

# INFORMATICS INSTITUTE OF TECHNOLOGY

## DEPARTMENT OF COMPUTING

Module: 5COSC009C.2

Software Development Group Project

Module Leader: Mr. Guhanathan Poravi

### **Paddy Weed Detector**

Team Hypesters		
Name	lit ID	UoW ID
Akila Nanayakkara	2018400	w1742308
Kisal Wedage	2018368	w1742101
Moveen Wijerathne	2018541	w1742328
Amilakelum Kodikara	2018395	w1742107
Danushka Samarakoon	2018382	w1742106
Malshama Perera	2018446	w1742313

## Table of Contents

List of Tables .....	4
Table of figures .....	4
1. Implementation .....	5
1.1. Chapter Overview .....	5
1.2. Overview of the Prototype .....	5
1.3. Selection of Technologies .....	5
1.3.1. Programming language for data science part.....	5
1.3.2. Image Processing Library .....	5
1.3.3. Web Frameworks Selection .....	6
1.4. Selection of tools.....	6
1.4.1. Selection of IDE .....	6
1.4.2. Selection of the version control system.....	7
1.5. Technology Stack .....	7
1.6. Data Set .....	8
1.7. Implementation of the features of the prototype in the back end .....	8
1.7.1. Data set Pre-Processing .....	8
1.7.2. Training the model .....	10
1.7.3. Selected Algorithm.....	11
1.7.4. Testing the model .....	11
1.8. Implementation of the GUI.....	13
1.8.1. Implementation of the input and the output page .....	13
1.9. Back-end and its Combination .....	19
1.10. Chapter Summary .....	21
2. Testing.....	22
2.1. Chapter Overview .....	22
2.2. Testing Goals and objectives.....	22
2.3. Testing Criteria .....	22
2.4. Testing Functional Requirements .....	22
2.5. Testing Non-Functional Requirements. ....	23
2.6. Performance Testing.....	23
2.7. Accuracy Testing .....	24
2.8. Usability Testing.....	24
2.9. Compatibility Testing .....	24

2.10.	Chapter Summary .....	25
3.	Evaluation .....	26
3.1.	Chapter Overview .....	26
3.2.	Evaluation Goals.....	26
3.3.	Evaluations Benchmark.....	26
3.4.	Evaluations Selections.....	26
3.5.	Concept and Scope .....	26
3.6.	Technical Aspects.....	26
3.7.	Impacts.....	26
3.8.	Limitations.....	26
3.9.	Research question.....	26
3.10.	Self-Evaluation .....	27
3.11.	Chapter Summary .....	27
4.	Conclusion.....	28
4.1.	Chapter Overview .....	28
4.2.	Achievement of the Project Aim.....	28
4.3.	Achievements of Operational Objectives .....	28
4.4.	Achievements of Academic Objectives.....	29
4.5.	Achievement of Requirements .....	30
4.5.1.	Achievement of Functional Requirements .....	30
4.5.2.	Achievement of Non-Functional Requirements .....	31
4.6.	Milestones Deliverables.....	31
4.7.	Problems and Challenges Faced .....	32
4.8.	Learning Outcomes .....	32
4.9.	Future Enhancements.....	33
4.10.	Chapter Summary .....	33
	References .....	34

## List of Tables

Table 1 Testing the functional requirements.....	23
Table 2 Performance Testing .....	23
Table 3 Compatibility testing .....	24
Table 4 Achievements of operational objectives.....	29
Table 5 Achievements of academic objectives .....	30
Table 6 Achievement of Functional Requirements.....	30
Table 7 Achievement of Non-Functional Requirements.....	31
Table 8 Milestones Deliverables .....	31

## Table of figures

Figure 1 Technology Stack .....	7
Figure 2 File path to the data set .....	8
Figure 3 Data Set gray scaling and resizing .....	8
Figure 4 Imports for pre-processing .....	8
Figure 5 Extracting the weights to two pickles .....	9
Figure 6 Preview of gray sailing .....	9
Figure 7 Imports for training the model .....	10
Figure 8 Calling our pre created pickles to train the model .....	10
Figure 9 Model training and saving the model .....	10
Figure 10 CNC Model Explanation (Saha, 2018) .....	11
Figure 11 Model testing in the console .....	11
Figure 12 Weed Image .....	12
Figure 13 Console Testing Output.....	12
Figure 14 Image detection vs image classification .....	13
Figure 15 First Page.....	13
Figure 16 User sign in.....	14
Figure 17 User Log in.....	14
Figure 18 Image Classifier Access .....	15
Figure 19 Image classifier input .....	15
Figure 20 System weed detecting.....	16
Figure 21 System paddy detecting.....	16
Figure 22 System showing the details and the location .....	17
Figure 23 Location where the image was taken .....	17
Figure 24 Input page 2 Angular.....	18
Figure 25 Input page Components.....	18
Figure 26 Code to get the image.....	19
Figure 27 Code to show the added image .....	19
Figure 28 Database of our system .....	19
Figure 29 Server – express .....	20
Figure 30 Database Connection .....	20
Figure 31 Loading the model to backend .....	20
Figure 32 Backend image pre processing .....	21
Figure 33 Accuracy testing using tensor board.....	24
Figure 34 Accuracy in PyCharm model training .....	24

## 1. Implementation

### 1.1. Chapter Overview

This chapter elaborates about the implementation and elaborates about the selection of technologies and the reason behind it. How the design discussions transformed to code, libraries, datasets used are documented. Languages, IDEs and the version control system are also described. Important code snippets are shown and explained. Then the implementation of components is discussed. Finally, screenshots of the system are given after which, the chapter concludes.

### 1.2. Overview of the Prototype

Over idea was to classify the weed “*echinocloa crus-galli*” in Sri Lanka commonly known as “maha maruk” from the basic paddy, the reason for that is that weed is little similar to the normal weed. We created an image classification system to do that. When we insert an image of a paddy or a weed, our classification model will classify whether it is “maha maruk” weed or a normal paddy. Also, when we insert an image which was taken from a camera where the location function is available. We can extract the location from the image and show it from the maps.

### 1.3. Selection of Technologies

#### 1.3.1. Programming language for data science part

When considering image processing systems python runs a major task in the current world among the competitors like Java, C, C++ etc. So in this case no point of deciding which language is to be used. Python takes a major part because it is an appropriate and timely choice for image processing. This is because of the availability of many states of Image processing tools in its ecosystem. Tools such as Numpy, OpenCV play a major role.

Machine learning	: CNN (Convolutional Neural Network), TensorFlow,
Keras Data analysis and visualization	: NumPy, SciPy

- Backed up with extremely good documentation and community support
- Easier to build models (CNN ) compared to other
- Existing literacy with the members, are some reasons to select python as the programming language.

#### 1.3.2. Image Processing Library

In python there are so many libraries: pillow, SciPy, OpenCV etc. after doing a big study it was decided to use OpenCV. OpenCV-Python is a Python wrapper for the \_OpenCV python implementation. All the OpenCV array structures are converted to Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib. Which we didn't use.

TensorFlow is used as the framework to build and train the module. It provides an excellent service rather than other frameworks. TensorFlow has a high-level API name Keras, which is used by the selected module (CNN).

For image processing there are so many libraries. For this project we have use OpenCV, TensorFlow, Keras, NumPy and Matplotlib as the libraries for the data science part.

- OpenCV is a free open source library that is used in image processing. We have used the OpenCV library for image preprocessing. We should preprocess the image before using it for the model training processes. We have used the OpenCV for resize the image and turn the color of the image to gray.
- Keras is an open source neural network library written in python. It is capable of running on top of the TensorFlow. It wraps the efficient numerical computation library and allows you to define and train neural network models in just a few lines of code
- TensorFlow is an open source artificial intelligence library, using data flow graphs to build models. It allows developers to create large-scale neural networks with many layers. TensorFlow is mainly used for: Classification, Perception, Understanding, Discovering, Prediction and Creation.
- NumPy is a library for python programming language. By this library we can convert the digital image to a dimensional array. This dimensional array become the source to the processing part.
- Matplotlib is a plotting library for the python programming language and its numerical mathematics extension NumPy.

### 1.3.3. Web Frameworks Selection

Since our proposed solution contains a web application which interact with the user and the backend which deploy the model and handle the application logic, in our case it is pre-processing the input image. Web framework will handle the interaction between frontend and backed other than implement everything from scratch. For python there are two web frameworks to consider, Flask and Django. We used Django for our development and its Rest Api to pass the images from front end to back end.

## 1.4. Selection of tools

### 1.4.1. Selection of IDE

JetBrains PyCharm, sublime text, Programiz most known IDEs for Python development. JetBrains PyCharm was decided to be used as the IDE, because of following reasons

- It has provided us with a powerful solution for a data science program in python language
- Existing experience of the members in the team

Also, for the front-end angular development, we used the Visual Studio Code as code editor.

### 1.4.2. Selection of the version control system

Git was decided to be used as the Version Control System,

- Changes can be committed offline, pushed when it is convenient, and the history can be reverted easily.
- Existing experience of the author.

