

**AN INTELLIGENT SYSTEM FOR FORECASTING
FARMER'S REQUIREMENTS IN MAHAWELI
PROJECT**

Project Id – 20_21-J11

Wijetunge W. P. T. T.

IT17167024

Final (Draft) Report

B.Sc. Special Honors Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

May 2021

**AN INTELLIGENT SYSTEM FOR FORECASTING
FARMER'S REQUIREMENTS IN MAHAWELI
PROJECT**

Project Id – 20_21-J11

Wijetunge W. P. T. T.

IT17167024

Final (Draft) Report

(The dissertation was submitted in partial fulfilment of the requirements
for the B.Sc. Special Honors degree in Information Technology)

B.Sc. Special Honors Degree in Information Technology

Department of Software Engineering

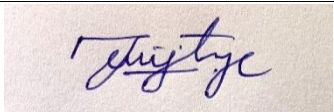
Sri Lanka Institute of Information Technology

Sri Lanka

May 2021

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as article or books).

Name	Student ID	Signature
Wijetunge W. P. T. T.	IT17167024	

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

.....

Dr. Pradeep Abeygunawardhana
(Supervisor)

.....

Date

ABSTRACT

In this document, I present how our system, the intelligent system for forecasting farmer's requirements in Mahaweli project; will help for farmers. Mainly this paper is about the component that wraps out, checking healthiness of a specific crop. Farmers can check the healthiness and the quality of a specific crop under our intelligent system for forecasting farmers' requirements in Mahaweli project. But it also can be used by other farmers who are living in Sri Lanka.

In this component, I explain the implemented platform to check healthiness of a specific crop. In order to capture healthiness of a specific crop, farmer simply need to hold the camera mobile above the desired crop. And it will predict the related healthiness of that specific crop. It will be a very user-friendly product for the user, because with any level of technology knowledge farmers could be able to use the product. And this platform will very much helpful to farmers and other agricultural people to check healthiness of that crop, otherwise they only do healthiness checking manually through eye level.

User needs to hold the mobile camera above the desired crop. And it will be uploaded into backend in real-time. Through that, deep learning model will be called and image classification will be done. And final level of healthiness will be displayed for user as well. I hope this research component will be useful to Mahaweli farmers as well to the every other farmer in Sri Lanka.

Key words: intelligent system, checking healthiness, agriculture, d

ACKNOWLEDGMENT

This research would not be possible without the assistance and dedication of numerous individuals and in the process of making an idea to a successful working product, we had to go through a difficult time period. Hardworking of the group members and the great support and the encouragement which we had laid the red carpet for this research.

First, I want to appreciate to my group members Vishal Thenuwara, Anuththara kavindi and Gopika De Silva for their greatest participation in our research. This research has become a reality with the dedication and hard working of the group members.

As well, I express my greatest appreciation to Dr. Pradeep Abeygunawardhana for his invaluable support for this research. Our research “An Intelligent System for Forecasting Farmer’s Requirements in Mahaweli Project” has become a reality with the guidance and the immense supervision within few months with the bless of our supervisor.

Finally, I dedicate my gratitude to my parents, my grandparents and my close friends who were always beside and helping me in every second in the way of achieving this success. And all others who were with me in every leap and bounds in the process of successions this research. As well, I am thankful for everyone who gave their priceless support and guidance to make this idea a success.

TABLE OF CONTENT

Declaration	ii
Abstract	Error! Bookmark not defined.
Acknowledgment.....	Error! Bookmark not defined.
Table of Content	Error! Bookmark not defined.i
List of Figures.....	Error! Bookmark not defined.
1. Introduction.....	Error! Bookmark not defined.
1.1. Background literature	1
1.2. Research gap	4
2. Research Problem.....	5
3. Research Objectives	6
3.1. Main Objectives	6
3.2. Specific Objectives	6
4. Methodology	7
4.1. Methodology	7
4.1.1. Delivery methodology	7
4.1.2. Implementation methodology	9
4.1.2.1. System overview	10
4.1.2.2. Android implementation methodology	13
4.1.2.3. Deep learning model implementation methodology	18
4.1.3. Gantt chart.....	28
4.2. Commercialization.....	29
5. Implementation and Testing	30
5.1. Implementation	30
5.2. Testing	33
6. Results and Discussion	34
6.1. Results	34
6.2. Discussion	39
7. Conclusion	40
8. References.....	42
9. Appendices	43

LIST OF FIGURES

• Figure 1 – Mahaweli map	1
• Figure 2 – MS Teams task board	7
• Figure 3 – GitLab branches	8
• Figure 4 – GitLab commits	9
• Figure 5 – Iterative waterfall model	10
• Figure 6 – System overview diagram	11
• Figure 7 – Home page interface	12
• Figure 8 – Two buttons interface	12
• Figure 9 – Camera features in Manifest	13
• Figure 10 – Camera services	14
• Figure 11 – Camera permission	14
• Figure 12 – Set fragment	15
• Figure 13 – Set autofocus	16
• Figure 14 – Get model and labels	17
• Figure 15 – Set result values	17
• Figure 16 – Python imports	19
• Figure 17 – Data preprocessing	20
• Figure 18 – Plot images	21
• Figure 19 – Sequential model	23
• Figure 20 – Model summary	23
• Figure 21 – Training the model	25
• Figure 22 – Classes of test batches	26
• Figure 23 – Plot confusion matrix define	27
• Figure 24 – Gantt Chart	28
• Figure 25 – Android folder structure	30
• Figure 26 – Assets folder	31
• Figure 27 – Gradle dependencies	32

- Figure 28 – Model prediction testing 33
- Figure 29 – Onion healthiness predict interface 34
- Figure 30 – Potato healthiness predict interface 35
- Figure 31 – Modal pop up interface 36
- Figure 32 – List view interface 37
- Figure 33 – Confusion matrix results 38
- Figure 34 – Turnitin similarity report 43

1. INTRODUCTION

1.1. Background literature

In a tropical country like Sri Lanka (formerly Ceylon), agriculture takes an important point of living and economy. As the chief occupation of the Sri Lankan's was agriculture, since the ancient times, each and every land in Sri Lanka were majorly used in order to farm and grow their own cultivations. Farmers who inspired with agriculture throughout years are spread out all over the island. Not like in present, people grow their own food and was very hard to find any foreign agricultural food trade. In earlier stages of agriculture, the main cultivation was only paddy. But gradually, farmers moved into vegetable and fruits cultivations as well. As government decided to start a development program called Mahaweli Project, it became the largest multipurpose agricultural movement in Sri Lanka. Throughout all the other areas, Mahaweli area cultivate significant number of crops based on two periods called 'Yala' and 'Maha'. For both periods, farmers produce mass number of crops to the entire country. And this Mahaweli project can be called as the largest agricultural and other developments in the country.

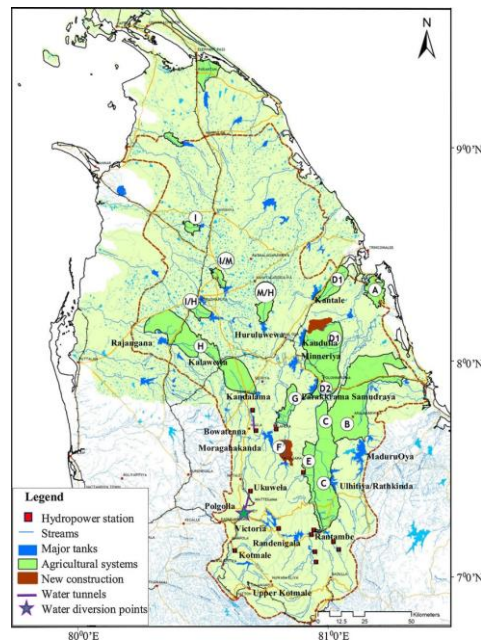


Figure 1: Mahaweli map

Even though in agricultural aspects farmers use some basic development technologies [1], they don't use any intelligent based technology in order to support their farmers' productivity and day to day farming purposes. Because of that since past farmers got occurred various problems such as what crop will be suitable for what season, weather a specific product is in good healthiness with quality or not, what price range they should sell their crops, likewise. If there is any kind of tool that will provide harvest predictions, healthiness and the quality of a crop, analysis of the up to date market and a way to sell out their products, they can be more productive in every kind of areas in agricultural development.

As we compare Sri Lankan farmers with the other country's farmers, in the past Sri Lankan farmers have used various techniques in agriculture to cultivate their crops. But in present, in foreign countries, they are using more new techniques in order to improve their cultivations. Those new techniques need to be carried out to our own farmers to improve the productivity and quality of their cultivations. This also will help to Sri Lankan economy as more productivity is coming from the local agriculture, there is no need to import the desired crops from foreign countries.

When talking about healthiness and quality prediction, nowadays farmers only categorize crops into different categories according to different qualities by just seeing by their eye level. And it is not well predicted as well. Because of that, there pricing categories may become a problem when selling their crops with different ratio of qualities. So, farmers should maintain a proper quality wise divided crop categorization in order to sell their crops for a good price point. In order to do that, they should use a proper way of measuring healthiness and the quality of a specific crop after they harvest the crops. So, process of taking decision is bit complex as there are several factors affecting to healthiness and quality prediction.

With use of the intelligent app according to this component, hopefully farmers will be able to predict the healthiness or quality with some good probability. The app will be built with user friendliness and easy to use by the farmers in order to get categorize their crops level of healthiness wise. This will be a major incident in farming as they can use way more productive way to measure healthiness of a specific crop.

There have been lots of researches on intelligent systems for used in agriculture. In the sequence of developing proposal and the intended component of capturing and predicting healthiness or quality of a specific crop these additional research articles will be helpful.

