** is the sequence of events/actions which happen in the system which give result to actor.

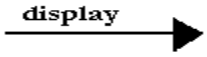


Use Case

Behavioral things:

Interaction:

An interaction comprises of messages transferred among the objects to accomplish several activities. An interaction includes several elements such as action sequences, messages.



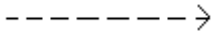
interaction

Relationships in UML:

The different types relationships available in UML are:

- Generalization
- Association
- Dependency
- Realization

Dependency is the relation among two things in the model and a change to any one of thing may according affect another thing.



dependency

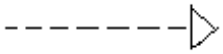
An **Association** is the relationship that describes links where a line is being a link among different objects. Aggregation is type of association, which represents a relationship among the whole and the sub-parts of it.

association

Generalization is the relation where the objects in one of the defined element are exchanged for the objects belonging to the general element.

generalization

Realization is the relation among classifiers, in which a classifier defines the contract and other classifier must guarantee to accept contract.



realization