Above reasons was responsible to use Git

### 1.5. Technology Stack

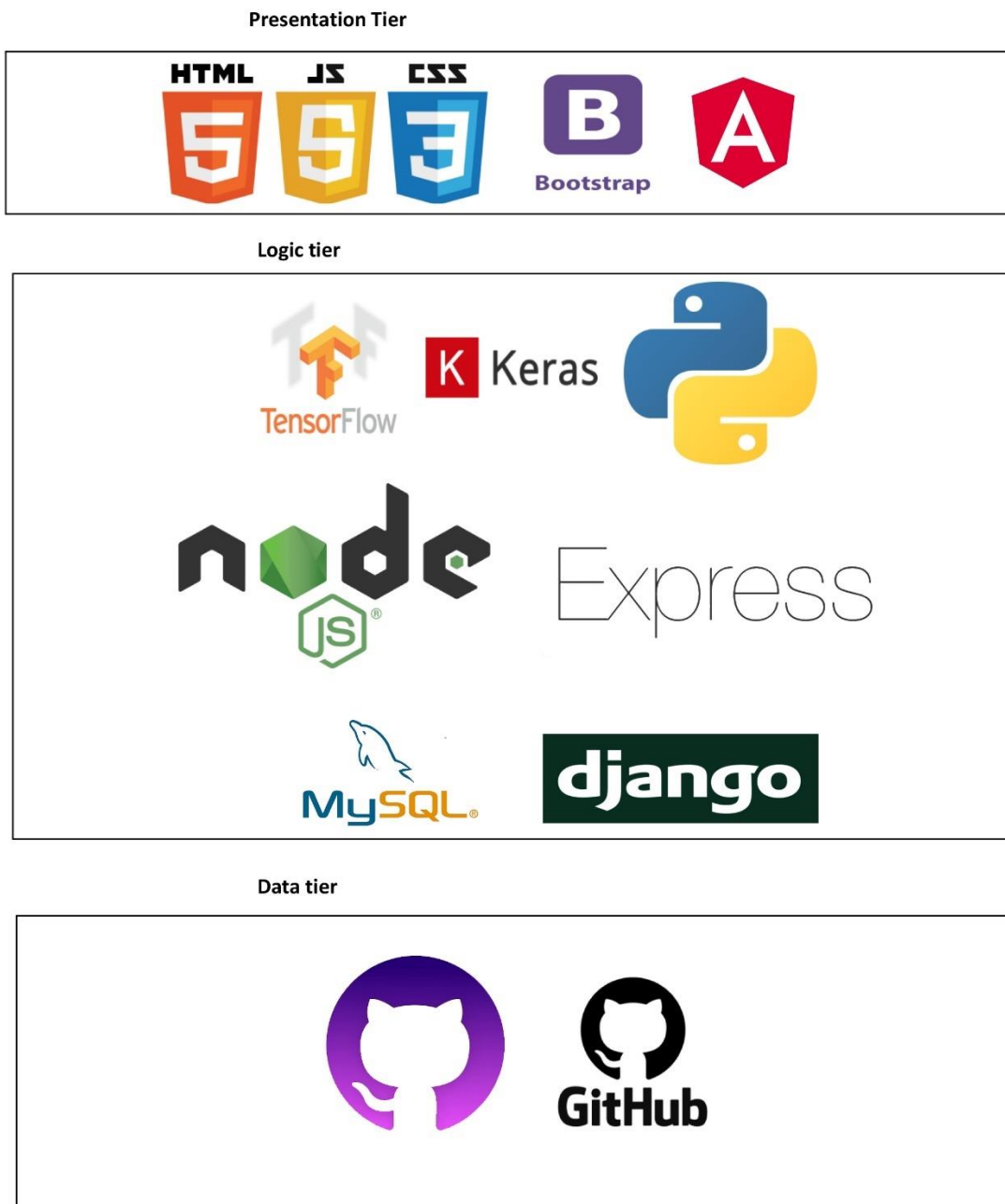


Figure 1 Technology Stack

## 1.6. Data Set

Any image processing project starts with images. Images with low quality will result in the program to malfunction and not being able to develop. Thus, the very first question was if there are any suitable datasets available for the system to be built.

Since we are specializing in weed, we had a hard time gathering this dataset. Image set with above thousand images and all with quality images wasn't easy. We got an image set from <https://www.kaggle.com/>. The data set was made by combining five to six dataset images. Some camera images were taken to test and train the module.

## 1.7. Implementation of the features of the prototype in the back end

### 1.7.1. Data set Pre-Processing

When we get an image data set, before training the model we should preprocess those images. The reason is there can be various types of images and various size of images. So, before we train them into a model, we should make them all similar to a particular size and color and then we should extract the data from those images.

```
DATA_DIRECTORY = "D:\\iit\\year-2\\SDGP\\weed-detector\\paddyWeedDetector\\Data_set\\data\\train" #path where the data set is located
CATEGORIES = ["paddy", "weed"]
```

Figure 2 File path to the data set

```
def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATA_DIRECTORY, category) # create path to paddy and weed
        class_num = CATEGORIES.index(category) # get the classification (0 or a 1). 0=paddy 1=weed

        for image in tqdm(os.listdir(path)): # iterate over each image per paddy and weed
            try:
                image_array = cv2.imread(os.path.join(path, image), cv2.IMREAD_GRAYSCALE) # convert to array
                image_array_2 = cv2.resize(image_array, (IMAGE_SIZE, IMAGE_SIZE)) # resize to normalize data size
                training_data.append([image_array_2, class_num]) # add this to training_data
            except Exception as e: #to keep the output clean...
                pass
```

Figure 3 Data Set gray scaling and resizing

In the above code, we take the image data set and then we categorize them as paddy and weed and then we put them into an array. After that we convert those images to gray scale for the easiness of the image data extraction and then we resize all those images to 50\*50.

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from tqdm import tqdm
import os
```

Figure 4 Imports for pre-processing



```

create_training_data()

print(len(training_data_array))

import random

#to balance the training data input to the model
random.shuffle(training_data_array) #this will shuffle the data and input to the model

for sample in training_data_array[:10]: #checking if the shuffling is working
    print(sample[1])

#making the modle (pickles)
X = []
y = []

for features, label in training_data_array: #packeting the shuffled data to arrays
    X.append(features)
    y.append(label)

print(X[0].reshape(-1, IMAGE_SIZE, IMAGE_SIZE, 1))

X = np.array(X).reshape(-1, IMAGE_SIZE, IMAGE_SIZE, 1)

y = np.array(y)

import pickle #saving the processed data inside a pickle

pickle_out = open("X.pickle", "wb")
pickle.dump(X, pickle_out)
pickle_out.close()

pickle_out = open("y.pickle", "wb")
pickle.dump(y, pickle_out)
pickle_out.close()

```

Figure 5 Extracting the weights to two pickles

In the above code first, we randomize the images and then we reshape our images and then we create two arrays and then we insert our weights of each image to the particular array according to the category. After that we save those two arrays as two pickles to be used in our model training.

```

for category in CATEGORIES:
    path = os.path.join(DATA_DIRECTORY, category) # create path to paddy and weed
    for img in os.listdir(path): # iterate over each image per paddy and weed
        gray_img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE) # convert to array and gray scaling
        plt.imshow(gray_img_array, cmap='gray') # graph it
        plt.show() # display
        break # to display one picture in gray scale
    break

print(gray_img_array)

print(gray_img_array.shape)

```

Figure 6 Preview of gray sailing

### 1.7.2. Training the model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import pickle
import time
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
```

Figure 7 Imports for training the model

```
#importing the pickles we create
pickle_in = open("X.pickle", "rb")
X = pickle.load(pickle_in)

pickle_in = open("y.pickle", "rb")
y = pickle.load(pickle_in)

X = X/255.0 #normalizing the data by scaling, min is 0 and max is 255

dense_layers = [2]
layer_sizes = [64, 128]
conv_layers = [3]
```

Figure 8 Calling our pre created pickles to train the model

In this code we import the pickles we previously saved and then we normalize the them. After that we create our required layers to train the model.

```
for dense_layer in dense_layers:
    for layer_size in layer_sizes:
        for conv_layer in conv_layers:
            NAME = "paddyweedCNN-{}-conv-{}-nodes-{}-dense-{}".format(conv_layer, layer_size, dense_layer, int(time.time()))
            tensorboard = TensorBoard(log_dir="logs\{}".format(NAME))
            print(NAME)

            # training the model
            model = Sequential()

            model.add(Conv2D(layer_size, (3, 3), input_shape=X.shape[1:])) # Convolutional layer
            model.add(Activation('relu')) # activation layer
            model.add(MaxPooling2D(pool_size=(2, 2))) # max pooling layer

            for i in range(conv_layer - 1):
                model.add(Conv2D(layer_size, (3, 3)))
                model.add(Activation('relu'))
                model.add(MaxPooling2D(pool_size=(2, 2)))

            model.add(Flatten()) # this converts our 3D feature maps to 1D feature vectors

            # model.add(Dense(64))
            # model.add(Activation('relu'))

            for i in range(dense_layer):
                model.add(Dense(512)) # dense layer
                model.add(Activation('relu'))
                model.add(Dropout(0.2))

            model.add(Dense(1))
            model.add(Activation('sigmoid'))

            model.compile(loss='binary_crossentropy',
                          optimizer='adam',
                          metrics=['accuracy'])
```

Figure 9 Model training and saving the model

Here according to our layers, we create our model and then save it as in .h5 format.