In order to become more productive in agricultural resources some farmers in worldwide have applied Agricultural information Technology (AIT) [2]. And as sub system of AIT, using technology for agricultural management has a huge impact on managing crop harvesting and in the production side also. It used to analyze features that capture from agriculture data also.

Researches have used to find out numerous ways to capture a difference of a specific crop. As an example, few people have developed a computer system to recognize and classify freeze-dried carrot and microwave vacuumed dry products [3]. In order to implement this system, they have used image processing and artificial neural networks.

As mentioned above image analyzing and neural networks, we can recognize differences of either vegetables or fruits. As carrots, some researchers have done a significant job on recognizing the potato varieties [4]. They developed the system to prepare image database with the help of image processing and analyzing. And the neural model is used to identify the key differences of the varieties and recognize the potato.

And also, some researchers have done quality analysis on wheat grains using image processing with the help of machine learning algorithms [5].

Therefore, using an image, analyze it and predict the healthiness or the quality of a specific crop will be more useful feature to all the farmers in order to gain more productivity and when pricing different qualities of that specific crop.

1.2. Research Gap

In predicting healthiness or quality of a specific crop after harvesting, there isn't any researches done related to this area even in the world, according to my knowledge. In order to get the idea of what quality their crops are in; farmers need to capture an image and it will predict that crop's healthiness.

But in earlier, people have done researches on capturing a fruit and check whether it is dried or not [3]. This will predict the status of the fruit other than healthiness or the quality. In my proposed component's solution, it will **directly be addressed into the healthiness** of that specific product and will predict answer.

As another example, some researches have done a research on image analyzing using neural network to find out the variety of potatoes [4]. The aim is to classify and keep different varieties of potatoes in heap. And this also, will not aim on the side of predicting the healthiness or the quality of specific crop.

So, currently, there is **no direct platform to predict healthiness** of a specific crop. As the gap defines likewise, I hope this research component will help to predict the healthiness or quality and achieve farmers' requirements in categorizing their crops healthiness wise.

2. RESEARCH PROBLEM

Each and every agricultural person lives in Sri Lanka needs a healthiness and quality review of their cultivated crops. They need to categorize according to various quality levels, because that's how they can divide them into different pricing categories. But nowadays in Sri Lanka, there isn't any kind of proper quality checking method for farmers. Instead they just look at harvested crops at eye level and predict a near quality for them.

When the process of pricing according to quality varieties, they actually need a proper healthiness or quality measurement technology. Otherwise they will try from their own as mentioned above and may divide them in some wrong manner. At it might occur problems when selling their crops also. Although farmers and agricultural personalities are not very much knowledgeable in tech side, if they can carry out some method that actually helps them to predict quality of their specific crop in a very easy manner, they will transfer to that method. Because every person is more concern about this quality and healthiness side and this will be more helpful for them.

This is the point my component of this research proposal will point out. It is easy to use by a zero knowledgeable person. User will only need to capture an image of their specific harvested crop and it will predict a level of healthiness or quality of that specific crop. This will mean a lot to farmers that they could get a fair point of healthiness and they can mark and categorize their pricing varieties more easily. So, it will surely helpful for all agricultural people.

3. RESEARCH OBJECTIVES

3.1. Main Objective

Main objective in this component is to develop a platform to address a level of healthiness of some specific crops using machine learning models and image classification.

3.2. Specific Objectives

- Develop a mobile environment for farmers to capture the image of the crop that they need to measure the healthiness level.
- Create a machine learning image classification model to identify the crop that captures through the camera.
- Create or enhance a deep learning image classification model to measure a level of healthiness of the crop that captures through the camera.

4. METHODOLOGY

4.1. Methodology

Here I will discuss the methodological approach toward delivery and implementation of the component, addressing the healthiness of some specific crops.

4.1.1. Delivery Methodology

This research component which is to address the healthiness of some specific crops, has followed best practices throughout this research project period. Because of following best practices to complete the tasks of this component, the objectives which needed to be completed are successfully covered within the planned time period.

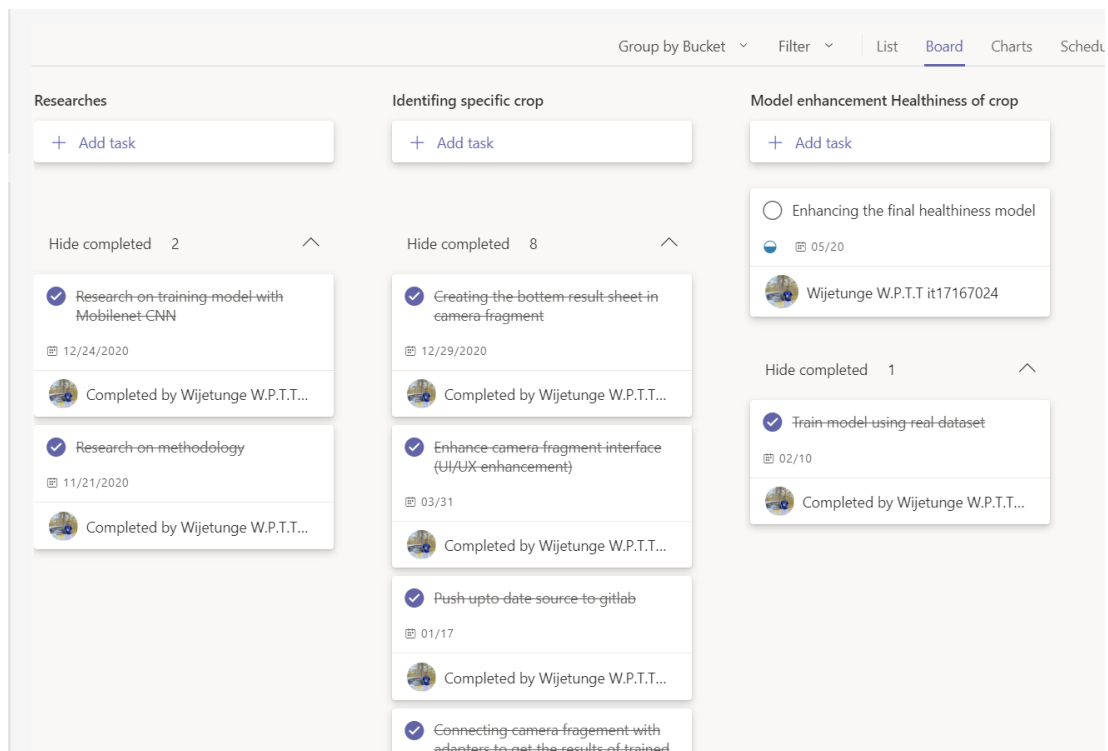


Figure 2: MS Teams task board

Mainly tasks with the Gantt chart has done using MS Planner and continuing the tasks has done using MS Teams planner. Also the MS teams schedule and charts were used to enhance the management level of this research component. And the other main added value to the deliverable is the integration of the system with all four components of the research. Mainly Git branching strategy has used in order to maintain code repository and in the integration purpose. Maintaining branches reflects the proof of a production level application. Also this will avoid the conflicts with the other components when merging all sources into one branch. This is an added advantage in timing manner and quality of the code manner as well.

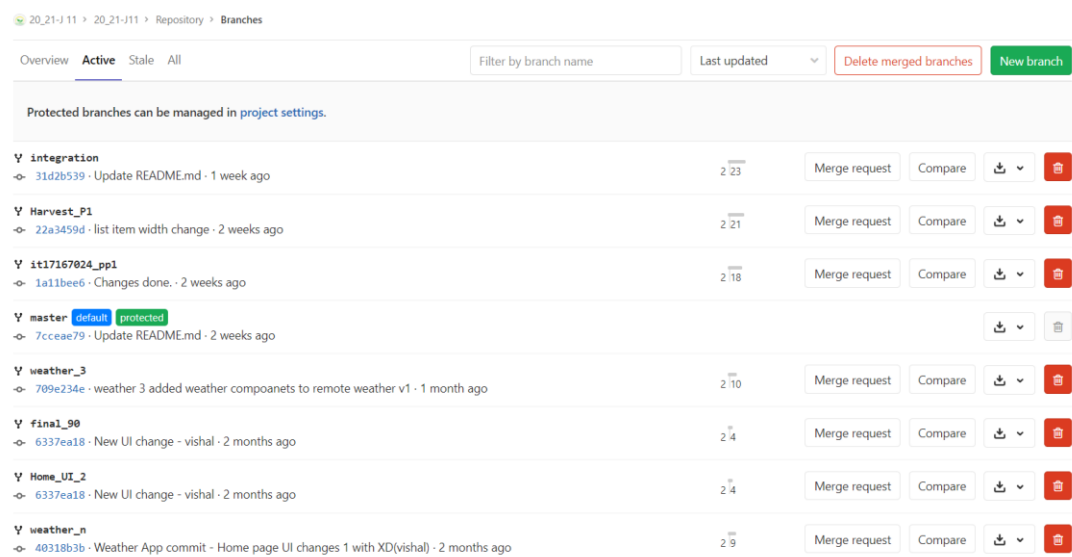


Figure 3: GitLab branches

Not only maintaining branches, also regular commits allows me to keep the repository of my research component up to date in case of lose code. Further, committing into my own branch helps me to carry my code even with few errors and when integrating with other three components it was way easier in that way. A graphical representation of few commits can be shown in the below figure.

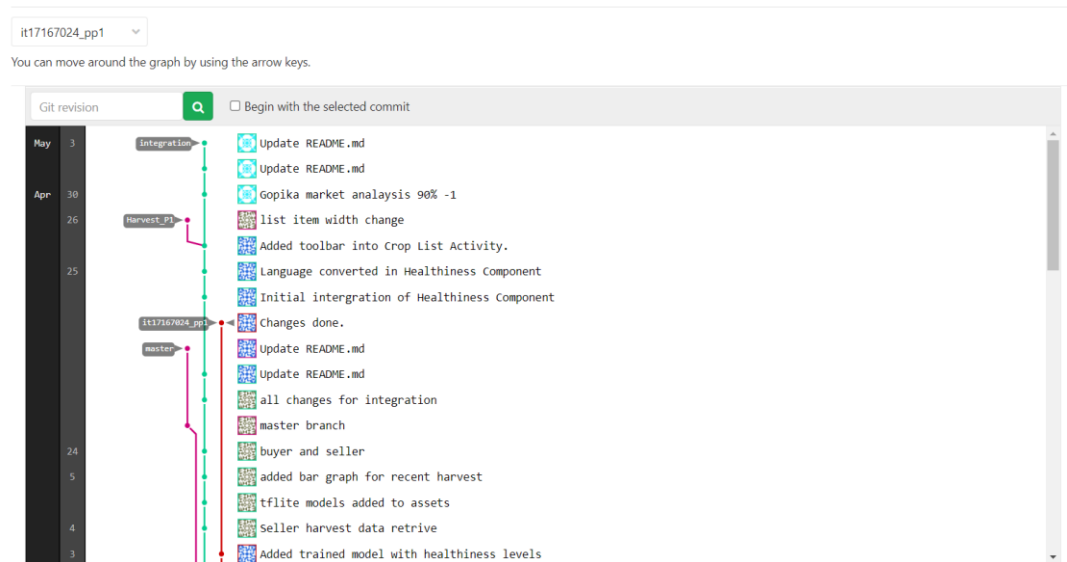


Figure 4: GitLab commits

4.1.2. Implementation Methodology

In this part, the methodology which I have followed during the implementation part is described. This research component is all about the measure a level of healthiness of some specific crops using a mobile device camera just by holding it to the crop.

As this is a research project and only one year is allocated, I preferred using iterative waterfall model as the main approach toward this project. And also because of fewer knowledge of technologies at start, this iterative model was more suitable as to switch to back when needed to undo what have done wrongly.

In the requirement analysis stage, mainly the requirements were gathered from the Mahaweli authority documentation and from internet and other articles. The system design has done using diagrams (both UML and high level). In the implementation stage, all designed system has converted into coding and gradually built the actual product.

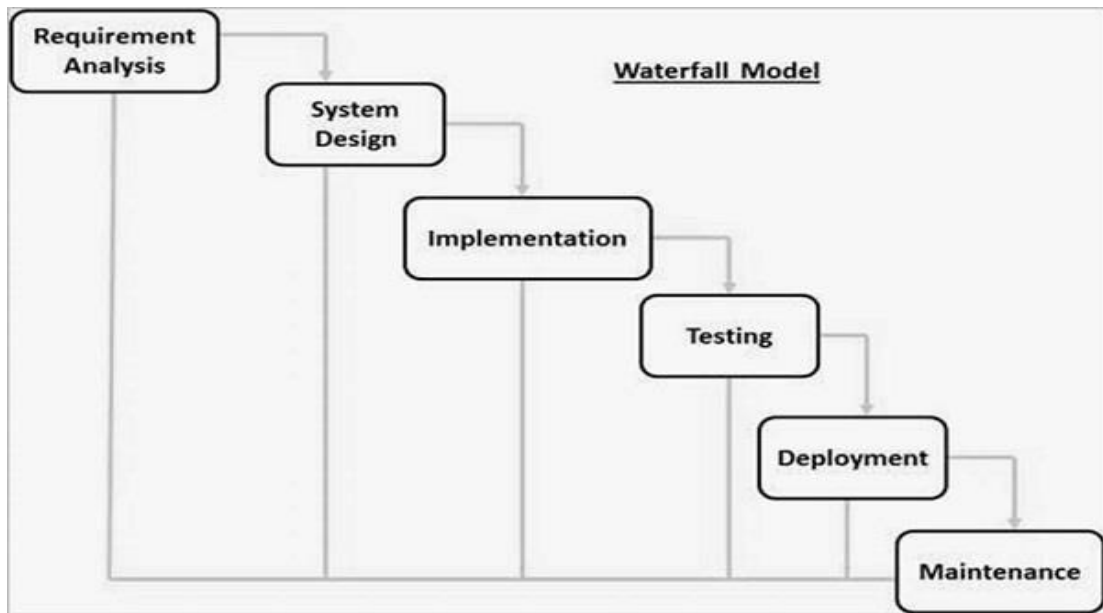


Figure 5: Iterative waterfall model

4.1.2.1. System Overview

Basically, we have implemented the system on Android using Java. As the research consists of four components, the project was split into 4 major android packages which runs on same core modules. For coding in android studio, pure java concepts have been used and the trained models were included into assets sector of the android project. As this component is all about capturing and measuring the level of healthiness of specific crops, a camera component and some retrieve UIs were used to get interacted with the user.

So, for uploading the crop image and for showing the expected results a mobile front end has been used. And the whole front end is connected with java OOP code segment in the backend. As I mentioned before, the deep learning model is included in assets sector in order to allow access to the backend code segment. Also, as database purposes, firebase has been used in connected with the backend.

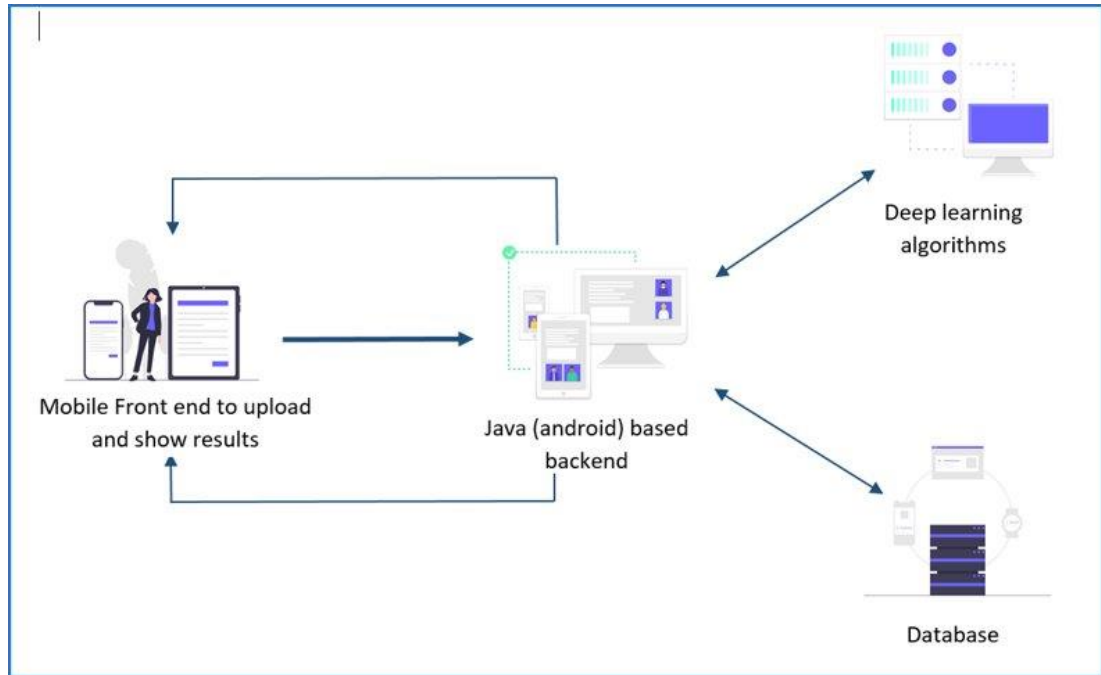


Figure 6: System overview diagram

So, as I mentioned above, the overall research component consists of four main sub components. Those are Mobile frontend to upload and show results, Java (android) based backend, deep learning algorithms and the database component.

First of all, farmer or seller needs to login to the system as a seller. Then from the recycler view of the home page, user needs to select healthiness in order to redirect into the healthiness component. Then by clicking check health button user can move onto the camera component which will predict healthiness levels for potatoes and onions. User only needs to hold the potato or onion to the camera, and the level of healthiness will be displayed at the bottom of the interface. User also can click on 'HEALTHINESS' button as there will be an option on a popup to save the specific healthiness crop with the name of the crop. Later user can refer the savings by going through the view health button, and it will be displayed as a list view with date and time as well. The home page which will be the page to choose healthiness component and the interface that divides two sections is attached below as figures.