### 1.7.3. Selected Algorithm

We used CNN algorithm to train our model.

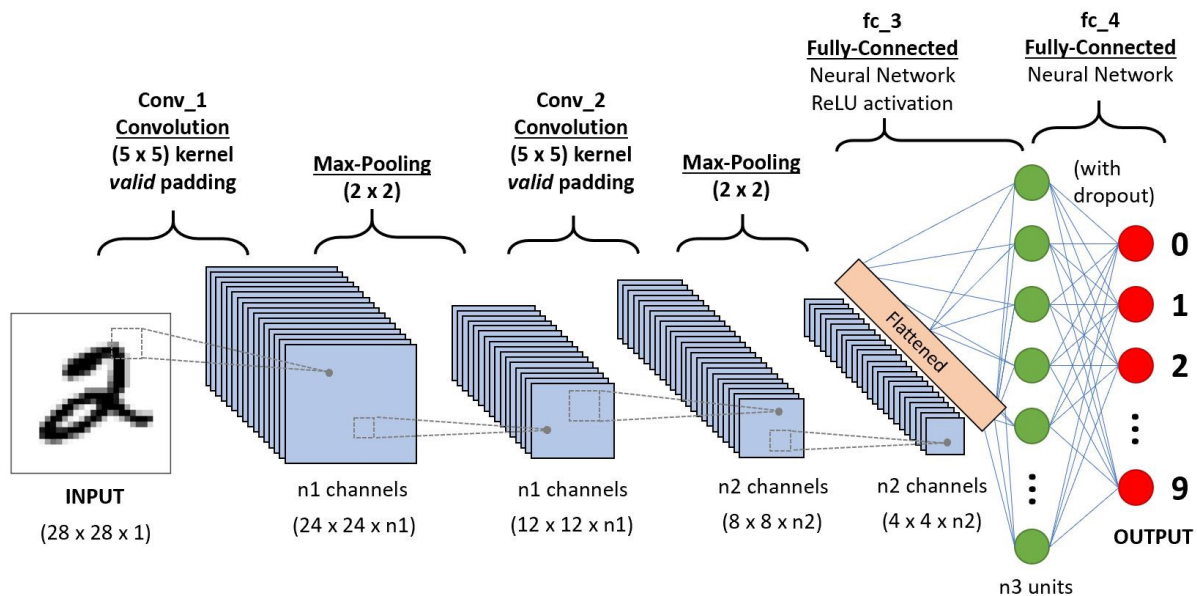


Figure 10 CNC Model Explanation (Saha, 2018)

A Convolutional Neural Network (**CNN**) is a Deep Learning **algorithm** which can take in an input image, assign learnable weights and biases to various objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. (Saha, 2018)

### 1.7.4. Testing the model

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import cv2
import tensorflow as tf

CATEGORIES = ["paddy", "weed"] # will use this to convert prediction num to string value

def prepare(filepath):
    IMAGE_SIZE = 50 # 50 in txt-based
    image_grayscale_array = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE) # read in the image, convert to grayscale
    image_resize_array = cv2.resize(image_grayscale_array, (IMAGE_SIZE, IMAGE_SIZE)) # resize image to match model's expected sizing
    return image_resize_array.reshape(-1, IMAGE_SIZE, IMAGE_SIZE, 1) # return the image with shaping that TF wants.

model = tf.keras.models.load_model("paddyWeedDetectorModelWithArch.h5")

prediction = model.predict([prepare('African-Lovegrass-BuckleyEcologyLab.jpeg')]) # PASSING A LIST OF THINGS YOU WISH TO PREDICT
print(CATEGORIES[int(prediction[0][0])])
```

Figure 11 Model testing in the console

In the above snippet we created a console app to test our classification model. In the first place we get the image that we want to classify. Then we pre-processed the image to insert to the model. Then we load the model to our console and then we feed pre-processed image

to the model. Finally, the model will give an output saying if the input image is a weed or a paddy.



Figure 12 Weed Image