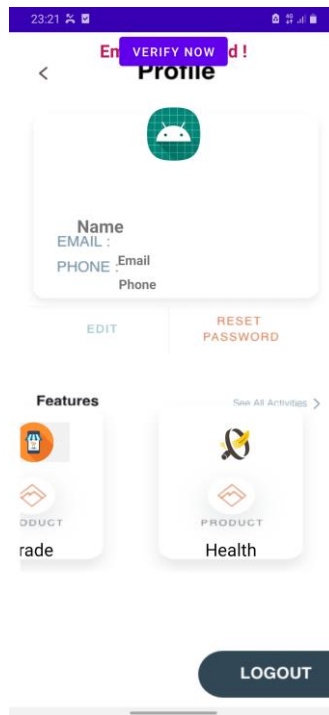


Figure 7: Home page interface

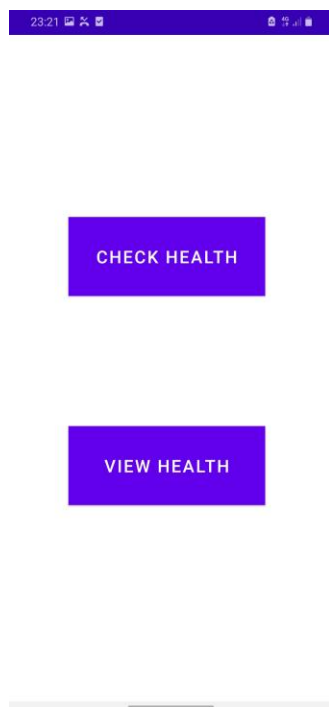


Figure 8: Two buttons interface

4.1.2.2. Android Implementation Methodology

As I mentioned above, our system is developed using android technology in order to support our users like farmers, sellers and buyers in user friendliness and easiness to use. In thinking of the user friendliness, I was able to implement the camera component to deliver the result of healthiness level just by holding the phone camera into the desired crop. In order to create the camera component, I implemented the 'OnImageAvailableListner' into the 'CameraActivity'. The camera services are defined as a user-feature as well in the Manifest file. And also in order to access the camera using the mobile device, the permission of the user is required.

In order to achieve the requirement of camera access, I used the user-permission of the camera access in the Manifest file as well as using android.camera.

```
<!-- Camera Features -->
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />

<uses-permission android:name="android.permission.CAMERA" />
```

Figure 9: Camera features in Manifest

And CAMERA_SERVICE is being used from context to get the camera service in the camera activity. Through camera characteristics, the facing lens is detected and avoids the use of the front facing camera as well. 'CameraAccessException' is used to handle the exception of accessing the camera services.

```

final CameraManager manager = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
try {
    for (final String cameraId : manager.getCameraIdList()) {
        final CameraCharacteristics characteristics = manager.getCameraCharacteristics(cameraId);
        //Not using front facing camera
        final Integer facing = characteristics.get(CameraCharacteristics.LENS_FACING);
        if (facing != null && facing == CameraCharacteristics.LENS_FACING_FRONT) {
            continue;
        }
        final StreamConfigurationMap map = characteristics.get(CameraCharacteristics.SCALER_STREAM_CONFIGURATION_MAP);
        if (map == null) {
            continue;
        }
        useCamera2API =
            (facing == CameraCharacteristics.LENS_FACING_EXTERNAL)
            || isHardwareLevelSupported(
                characteristics, CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL_FULL);
        return cameraId;
    }
} catch (CameraAccessException e) {
}

```

Figure 10: Camera services

As I mentioned above, user permission is required to use the camera in the mobile device. After entering CAMERA as a user-permission into the Manifest file, a method called 'requestPermission()' is written to check whether user has given the permission to access the camera. If user has not given permission to access the camera, a toast message containing: 'Camera permission is required' will pop up towards user.

```

private void requestPermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (shouldShowRequestPermissionRationale(PERMISSION_CAMERA)) {
            Toast.makeText(
                context, CameraActivity.this,
                text: "Camera permission is required",
                Toast.LENGTH_LONG)
                .show();
        }
        requestPermissions(new String[] {PERMISSION_CAMERA}, PERMISSIONS_REQUEST);
    }
}

```

Figure 11: Camera permission

Like user permission is required to access the camera through the app, a level of hardware support is required from the device to achieve the camera objective. The supported information of the level of hardware is gathered through the camera characteristics object which was created when accessing the camera service in the beginning. After gathering the information supported hardware level of the device, it is compared with the required level of hardware to run the camera component. In case of the legacy level is not achieved the required level, the numerical sort can be used. Also the package manager permission is need to be granted to in equal to camera permission to access the camera through the app.

After getting the permissions, it is all set to launch the camera. But before launching the camera, the camera fragment is need to be set according to a specific frame size. In order to do that, first camera Id is take from a method that includes the camera services. And a new instance is created from the camera connection fragment. Then using `onPreviewSizeChosen` method, the preview height and the preview width is need to be set. And using `getFragmentManager` core function the fragment transaction is started.

```
protected void setFragment() {
    String cameraId = chooseCamera();
    Fragment fragment;
    CameraConnectionFragment camera2Fragment =
        CameraConnectionFragment.newInstance(
            new CameraConnectionFragment.ConnectionCallback() {
                @Override
                public void onPreviewSizeChosen(final Size size, final int rotation) {
                    previewHeight = size.getHeight();
                    previewWidth = size.getWidth();
                    CameraActivity.this.onPreviewSizeChosen(size, rotation);
                }
            },
            imageListener: this,
            getLayoutId(),
            getDesiredPreviewFrameSize());
    camera2Fragment.setCamera(cameraId);
    fragment = camera2Fragment;
    getFragmentManager().beginTransaction().replace(R.id.container, fragment).commit();
}
```

Figure 12: Set fragment

After all permissions have granted through the application, a preview of the camera can be started. The 'CameraConnectionFragment' activity is intended after permission granting. Mainly there are three functions regarding the state of the camera device. Those are 'onOpened', 'onDisconnected' and 'onError'. 'OnOpened' is the method that calls the camera API to launch a camera preview to capture the image. It created a new camera capture session for the camera preview. And the preview can be displayed to the user using 'previewRequestBuilder'. Then from the camera Id, generated from 'onOpened' method, open camera method is fired and by creating a new camera manager object that taken from the camera service the camera is launched.

In the camera connection fragment activity, auto focusing of the camera also is handled. As I mentioned above, there is a function called createCameraPreviewSession that creates a new camera capture session for the camera preview.

```
// Auto focus
previewRequestBuilder.set(
    CaptureRequest.CONTROL_AF_MODE,
    CaptureRequest.CONTROL_AF_MODE_CONTINUOUS_PICTURE);
// Flash to be enable when necessary
previewRequestBuilder.set(
    CaptureRequest.CONTROL_AE_MODE, CaptureRequest.CONTROL_AE_MODE_ON_AUTO_FLASH);
```

Figure 13: Set autofocus

Using the preview request builder service, I defined both autofocus mode and auto focus with continuous picture. And using above function, flashing also set when necessary. Because of these two features, farmers and sellers are able to capture the image of the crop very easily without any trouble on focusing or in low lite.

The deep learning model which needs to predict the level of healthiness is positioned inside the assets folder in the project folder structure. Then there is a classifier class

to access that model with the labels. To access both model and the labels text file, two methods are implemented.

```
@Override
protected String getModelPath()
{
    return "health_unquant.tflite";
}

@Override
protected String getLabelPath()
{
    return "health_labels.txt";
}
```

Figure 14: Get model and labels

The result which are generated from the model are shown in the bottom sheet which is positioned in the camera interface itself. Recognition value text view is used to set the value into that bottom sheet.

```
protected void showResultsInBottomSheet(List<Recognition> results) {
    if (results != null && results.size() >= 2) {
        Recognition recognition = results.get(0);
        if (recognition != null) {
            if (recognition.getTitle() != null) recognitionTextView.setText(recognition.getTitle());
            if (recognition.getConfidence() != null)
                recognitionValueTextView.setText(
                    String.format("%.2f", (100 * recognition.getConfidence())) + "%");
        }
        recognition1TextView.setText("Made by Ithamal Wijetunge.");
    }
}
```

Figure 15: Set result values

4.1.2.3. Deep Learning Model Implementation Methodology

According to the literature survey I have done during the proposal period and further, I have found out that this research component is a huge task which can be done by all four members as a single project. Because of the wide range of the component, I have narrowed out the task by moving to image classification from image processing methods. In order to complete the component using image classification itself, there was a mandatory need of huge dataset which suits to my component as well.

a) Dataset Preparation

My research component is to develop a platform to address a level of healthiness for some specific crop in the Mahaweli project. So, I needed to find out some most cultivating crops by Mahaweli farmers. According to the Mahaweli authorized documentation we all chose potato, onion and paddy as our research data. Because of that, I used potato and onion as data to my research component as well. In gathering the dataset, I was not forced to use the data from internet as those image datasets were not that much accurate with the type of crops that cultivated by Mahaweli farmers. In Sri Lanka there is a whole new level of crop cultivation other than foreign countries. Because of that, I didn't use any image data from the internet of any other media source. And, there wasn't any image dataset of the crops they cultivate maintained by the Mahaweli authority as well. Because of this situation, I had to seek out local potatoes and onion for my research component. Any other local farmers' vegetables are much similar to the Mahaweli cultivated vegetables. It was a huge task to find potatoes and onions with different healthiness levels in grocery shops. Anyhow, I bought many potatoes and onions with four different level of healthiness from each of them. Then I captured photos of those potatoes and onions and created more than 3000 images for the training dataset by four levels of healthiness from each one and gathered enough images for the test and valid datasets as well.

b) Training the model

For python coding and training the deep learning model, I used jupyter notebook associated with anaconda navigator. First I created the variable that assigns train, valid and test paths. So, it will point to the location disk where are different datasets were positioned.

In order to train the deep learning model, first some packages were needed to be imported. Those are numpy, Tensorflow, Keras, Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D, Adam optimizer, image categorical_crossentropy, ImageDataGenerator, Sequential, confusion_matrix, itertools, matplotlib.pyplot etc.

```
In [1]: import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from sklearn.metrics import confusion_matrix
import itertools
import os
import shutil
import random
import glob
import matplotlib.pyplot as plt
import warnings
from PIL import Image
```

Figure 16: Python imports

In implementing the deep learning model, I used the Keras Sequential Model. I needed to put the data in a format that model expects when train the model. When using Keras sequential model the model receives the data whenever the fit function is called. Because of that, the image dataset were put into a format of a Keras generator. First, the train, valid and test batches were created and set them equal to flow from directory followed by image data generator which returns a directory iterator. Basically it creates batches of data from the directories where the datasets aside.

And the batches of data are able to pass to the sequential model using the fit function. To the image data generator, I specified the pre-processing function and set that equal to 'tf.keras.applications.mobilenet.preprocess_input'. This is a function that applies some type of pre-processing on the images before they get passed to the network that uses. By this, I process the images in the dataset in the same format as which images that get passed to the vgg16 model over process.

In the flow from directory function the directory is equals to the path that defines before where dataset locates in the disk. The target size is equal to 100x100. This is the height and width that I wanted the potatoes and onions to be resized. Then the classes were specified which are just the classes for the potential labels of my dataset and the batch size is set to ten.

The exact same thing is done towards the validation set and the test set. The only difference is, in test batch, shuffle parameter is equaled to false. That is because, whenever the test batch is used later for inference to get the model to predict on images of potatoes and onions and after training the validation has been completed, the prediction results is needed to be shown in a confusion matrix. In order to do that, it need to be able to access the unshuffled labels for the test set.

```
In [37]: train_path = 'E:/potato-vs-onion/train'
        valid_path = 'E:/potato-vs-onion/valid'
        test_path = 'E:/potato-vs-onion/test'

In [38]: train_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input) \
        .flow_from_directory(directory=train_path, target_size=(100,100), classes=['Onion - 25% Health', 'Onion - 50% Hea
        valid_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input) \
        .flow_from_directory(directory=valid_path, target_size=(100,100), classes=['Onion - 25% Health', 'Onion - 50% Hea
        test_batches = ImageDataGenerator(preprocessing_function=tf.keras.applications.mobilenet.preprocess_input) \
        .flow_from_directory(directory=test_path, target_size=(100,100), classes=['Onion - 25% Health', 'Onion - 50% Hea
```

Figure 17: Data preprocessing

Then the function `plotImages` is defined that use to plot the images from the train batches that obtained above. This function is directly taken from tensor flow's website. Next the defined function is used to plot the images from the batches defined earlier and print the corresponding labels for those images.

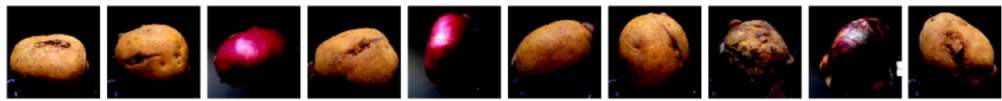
After plotting the images, there can be seen, what s batch of training data looks like. In there we can see that the color data has been a little bit distorted and that is due to the pre-processing function that called earlier to preprocess the batch images. It skewed the RGB data in some way.

Below the plotted images, the corresponding for the data can be seen. Those are one hot encoded vectors that represent specific healthiness level of potato or onion. Each place that '1' occurs represent each level of healthiness of a crop (the class for the healthiness level).

```
In [40]: def plotImages(images_arr):
          fig, axes = plt.subplots(1, 10, figsize=(20,20))
          axes = axes.flatten()
          for img, ax in zip( images_arr, axes):
              ax.imshow(img)
              ax.axis('off')
          plt.tight_layout()
          plt.show()

In [41]: plotImages(imgs)
          print(labels)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



```
[[0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]]
```

Figure 18: Plot images

To create the convolutional neural network, Keras sequential model is taken as the reference. The sequential model can include number of layers as developer wishes. In my model creation, I included six layers into the sequential model.

The first layer to the model that I passed is a Conv2D layer. This is the standard convolutional layer that accepts image data and to this layer I arbitrarily set the filter value equal to 32 with a kernel size of 3x3. Even though the choice of 32 for the filter value is bit arbitrary, the kernel size of 3x3 is a very common choice for image data. And this Cnv2D layer is followed by the 'relu' activation function and the padding is specified as same. It means that the images will have zero padding to outside that the dimensionality of the images aren't reduced after the convolution operations. And the last parameter for the first layer is the input shape of the data. This Conv2D is actually the first hidden layer as the input layer is made up of the input data itself. And shape of the input data should be mentioned there. As we set the target size parameter in data pre-processing to 100x100, we use that here as well as 100x100x3. The last 3 is regarding the color channels since the images are in RGB format. Then the first convolutional layer is followed by a max pooling layer where the pool size is set to 2 by 2 and strides by 2. This is to cut the image dimensions in half.

After the max pooling layer, another convolutional layer is added is almost same as first convolutional layer excepting the input shape parameter as it is only specified in the firsts hidden layer. And filters is set to 64 instead of 32. Again 64 is an arbitrary choice, but the general rule of increasing functions as layers go until later layer of the network is common practice. Then the second convolutional layer is followed by another max pooling layer identical to the first one.

Then I flatten all of the passing data through layers into a one dimensional tensor before passing it to the dense output layer which has nodes corresponding to the classes that declared about according to the healthiness levels of the potatoes and onions. And the output layer is followed by the softmax activation function which gives probabilities for each corresponding output from the model. After that we can visualize the summary of the created model.

```
In [42]: model = Sequential([
    Conv2D(filters=32, kernel_size=(3, 3), activation='relu', padding = 'same', input_shape=(100,100,3)),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'),
    MaxPool2D(pool_size=(2, 2), strides=2),
    Flatten(),
    Dense(units = 8, activation='softmax'),
])
```

Figure 19: Sequential model

```
In [43]: model.summary()
```

Model: "sequential_6"

Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 100, 100, 32)	896
max_pooling2d_12 (MaxPooling)	(None, 50, 50, 32)	0
conv2d_13 (Conv2D)	(None, 50, 50, 64)	18496
max_pooling2d_13 (MaxPooling)	(None, 25, 25, 64)	0
flatten_6 (Flatten)	(None, 40000)	0
dense_6 (Dense)	(None, 8)	320008

=====
 Total params: 339,400
 Trainable params: 339,400
 Non-trainable params: 0

Figure 20: Model summary

The model is built at to this point. The model can be prepared for training by calling 'model.compile'. The optimizer is set as equal to the 'Adam' optimizer with a learning rate of 0.0001. To measure loss I used categorical_crossentropy and the accuracy as the metrics to be able to judge the model performance. Categorical_crossentropy is used because I can't use binary_crossentropy as there are more than two outputs. Using categorical_crossentropy and using softmax activation function for the output layer are done because those are the common approaches when there are more than two classes according to the research I have done.

After compiling the model, the model is trained using 'model.fit' function. To the fit function the first specified parameter is the training data which is stored in the train batches. Then the validation data is specified, which is stored in valid batches. I didn't use validation split, because I already created a validation set separately before fitting the model. Then epochs is equaled to 10 as I needed to run it 10 times. And the verbose is equaled to 2, because most verbose output can be seen during training.

The reason why I didn't specified 'y' is, when data is stored as generator as I done in early stages, the generator itself contains the corresponding labels. After finishing the result, by tenth epoch, the accuracy of the training set has reached 100% but the validation accuracy is calculated as around 98%. As a whole with the summary of the trained set, by the tenth epoch model has reached a good position to predict results.


```
In [44]: model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy')

In [45]: model.fit(x=train_batches, validation_data=valid_batches, epochs=10, verbose=2)

Epoch 1/10
160/160 - 17s - loss: 0.8460 - accuracy: 0.7362 - val_loss: 0.2473 - val_accuracy: 0.9525
Epoch 2/10
160/160 - 16s - loss: 0.1173 - accuracy: 0.9837 - val_loss: 0.0679 - val_accuracy: 0.9950
Epoch 3/10
160/160 - 16s - loss: 0.0364 - accuracy: 0.9994 - val_loss: 0.0323 - val_accuracy: 1.0000
Epoch 4/10
160/160 - 16s - loss: 0.0148 - accuracy: 1.0000 - val_loss: 0.0153 - val_accuracy: 0.9975
Epoch 5/10
160/160 - 16s - loss: 0.0082 - accuracy: 1.0000 - val_loss: 0.0095 - val_accuracy: 1.0000
Epoch 6/10
160/160 - 16s - loss: 0.0045 - accuracy: 1.0000 - val_loss: 0.0077 - val_accuracy: 1.0000
Epoch 7/10
160/160 - 17s - loss: 0.0032 - accuracy: 1.0000 - val_loss: 0.0055 - val_accuracy: 1.0000
Epoch 8/10
160/160 - 17s - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0060 - val_accuracy: 0.9975
Epoch 9/10
160/160 - 17s - loss: 0.0017 - accuracy: 1.0000 - val_loss: 0.0055 - val_accuracy: 0.9975
Epoch 10/10
160/160 - 18s - loss: 0.0013 - accuracy: 1.0000 - val_loss: 0.0045 - val_accuracy: 0.9975

Out[45]: <tensorflow.python.keras.callbacks.History at 0x166cdb2a2e0>
```

Figure 21: Training the model

After training the model, the thing I had to do is test the model using test data. The test results will be explained in the testing and results section and the implementation part of the testing will be covered here.

First thing I had done was, get a batch of test data from the test batches. And this also can be plotted using plotImages function that I mentioned above. And the section test_batches.classes is implemented to get a array that has all of the corresponding labels for each image in the test set. Because we pass the shuffle parameter into false in the test set earlier, I was able have the one-one direct mapping from the unshuffled

labels to the test dataset and if true was passed to shuffle parameter of test dataset every time a batch was generated, then I wouldn't be have the correct mapping between labels and samples. I cared about having the correct mapping because later after I get the predictions from the model, I had plotted the predictions into a confusion matrix and so I wanted the corresponding labels that belongs to the samples in the test set.