```
"C:\Users\All Users\Anaconda3\envs\paddyWeedDetector\python.exe" D:/iit/year-2/SDGP/weed-detector/paddyWeedDetector/Final-Data-Science-Model/paddyWeedClassificationModel-modelTesting.py
WARNING:tensorflow:From C:\Users\All Users\Anaconda3\envs\paddyWeedDetector\lib\site-packages\tensorflow_core\python\ops\init_ops.py:97: calling GlorotUniform.__init__ (from tensorflow.py
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\Users\All Users\Anaconda3\envs\paddyWeedDetector\lib\site-packages\tensorflow_core\python\ops\init_ops.py:97: calling Zeros.__init__ (from tensorflow.python.ops
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:From C:\Users\All Users\Anaconda3\envs\paddyWeedDetector\lib\site-packages\tensorflow_core\python\ops\resource_variable_ops.py:1630: calling BaseResourceVariable.__init
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From C:\Users\All Users\Anaconda3\envs\paddyWeedDetector\lib\site-packages\tensorflow_core\python\ops\nn_impl.py:183: where (from tensorflow.python.ops.array_ops) is de
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
weed
Process finished with exit code 0
```

Figure 13 Console Testing Output

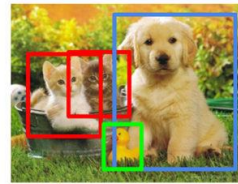


But there is exception for our model. In the first place our model is not an image detection model. It's an image classification model. It means we train our model to specific sets of images. It will get their weights and train. So, if we input a non-related image still our image classification will tell it's a paddy or a weed. The reason is some of the weight files can be there in the non-related image too. There is a work around for this. First, we should make an image detection model and then pass the detected image to the classifier. We didn't go through that because our prototype was about the image classifier.