```
In [47]: test_batches.classes
Out[47]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2,
                2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3,
                3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4,
                4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5,
                5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
                6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
                7, 7, 7, 7, 7, 7])
```

Figure 22: Classes of test batches

The next step is to go ahead and obtain the predictions by calling `model.predict`. To the parameter 'x' I had specified the test batches of all the test dataset. And the verbose is chosen to be 0 to get no output whenever the predictions are run. The predictions can be printed as the rounded predictions of the model. And again the predictions can be passed to the plot confusion matrix and view the test results in a more familiar way. These parts will be covered in the testing and results sections.

A tool named confusion matrix is used here to visualize the results. The confusion matrix is created using `confusion_matrix` function from `scikit-learn`. For the first parameter I have passed the true labels using test batches classes and for the predicted label parameter, I have passed the predictions from the model. And the `argmax` is used to pass in the index of where the most probable prediction was from the prediction list.

Next the plot confusion matrix function is implemented which is directly taken from `scikit-learn`'s website. This allowed to plot the confusion matrix.

Then the class indices are declared as the 8 classes I have used for 4 level of healthiness used for each potato and onion. Using them the plot labels are defined for the confusion matrix.

```
In [50]: cm = confusion_matrix(y_true=test_batches.classes, y_pred=np.argmax(predictions, axis=-1))

In [51]: def plot_confusion_matrix(cm, classes,
                                normalize=False,
                                title='Crop Identification confusion matrix',
                                cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

In [52]: test_batches.class_indices

Out[52]: {'Onion - 25% Health': 0,
          'Onion - 50% Health': 1,
          'Onion - 75% Health': 2,
          'Onion - 100% Health': 3,
          'Potato - 25% Health': 4,
          'Potato - 50% Health': 5,
          'Potato - 75% Health': 6,
          'Potato - 100% Health': 7}
```

Figure 23: Plot confusion matrix define

4.1.3. Gantt Chart

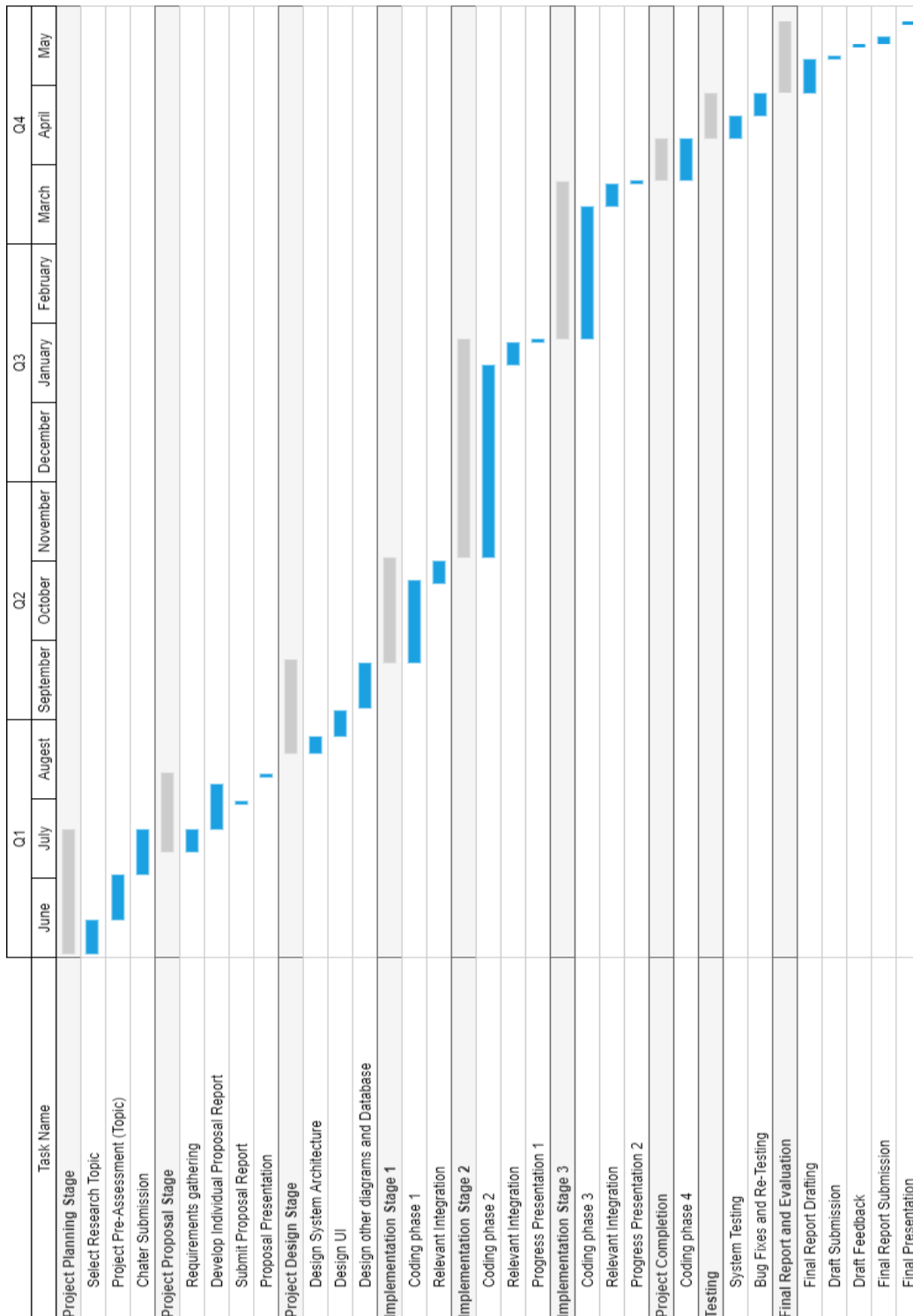


Figure 24: Gantt chart

4.2. Commercialization

In this component, I was able to develop and implement a successful mobile application feature to measure a level of healthiness of specific crops just by holding the mobile camera towards the desired crop.

As I mentioned in the introduction, in agriculture sector measuring the correct level of healthiness is very much important as the prices for those crops is solely decided according to the healthiness of that crop. Currently the job is done by just measuring the healthiness of that crop using human eye level in most of the developing countries. So, this research component feature is a very useful feature that will help to entire agriculture sector, farmers and sellers to measure the healthiness of their crops and categorize them into different criterias that later when selling they can assign prices according to those healthiness criteria.

After successful development of this whole app, the Mahaweli farmers and sellers will use this application as an intelligent application towards their farming and marketing strategy. So, they can get use of this component's feature and improve their quality of selling by measuring correct healthiness of their crops to assign prices to them.

For a specific time period, we can let the Mahaweli farmers and sellers to use the intelligent application for their day to day basis and we can get a clear idea that whether the app is doing its job properly as a prior to commercial it to the outer farmers and sellers as well. So, if this application with especially the healthiness feature is successful among the Mahaweli farmers and sellers, we can step up to next step to commercialize to other Sri Lankan farmers and sellers as well.

After the deployment of the application towards the farmers and seller of the entire country, we can get a clear knowledge about how they use the application and how the application is needed to be improved. And later, we can improve the things that needs to be improved and use this agriculture related application especially measuring the healthiness of the crops. Then we hopefully can successfully commercialize the intelligent application to the globe as well.

5. IMPLEMENTATION AND TESTING

5.1. Implementation

The entire implementation is done using best practices and coding standards. As I mentioned in above sections, the whole application is implemented using latest Android (java) in Android Studio. Apart from the android development, machine learning and deep learning models have been trained using python.

- Used Technologies
 - Mobile – Android with Java
 - Deep Learning - Python, Tensorflow, Tensorflow lite, Keras
 - Database – Firebase
- Used Development Environments
 - Mobile – Android Studio
 - Deep Learning – Anaconda Navigator, Jupyter Notebook

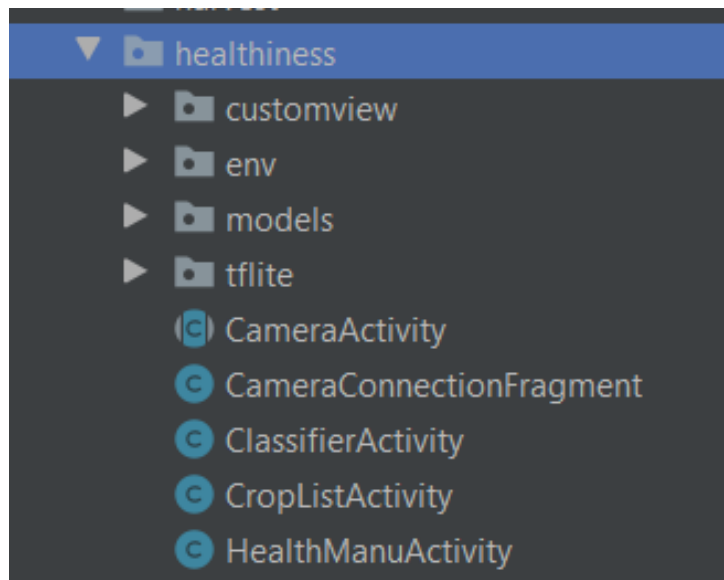


Figure 25: Android folder structure

When development is being done in Android studio, all the members have done their component inside a specific package. As an example, to this research component healthiness measuring, I have created a healthiness package that includes all java classes. Apart from the classes that connected with specific component there are other activities implemented such as language selection for main three languages, login, selecting the user is either a buyer or a seller, and the home page that navigates to all four sub components. These activities and other relevant activities are sitting alongside with the sub component packages.

The layouts are designed inside the resources folder including the healthiness component layouts. And the deep learning model which has trained to predict the level of healthiness is positioned inside the assets folder.

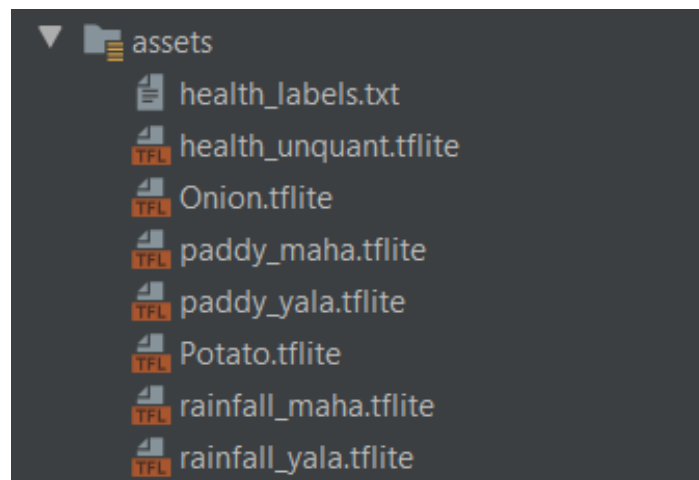


Figure 26: Assets folder

There are set of dependencies which we have used in order to achieve specific features when using android. Some of the main dependencies are androidx, google with android material, google firebase, Tensorflow lite etc.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.firebase:firebase-auth:20.0.1'
    implementation 'com.google.firebase:firebase-firestore:22.0.0'
    implementation 'com.google.firebase:firebase-storage:19.2.0'

    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:26.8.0')

    // Declare the dependency for the Realtime Database library
    implementation 'com.google.firebase:firebase-database'

    //----- Added by Ihamal -----
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    //----- Tflite
    implementation 'org.tensorflow:tensorflow-lite:0.0.0-nightly'
    implementation 'org.tensorflow:tensorflow-lite-gpu:0.0.0-nightly'
    implementation 'org.tensorflow:tensorflow-lite-support:0.0.0-nightly'

    implementation 'com.squareup.picasso:picasso:2.71828'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'androidx.recyclerview:recyclerview:1.1.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.android.volley:volley:1.1.1'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'de.hdodenhof:circleimageview:3.1.0'
    implementation 'org.jetbrains:annotations:15.0'
    implementation 'com.google.firebase:firebase-database:19.7.0'
    implementation 'org.tensorflow:tensorflow-lite-support:0.1.0-rc1'
    implementation 'org.tensorflow:tensorflow-lite-metadata:0.1.0-rc1'

}
```

Figure 27: Gradle dependencies

There are some version that we used in the gradle as well. Those are, compile SDK version as 30, minimum and target SDK versions as 23 and 30 respectively and Java version as 8. Apart from android we have used python 3.9 to implement the machine learning and deep learning models. These are the main implementation methods we have used in order to develop our intelligent application to the Mahaweli farmers and sellers. From the next section I have mentioned the testing I have done towards the android application and the deep learning model prediction testing as well.

5.2. Testing

In the research component of measuring healthiness levels, the main testing that needed to be done is the deep learning model testing. The deep learning model testing is first done from the python coding itself in the jupyter notebook. As I mentioned in the implementation methodology, first thing I had done was, get a batch of test data from the test batches.

By calling `model.predict` we can obtain the prediction of the trained deep learning model. To the `x` parameter I had specified the test batches of all the test dataset. And the `verbose` is chosen to be 0 to get no output whenever the predictions are run.

```
In [48]: predictions = model.predict(x=test_batches, verbose=0)

In [49]: np.round(predictions)

Out[49]: array([[1., 0., 0., ..., 0., 0., 0.],
                [1., 0., 0., ..., 0., 0., 0.],
                [1., 0., 0., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 0., 0., 1.],
                [0., 0., 0., ..., 0., 0., 1.],
                [0., 0., 0., ..., 0., 0., 1.]], dtype=float32)
```

Figure 28: Model prediction testing

The predictions can be printed as the rounded predictions of the model. The way that we can read the preview of the prediction is, first one of the arrays is a prediction for a single sample. If we look at the first sample of the test set as above figure displays, wherever there is a one with each prediction is the index for the output class that had the highest probability from the model. In this case, the 0th index has the highest probability. So, we can get finalized that the label that the model predicted for this first sample was a zero, because there is a 1 in the 0th index. And also the predictions can be passed to the plot confusion matrix and view the test results. This part will cover in the results section of the document.

6. RESULTS AND DISCUSSION

6.1. Results

Mainly results can be divided into two sections. Those are the results comes out from the measuring level of healthiness feature of the intelligent application, and the test results came out from the trained deep learning model. The third objective of this research component is to complete the application by creating deep learning model to predict healthiness using the camera component which was integrated into this feature component. As it is developed towards the farmers and sellers of Mahaweli market and according to the requirement of Mahaweli authority, the system can predict four levels of healthiness for potatoes and onions. Below 2 figures will show the results for onion and for potatoes.