**Classification**

CAT

**Object Detection**

CAT, DOG, DUCK

*Figure 14 Image detection vs image classification*

## 1.8. Implementation of the GUI

The Angular framework was used to build the web application. Angular modules and Bootstrap stylings are included to come up with a high responsive beautiful web application. This web application contains one component called “First-page”. All the relevant sections are included in this component.

### 1.8.1. Implementation of the input and the output page

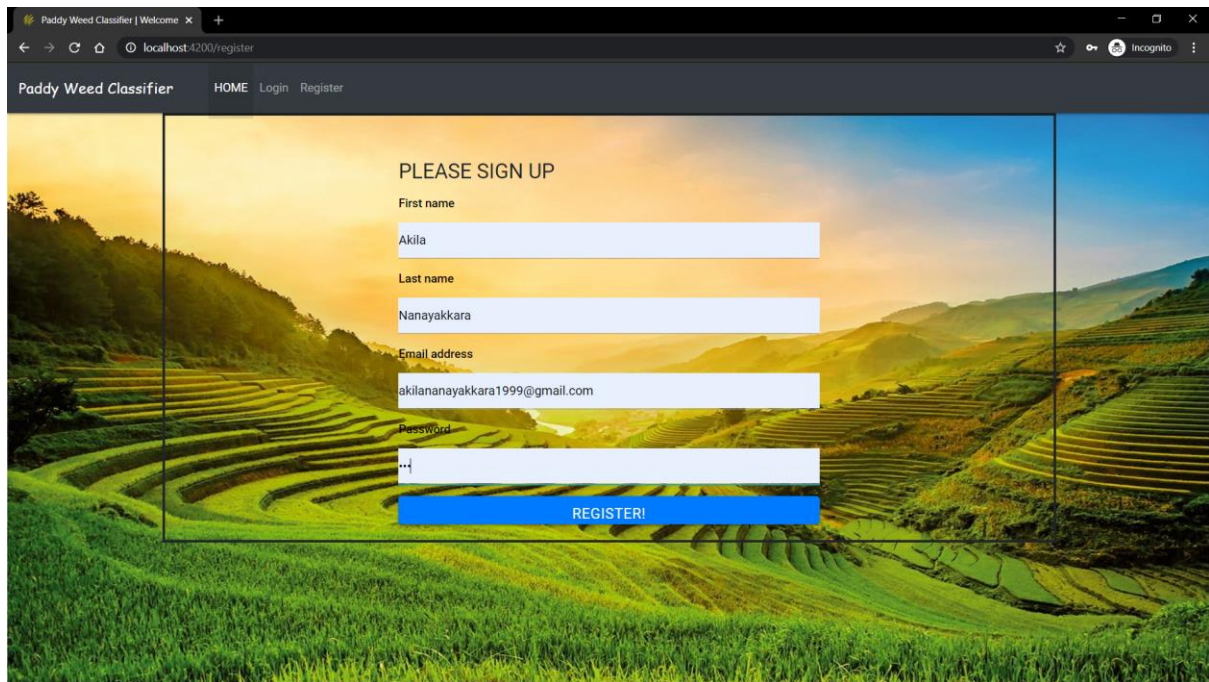
These figures declare the view of the GUI which was implemented for the Paddy Weed Classifier.

When the user first come to our WebApp first user sees the bellow webpage.

*Figure 15 First Page*

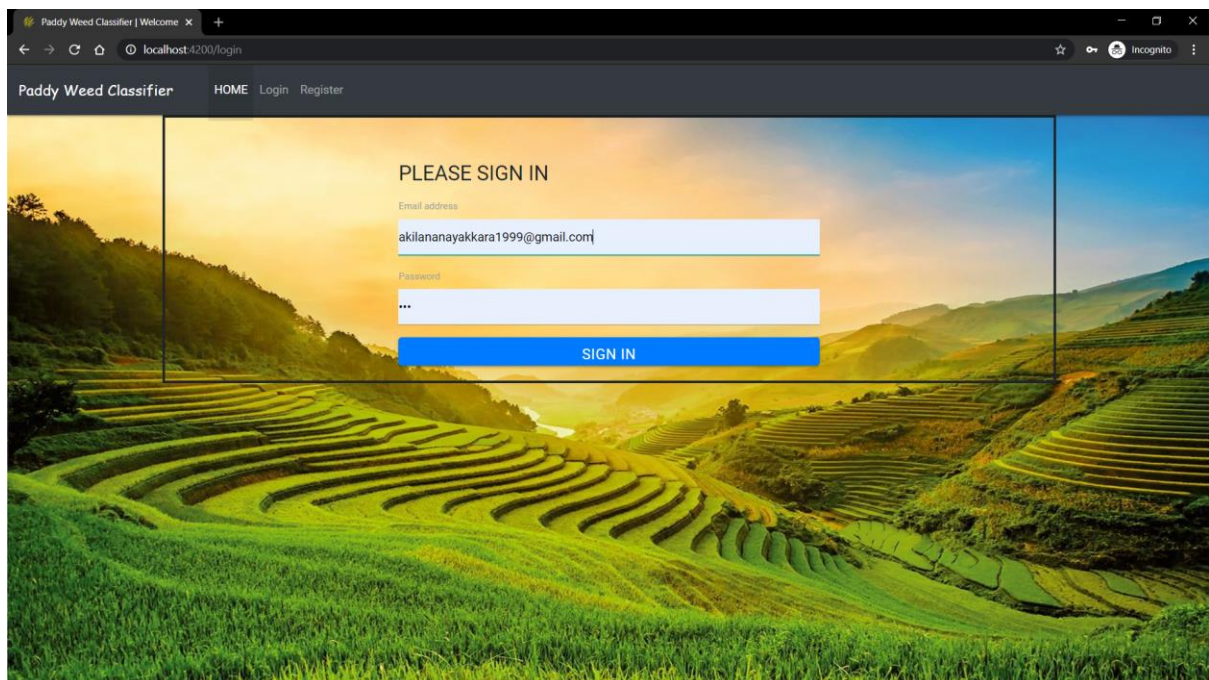


After the above, to use our image classifier user should log in to our system. If the user is not there in our system. User should sign up.



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/register'. The page title is 'Paddy Weed Classifier | Welcome'. The navigation bar includes 'HOME', 'Login', and 'Register'. The main content area features a background image of terraced rice fields. Overlaid on this is a 'PLEASE SIGN UP' form with the following fields: 'First name' (filled with 'Akila'), 'Last name' (filled with 'Nanayakkara'), 'Email address' (filled with 'akilanayakkara1999@gmail.com'), and 'Password' (filled with '\*\*\*'). A blue 'REGISTER!' button is at the bottom of the form.

Figure 16 User sign in



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/login'. The page title is 'Paddy Weed Classifier | Welcome'. The navigation bar includes 'HOME', 'Login', and 'Register'. The main content area features a background image of terraced rice fields. Overlaid on this is a 'PLEASE SIGN IN' form with the following fields: 'Email address' (filled with 'akilanayakkara1999@gmail.com') and 'Password' (filled with '\*\*\*'). A blue 'SIGN IN' button is at the bottom of the form.

Figure 17 User Log in

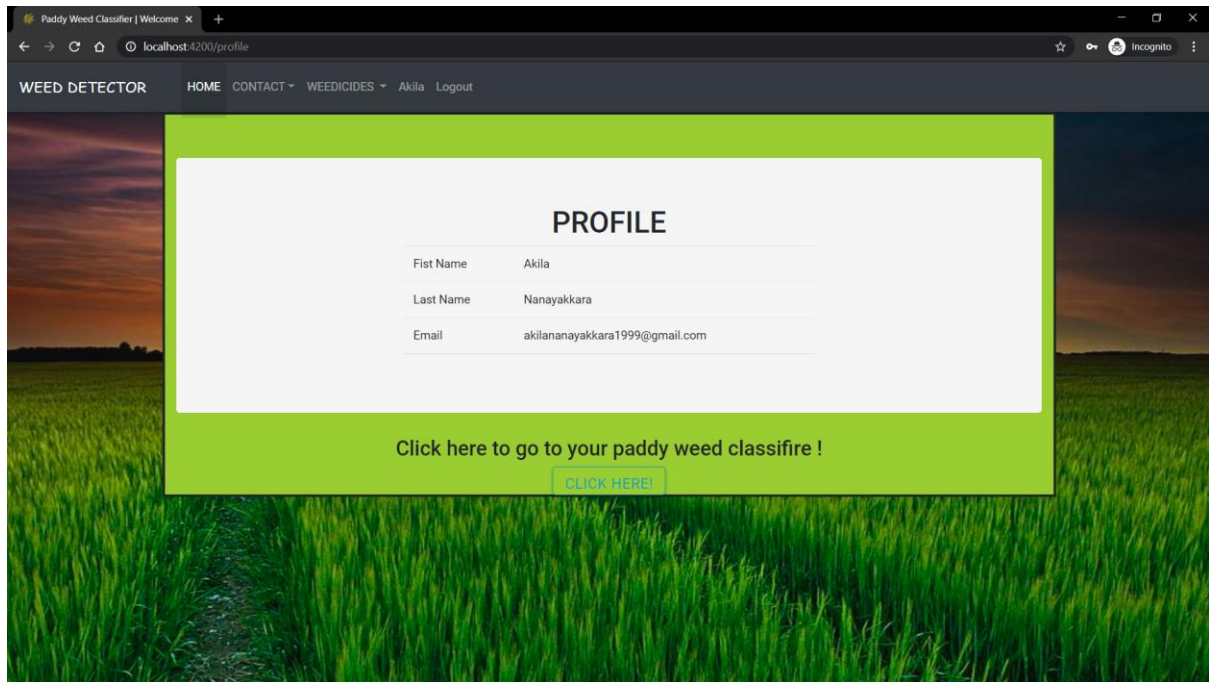


Figure 18 Image Classifier Access

In the above snippet when the user logged in to our system. It will recognize the user correctly and then it will give the direction to use our image classifier.

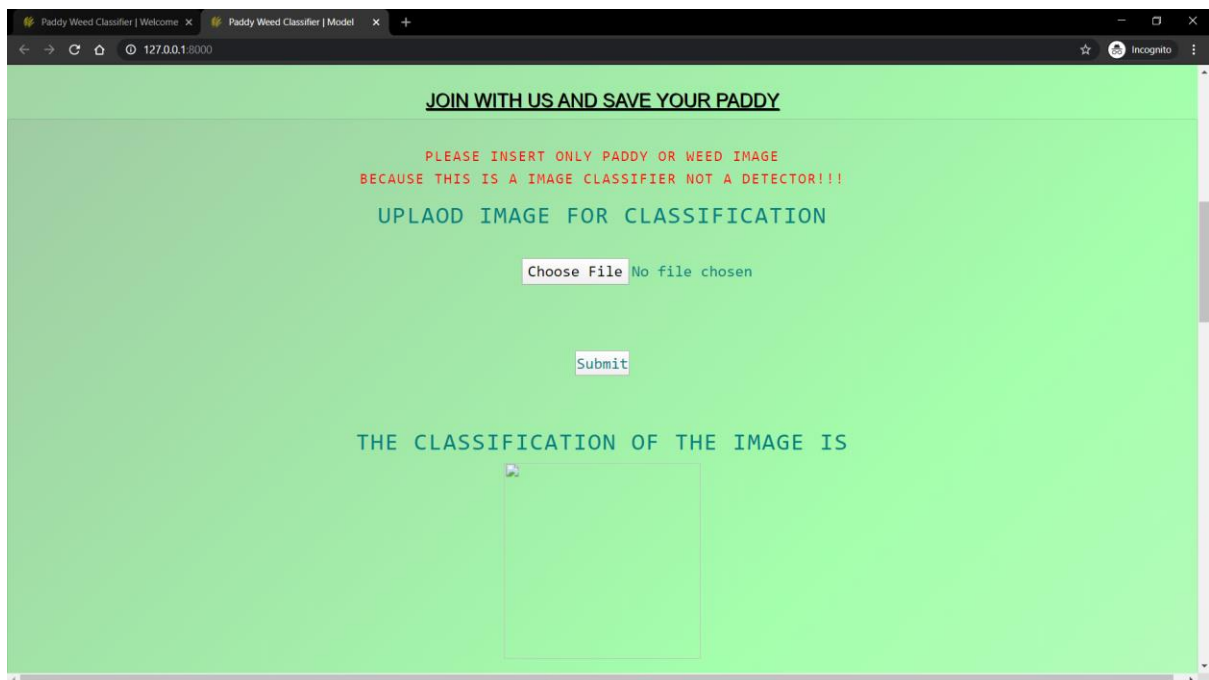


Figure 19 Image classifier input

Here the user can insert a paddy or a weed image and classify where it is a paddy or a weed. Also after upload the image, by clicking on the image user can get details about the image and the user can get the location where the image was taken.

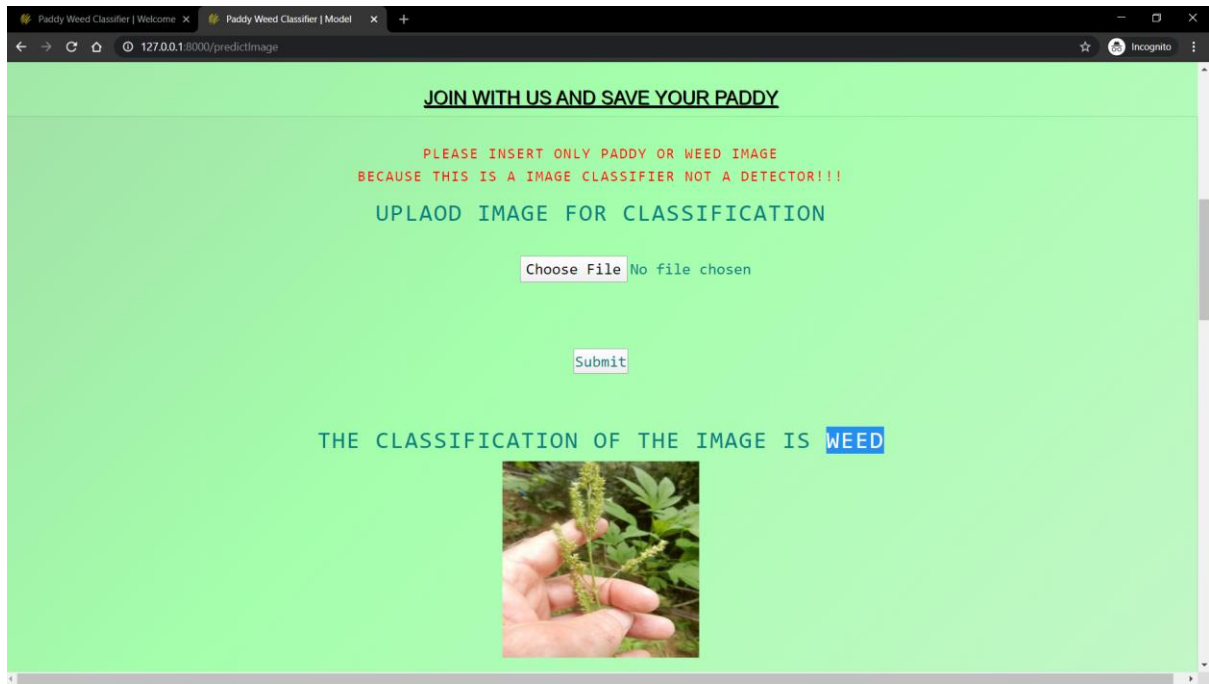


Figure 20 System weed detecting

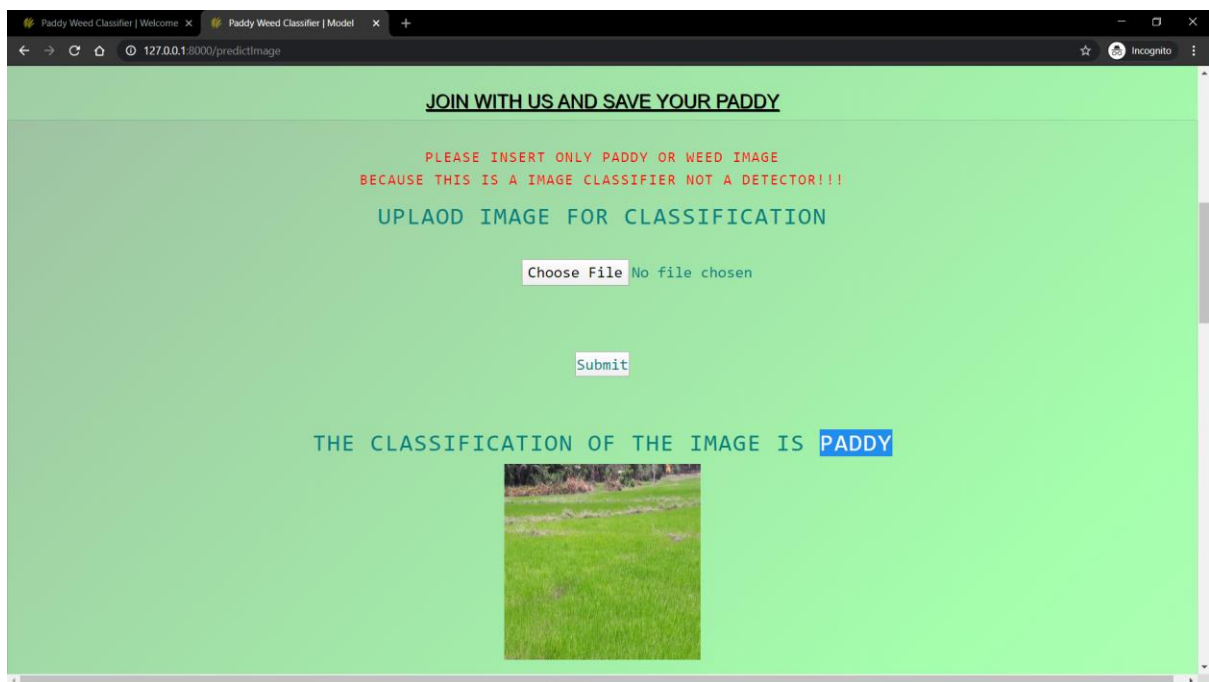


Figure 21 System paddy detecting



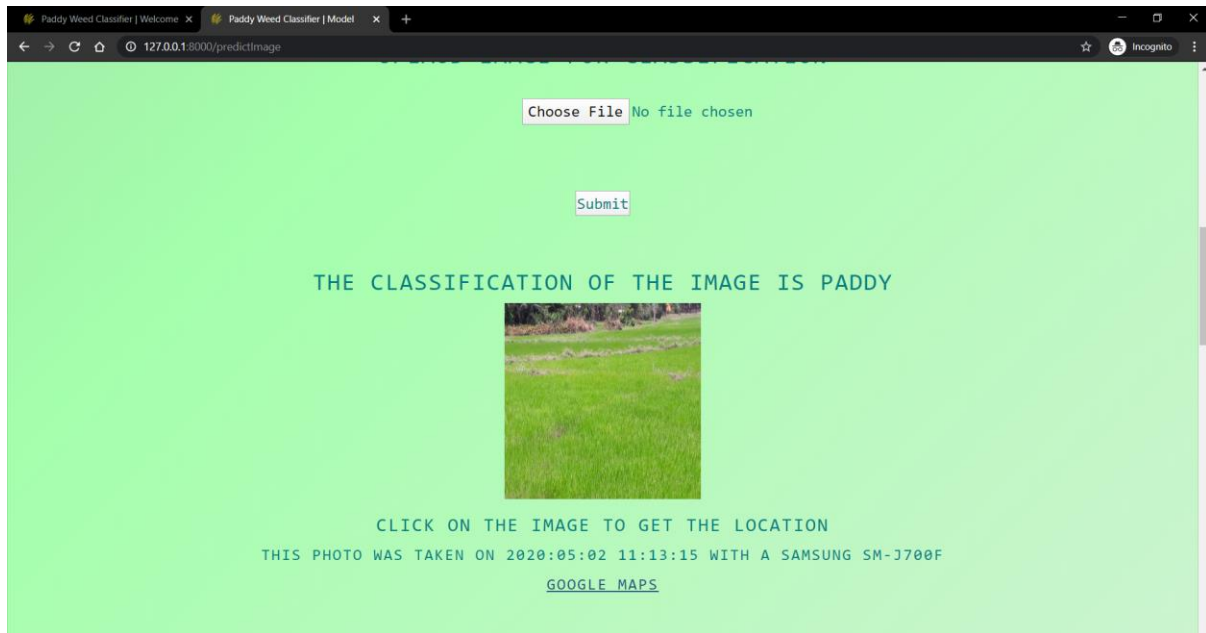


Figure 22 System showing the details and the location

When we click on the google maps link we can get the location from maps.

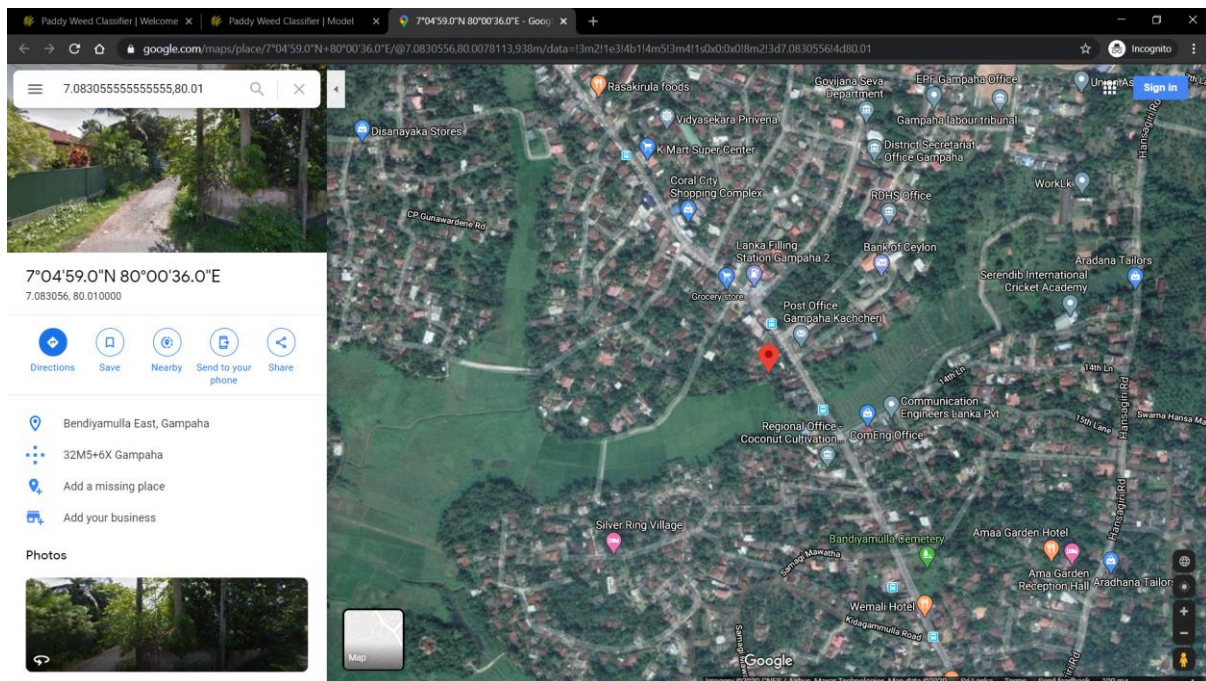


Figure 23 Location where the image was taken

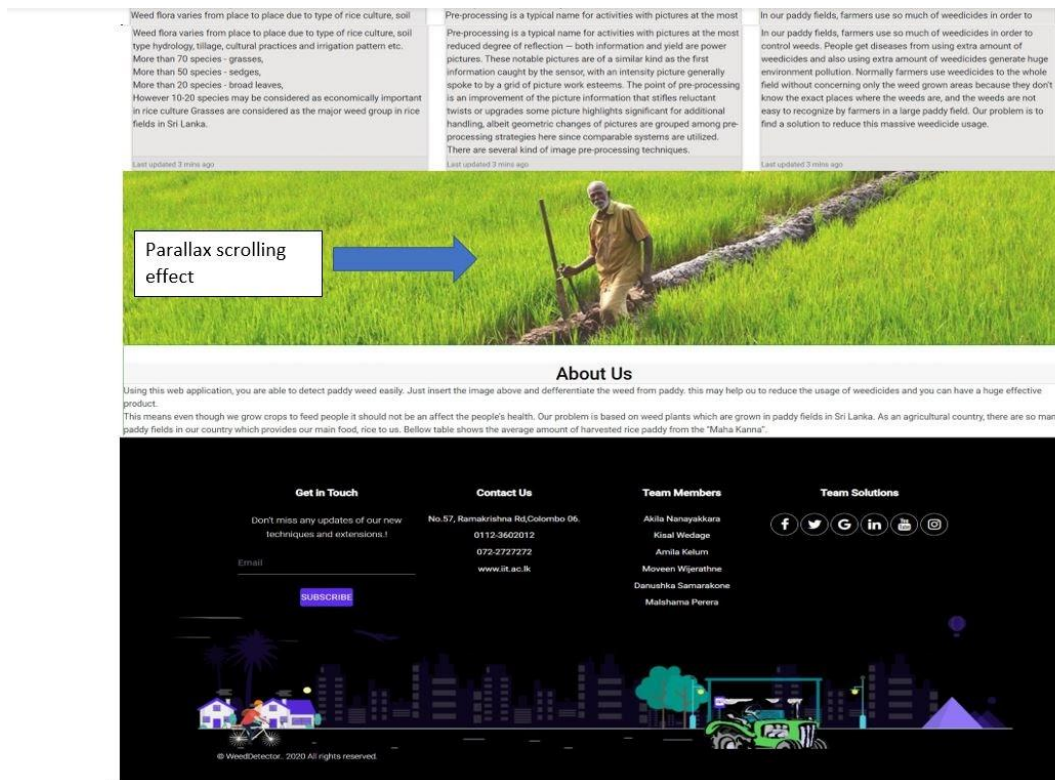


Figure 24 Input page 2 Angular

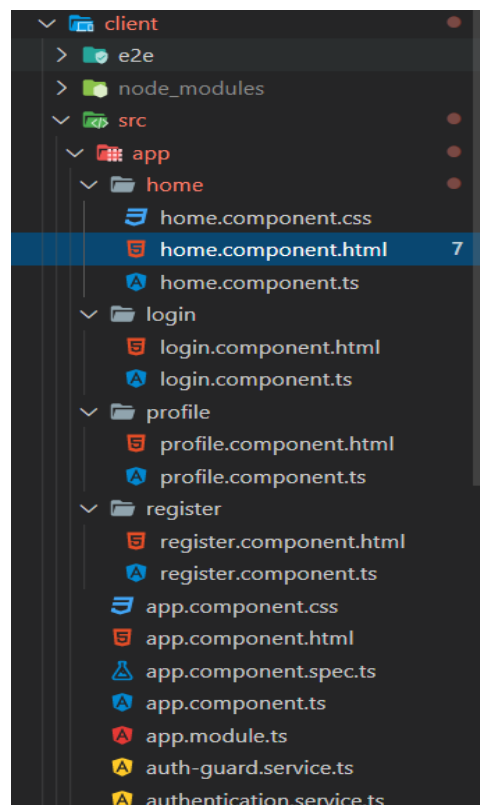


Figure 25 Input page Components

The following figure shows the components that are used for constructing the frontend using angular Js.

The following html and css code snippet declares the image inserting part in the front end.

```
<div class="card">
  <div class="card-body">
    <div class="mat-headline" style="text-align: center;">Insert the Image here</div>
  </div>

  <form #imageForm=ngForm style="text-align: center;">

    <img [src]="imageUrl" style="width: 250px;height: 250px;" />
    <input type="file" #image accept="image/*" (change)="handleFileInput($event.target.files)">
    <button class="mat-raised-button" [disabled]="image.value===' ' || !imageForm.valid">SUBMIT</button>
    <button class="mat-raised-button" [disabled]="image.value===' ' || !imageForm.valid" type="reset">REMOVE</button>
  </form>
</div>
```

Figure 26 Code to get the image

```
ngOnInit() {
}

handleFileInput(file: FileList){
  this.fileToUpload=file.item(0);

  //preview of the image
  var reader=new FileReader();
  reader.onload=(event:any)=> {
    this.imageUrl=event.target.result;
  }
  reader.readAsDataURL(this.fileToUpload);
}
```

Figure 27 Code to show the added image

## 1.9. Back-end and its Combination

We have used node and node express to combine our front end and the back end. We used MySQL database to store our user details and then to retrieve them we used node express.

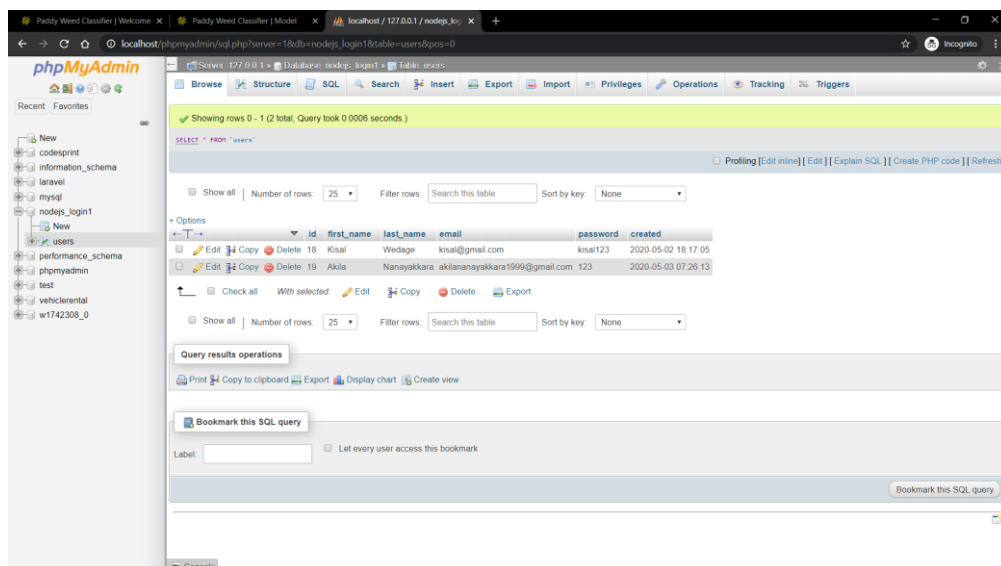


Figure 28 Database of our system

```

server.js > ...
1  var express = require('express')
2  var cors = require('cors') 5.2K (gzipped: 2.1K)
3  var bodyParser = require('body-parser')
4  var app = express()
5  var port = process.env.PORT || 3000
6
7  app.use(bodyParser.json())
8  app.use(cors())
9  app.use(
10   bodyParser.urlencoded({
11     extended: false
12   })
13 )
14
15 var Users = require('./routes/Users')
16
17 app.use('/users', Users)
18
19 app.listen(port, function() {
20   console.log('Server is running on port: ' + port)
21 })

```

Figure 29 Server – express

```

db.js
database > db.js > [?] <unknown>
1  const Sequelize = require('sequelize')
2  const db = {}
3  const sequelize = new Sequelize('nodejs_login1', 'root', '', {
4    host: 'localhost',
5    dialect: 'mysql',
6    operatorsAliases: false,
7
8    pool: {
9      max: 5,
10     min: 0,
11     acquire: 30000,
12     idle: 10000
13   }
14 })
15
16 db.sequelize = sequelize
17 db.Sequelize = Sequelize
18
19 module.exports = db

```

Figure 30 Database Connection

For our image classification backend, we used Django. The reason for using Django is that there is a part of image pre-processing to be done when the user input an image. So, to do that we need python, that's why we used Django as our backend.

```

categories = ["paddy", "weed"]

model_graph = Graph()
with model_graph.as_default():
    tf_session = Session()
    with tf_session.as_default():
        model=tf.keras.models.load_model('./model/paddyWeedDetectorModelWithArch.h5')

```

Figure 31 Loading the model to backend

Here, we are naming our two categories and then we are importing our classification model to the backend.

```

IMG_SIZE = 50

def index(request):
    context = {'a': 1}
    return render(request, 'index.html', context)

def predictImage(request):
    print(request)
    print(request.POST.dict())
    fileObj=request.FILES['filePath']
    fs=FileSystemStorage()
    filePathName=fs.save(fileObj.name, fileObj)
    filePathName=fs.url(filePathName)
    testimage = '.' + filePathName

    img = image.load_img(testimage, target_size=(IMG_SIZE, IMG_SIZE))
    x = image.img_to_array(img)
    x = cv2.imread(testimage, cv2.IMREAD_GRAYSCALE)
    x = cv2.resize(x, (IMG_SIZE, IMG_SIZE))

    x = x.reshape(-1, IMG_SIZE, IMG_SIZE, 1)
    with model_graph.as_default():
        with tf_session.as_default():
            prediction1 = model.predict(x)

    predictedLabel = (categories[int(prediction1[0][0])])

    context = {'filePathName': filePathName, 'predictedLabel': predictedLabel}
    return render(request, 'index.html', context)

```

Figure 32 Backend image pre processing

Here we are getting the image from the front end and then we are gray scaling and resizing and do the main preprocessing part to that input image. Then are feeding that image to our model. Then the model will give the classification. After that we are passing the value to the front end.

### 1.10. Chapter Summary

This chapter began with the technologies that are being used, libraries and the tools. The code snippets and the output snippets were provided when it is needed when the functionality is discussed. First, we have shown the input and the output of our system. How it works. The login page and the image classification system and the map location. Then we discussed Then we have talked about the front and the backend connection in depth and also about how the backend is working. Next chapter is testing which focused on the approaches carried-out to test our system.



## 2. Testing

### 2.1. Chapter Overview

The previous chapter looked into how the implementation is done for the system. This chapter initially outlines the various testing done on the system and the goals that are expected to be achieved through the testing phase and the testing criteria is discussed. The tests covered functional testing and finally the chapter is concluded by non-functional requirement testing, Accuracy, usability and compatibility testing.

### 2.2. Testing Goals and objectives

The purpose of the testing is to verify the system requirements are up to expect in the system. Objectives that are targeted by the testing phase are highlighted below.

- To verify features of the system and validate, in terms of functional requirements.
- To verify features of the system and validate, in terms of non-functional requirements.
- To verify potential bugs and rectify undetected in development.
- To add more necessary in improvements considering the testing phase.

### 2.3. Testing Criteria

Testing is mainly conducted based on the following functional and non-functional requirements to measure the quality of the implemented system.

- Functional quality - To identify the functional requirements will work in this system. This criteria is to make sure the goal are achieved or not
- Non-functional testing - In here more focused on the accuracy and the performance of the current system are achieved, which supports delivering of the functional quality.
- 

### 2.4. Testing Functional Requirements

Testing functional requirements done using black box testing approach. After conducting various test cases and create summary result of the test cases related to functional requirements contained in below table.

ID	Functional Requirements	Status
FRT01	The user is able to put the image into the system.	Success
The system must have an option to get the input image from the user.		

FRT02	The system is able to detect the weed in the paddy field.	Success
Main target of the system is to detect the specific weed that is on the image.		
FRT03	The system is able to give the exact location where the weeds are.	Success
FRT04	The system should give the exact location where the weeds are.	Success
One of the features of our program is to show the location where the weeds are. To do that we will extract the location from the image and show it to the user.		
FRT05	Identify other weed plants other than the “MahaMaruk” weed plant.	Success

Table 1 Testing the functional requirements

### 2.5. Testing Non-Functional Requirements.

Non-functional requirements more clarifies how a system ought to act as opposed to functionalities of the system. Non-functional requirements are also important as functional requirements, considering it affects the overall performance and the accuracy of the system. Identifying and listing of non-functional requirements are done in the SRS.

### 2.6. Performance Testing

Since the system is developed as a web application, performance of the system is a main requirement. To get the reliable values, each test case is executed several times and the average time is calculated. To be considered as this is a web application, the performance depends on the network connection performance.

ID	Test Case	Input	Expected Output	Actual Output	Status
PT01	Loading the front-end Page	N/A	Less than 25s.	24.634s.	Pass
PT02	Validate whether input is successful or not.	Paddy Image	Less than 25s.	N/A	Fail
PT03	Generate the Final Result	Paddy Image	Less than 5s.	Less than 5s.	Pass
PT04	Extract the location and show	Paddy Image	Less than 5s	Less than 5s.	Pass

Table 2 Performance Testing

## 2.7. Accuracy Testing

As this is a classification web application, accuracy plays a major role. Though the system has various functional requirements in the proposed system, if the system is not accurate the system is pointless, worthless. System is calculated the accuracy of the data science part can be found below.

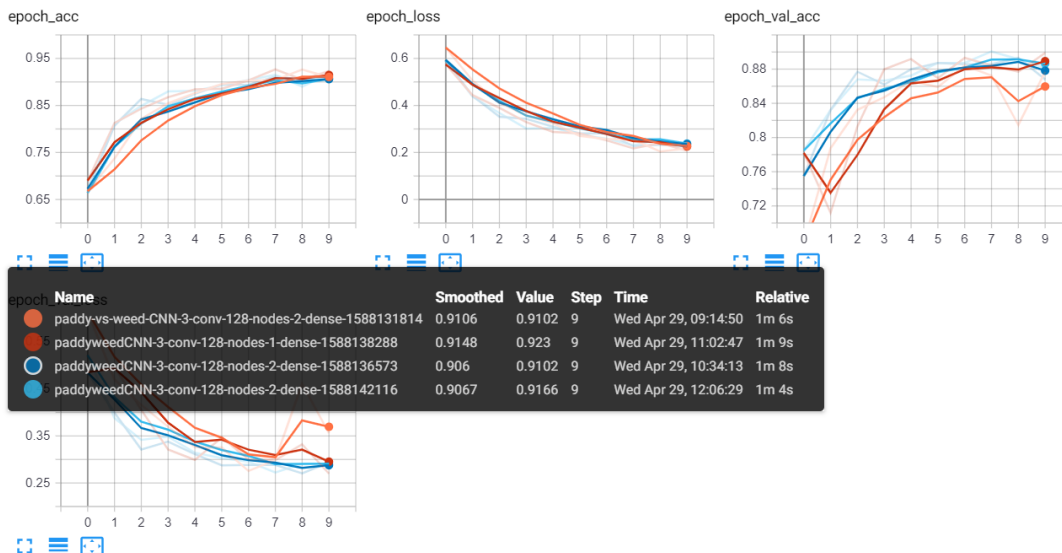


Figure 33 Accuracy testing using tensor board

```
1571/1571 [=====] - 7s 5ms/sample - loss: 0.2198 - acc: 0.9166 - val_loss: 0.2912 - val_acc: 0.8813
```

Figure 34 Accuracy in PyCharm model training

## 2.8. Usability Testing

Here the testing guarantees that the user only needs a minimum guide throughout the system to use and get the output. The user is offered with a user-friendly interface which helps the user to get ease on the application. After logging in to the system, the user can get the classification by only two clicks. If the user is lost how to use the system, the user can jump to the middle of the page to get the necessary information about how to use this application.

## 2.9. Compatibility Testing

Browser	Compatible
Chrome	Yes
FireFox	Yes
Edge	Yes

Table 3 Compatibility testing



### 2.10. Chapter Summary

The chapter starts with discussing the goals and the objective of testing. Then the functional and the non-functional testing was focused with the testing criteria. Non-functional requirement testing was divided into four performance, Accuracy, usability and compatibility) and discussed briefly with screenshots and tables. The next chapter is discussed about the critical evaluations done for the system.

## 3. Evaluation

### 3.1. Chapter Overview

### 3.2. Evaluation Goals

### 3.3. Evaluations Benchmark

### 3.4. Evaluations Selections

### 3.5. Concept and Scope

### 3.6. Technical Aspects

### 3.7. Impacts

### 3.8. Limitations

### 3.9. Research question

### 3.10. Self-Evaluation

### 3.11. Chapter Summary

## 4. Conclusion

### 4.1. Chapter Overview

This chapter covers the achievements of the project. The chapter firstly discusses the project aims and objectives. Further, the problems faced, how the existing knowledge of the members utilized with the degree program. Limitations of the projects are outlined followed by the learning outcomes of the project. The chapter proposes the future enhancements.

### 4.2. Achievement of the Project Aim

“This project aims to build up a system to recognize the weed plants in-between the paddy and to generate the map with location showing where weeds are.”

The aim of the project was successfully achieved during the development time. System that detects the weed and maintaining was successfully designed, developed and evaluated. With the original camera photographs with higher quality images made possible with the system.

### 4.3. Achievements of Operational Objectives

Description	status
<b>Validation of the idea and the introduction</b>  To gather proper knowledge about the problem. 1. Research on the problem domain. 2. Get the idea validated according to the problem domain. 3. Conducting interviews with domain experts. 4. Distributing a questionnaire to validate the problem.	<b>Completed</b>
<b>Literature Review</b>  Researching about the problem and its existing solutions. 1. Discovering the existing methods to solve the problem. 2. Comparing the different approaches to solve the problem.	<b>Completed</b>
<b>Choosing our approach</b>  With the help of the literature review, find out the best approach that we can solve the problem.	<b>Completed</b>
<b>Requirement gathering</b>  Gather user requirements mainly to come up with functional and non-functional requirements which are necessary.	<b>Completed</b>
<b>Designing the system</b>	<b>Completed</b>

The basic domain models, onion diagram, use case diagrams and its descriptions and class diagrams, use case diagrams. This is the basic overview of the system which anyone can understand easily.	
<b>SRS</b>  Preparing the SRS document with the help of objective three and requirements engineering process.	<b>Completed</b>
<b>Proof of Concept</b>  Initial development stage of the implementation which will include the core functionality and other functionalities depending on what time permits with the help of the best software and hardware which were validated on previous objectives.	<b>Completed</b>
<b>Test and evaluate the Proof of Concept</b>  1. Write test cases. 2. Run the tests on the implementation. 3. Record the outputs. 4. Validate them and check if it meets the requirement.	<b>Completed</b>
<b>Build up the Proof of Concept and complete the implementation.</b>	<b>Completed</b>
<b>Test and evaluate the implementation.</b>	<b>Completed</b>
<b>Complete the final documentation.</b>	<b>Completed</b>
<b>Conclude the project and submit.</b>	<b>Completed</b>

Table 4 Achievements of operational objectives

#### 4.4. Achievements of Academic Objectives

Description	Achievement
<b>Using data science and applying image processing.</b> Applying python open cv to do the image processing to identify the weed plants from the paddy field.	<b>75%</b>
Learning python, angular js, weka and MATLAB and using them for our project.	<b>80%</b>
Learning designing, developing, testing and evaluating	<b>100%</b>

Practicing the software development life cycle To learn and experience the full SDLC and how to manage a software development project. This will give us exposure in all aspects of software development such as requirement gathering, analysis, design, implementation, testing, deployment and maintenance.	<b>100%</b>
---	-------------

Table 5 Achievements of academic objectives

## 4.5. Achievement of Requirements

### 4.5.1. Achievement of Functional Requirements

Requirement	Description	Achievement
The connection between the system and the dataset.	As this system is based on image processing and data science, the system must always have the connection with the dataset.	100%
The user must be able to put up the drone and snap the paddy field.	User must capture a picture using a drone. The image quality should be at a certain good level.	100%
The user should be able to put the snap into the software.	The system must have an option to get the input image from the user.	100%
The system should detect the weed in the paddy field.	Main target of the system is to detect the specific weed that is on the image.	100%
User Log-in	-	0%
Identify other weed plants other than the “Maha-Maruk” weed plant.	-	0%

Table 6 Achievement of Functional Requirements

#### 4.5.2. Achievement of Non-Functional Requirements

Requirements	Achievement
The software should load up soon and then identify the weed quickly.	50%
Identifying the weed should be accurate.	80%
The system should improve when it gets more and more images	90%
The system is a web application, it can be accessed through a laptop from any location, when the user has a stable internet connection.	0%
The system will keep the photos that farmers insert to the system. In the backend, we can use those images and train our system to be more accurate.	100%

Table 7 Achievement of Non-Functional Requirements

#### 4.6. Milestones Deliverables

No.	Milestone/Deliverable	Completed	End Date	Remarks
1	Project Initiation Phase	100%	15/11/19	Project initiation document (idea pitching) was done on time
2	Literature Review Phase	100%	01/01/20	Research reports were referred during the phase. Submitted the final report 1st stage report on time
3	Requirements Gathering Phase	100%	10/02/20	This phase was started and finished on time
4	Design Phase	100%	09/03/20	This phase finishes on time
5	Implementation Phase	100%	25/4/20	Due to various circumstances it was unable to start this phase on time.
6	Testing and Evaluation Phase	100%	26/4/20	Due to the delay of the above phase testing and evaluation phase was in a hurry
7	Project Closure	100%	26/4/20	Unprepared delays pay the price to rush closure

Table 8 Milestones Deliverables

#### 4.7. Problems and Challenges Faced

##### 1. Limited time

- The project nearly carried a year. The time frame was highly challenging to get a better project.

##### 2. Unrest in the country

- Due to the covid-19 virus curfew was lifted, continuous power cuts on some areas in the country. To gain a better outcome team meetup (physical meetups) is key. It was a challenge to the members to communicate and implement the project through online. This was a major challenge that affected the project.

##### 3. Lack of expertise

- It was hard to find the correct and the most accurate solution for an error. This is because of the scarcity of machine learning experts.

##### 4. Implementation of the data science part

- Data science part was asked to submit earlier than the project deadline. Rush coding was needed to survive the challenge. This was harder because the library was not familiar. Had to do massive research before handling the libraries.

#### 4.8. Learning Outcomes

The Learning outcomes that were obtained are mentioned below.

- Skills such as time management, problem solving, analytical thinking and team communication were improved.
- An outstanding improvement has been seen in academic writing and formal report writing abilities.
- Skills such as best practices in coding, commenting, design and architectural style have been improved precisely.
- Totally new knowledge was gained through this project, such as new libraries like koras, TensorFlow, Django.



#### 4.9. Future Enhancements

- The system should be able to detect or locate other weed types or even few more
  - Priority - High
- The system can be developed for detecting more weeds other than “Mahamuruk” if there were more data sets.
  - Priority - High
- The system should be run on mobile devices, such as IOS, android with a minimum time taken.
  - Priority - Medium
- The system with user login, with user information database should be considered
  - Priority - low

#### 4.10. Chapter Summary

This chapter concludes the report by discussing the aim and objectives which are in the very first chapter. How the requirements of the system have been satisfied is being discussed. Then the Milestones and Deliverables have been mentioned with an ending date and the problems and the challenges met are being mentioned and discussed. Learning outcomes of the project are noted following, finally with the future enhancements are documented.

## References

Saha, S., 2018. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. [Online] Available at: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> [Accessed 27 04 2020].