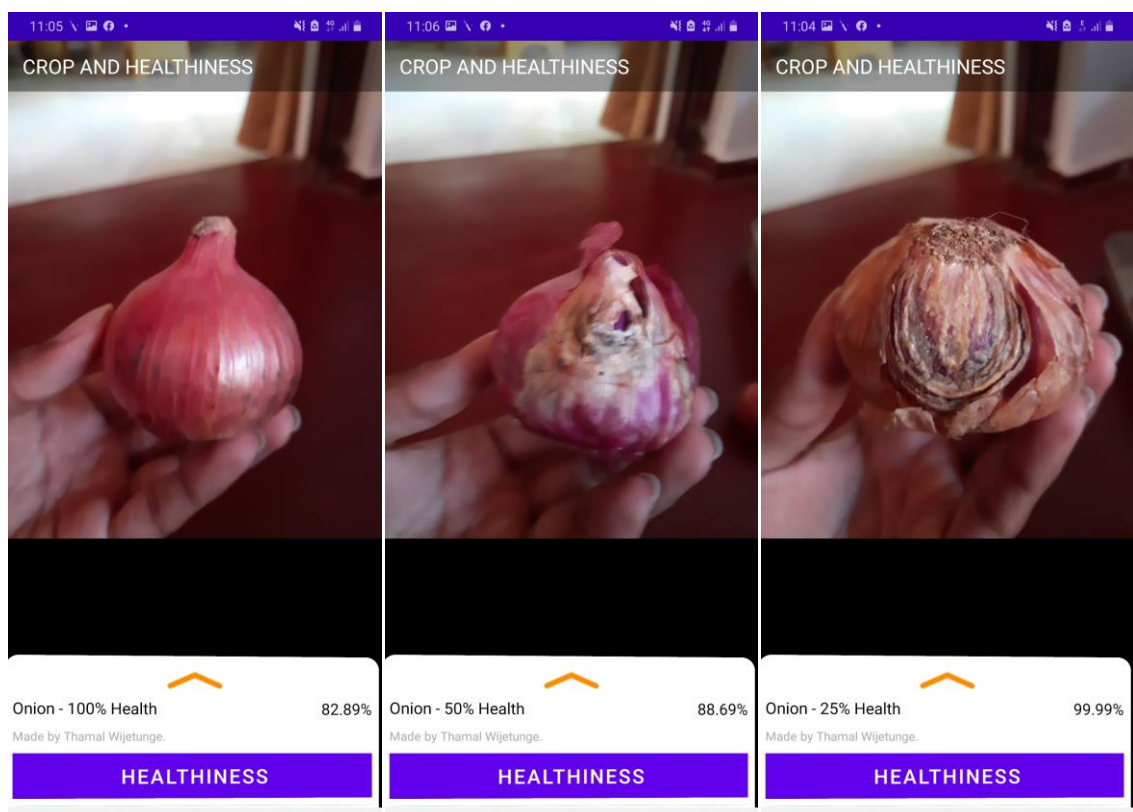


Figure 29: Onion healthiness predict interface

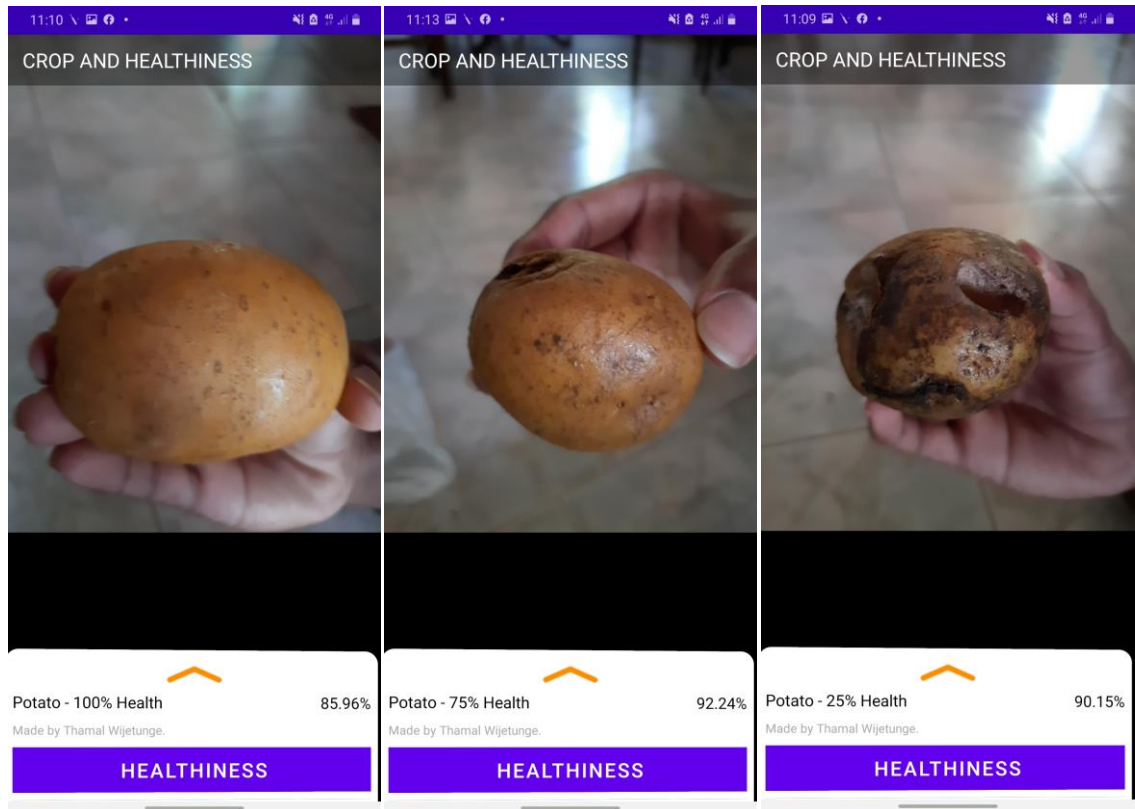


Figure 30: Potato healthiness predict interface

According to above figures, the results are shown to the users in the bottom part of the interface. Users can see whether the crop is potato or onion and the level of healthiness for that crop out of four levels as 25%, 50%, 75%, 100%. The novelty feature of this component is that the user don't need to wait after clicking a button or do something to get the result as the healthiness. User can simply hold the camera above the desired crop and it will automatically predict the crop with the level of healthiness real-time. This resulting feature will help a lot for Mahaweli farmers and sellers by saving their time. Also the UX part that is user friendliness and easy to use is clearly depicted through this feature.

Also, in any case that farmer or seller need to verify the healthiness of that specific crop, he/she can simply click the 'HEALTHINESS' button which positioned under the predicting label.

When user clicks the 'HEALTHINESS' button, it pops up a simple modal that shows the information about what the crop is and the level of healthiness the crop has. Further, there are two buttons in the modal. By clicking the save button user can save that crop with the level of healthiness including the date and the time as well. For that, firebase database system is used as mentioned in the methodology. The pop up modal is shown like the below figure.

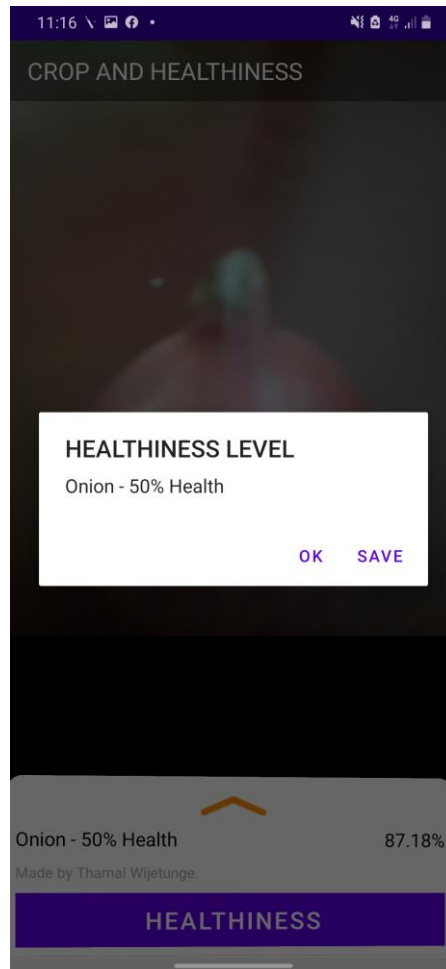


Figure 31: Modal pop up interface

If user doesn't need to save the crop with healthiness for future needs, he/she can just click on 'OK' button to close the modal.

The extra feature I have provided in this research component is to save the crop and the level of healthiness of that crop to database for future purposes. As I mentioned above, farmer or seller can save the information by clicking the button underneath the camera fragment and then clicking on the save button which is positioned on the pop up modal.

The farmer or seller also can view the saved data from a different interface. It can be navigated from the two button interface by clicking on the 'VIEW HEALTH' button. Click that button will redirects user to the view interface of the crop and healthiness information. The list of data are looped into a list view and it is shown as the figure below.



The screenshot shows a mobile application interface with a purple header bar containing the title 'Crop Healthiness Details'. Below the header is a list of ten entries, each representing a crop health check. Each entry includes a date and time, the crop name, and a health percentage. The list is separated by thin horizontal lines. At the bottom of the screen, there is a light gray bar with a small horizontal line in the center, likely representing a home indicator on an iPhone.

Crop Healthiness Details		
Sun Mar 28 2021 11:31:52	Onion	50%
Sun Mar 28 2021 11:32:52	Onion	50%
Sat Apr 03 2021 12:27:08	Onion	100%
Sat Apr 03 2021 12:28:59	Potato	25% Health
Sat Apr 03 2021 12:33:21	Potato	25% Health
Sun Apr 25 2021 16:24:44	Potato	25% Health
Sun Apr 25 2021 16:30:12	Onion	100% Health
Sun Apr 25 2021 16:39:03	Potato	25% Health
Sun Apr 25 2021 18:14:18	Potato	50% Health
Sun Apr 25 2021 18:14:42	Onion	25% Health

Figure 32: List view interface

Not only the results that mentioned above, but there are test results that given from the deep learning model predictions as well.

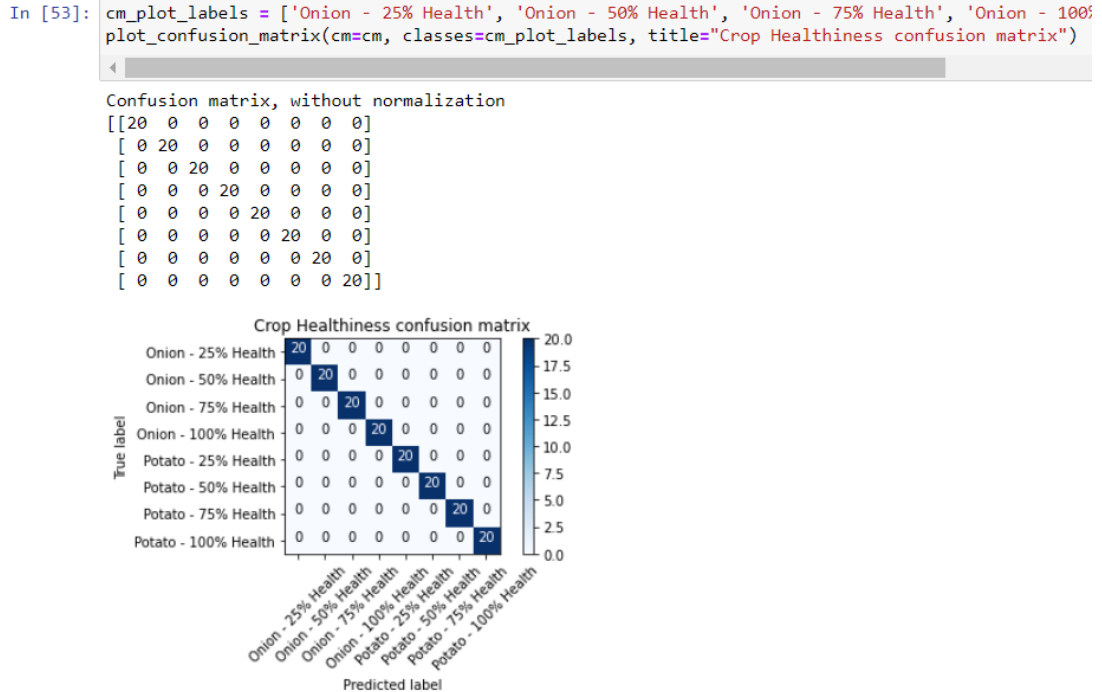


Figure 33: Confusion matrix results

As the above figure, in order to plot confusion matrix the plot labels are defined in the correct order. Next ‘plot_confusion_matrix’ function is called with some parameters. First parameter is the confusion matrix itself. Then labels for the confusion matrix is passed along side with a title for the entire matrix.

In the left side of the matrix, the true labels are positioned from top to bottom as the y axis. In the bottom the predicted labels are positioned from left to right. The diagonal which started from top left corner to bottom right corner indicates the correct probability that the model predicts. If any number is positioned outside the diagonal it means that the label is predicted incorrectly.

6.2. Discussion

After implementing all these mentioned methods, finally I was able to complete the research component successfully. The reason I used image classification instead of image processing is the wide range of this component. If using image processing, the single component can be done as a full research project according to the literature surveys I have done. Our sole intention at the beginning was to deliver a complete intelligent agricultural app for the Mahaweli farmers and sellers.

From this research component measuring the levels of healthiness for potatoes and onions Mahaweli farmers and seller now won't have to just measure their crops healthiness by eye level. They just need to log into the system, continue as a seller, redirect to the healthiness checking camera fragment and hold the camera to the desired crop. This helps the farmers and sellers to categorize the crops according to level of healthiness that the application predicts.

We hope that the Mahaweli farmers and sellers get the most of this agricultural intelligent application, especially the healthiness measuring component as it clearly help them to maintain the pricings and selling in a correct manner. We can let Mahaweli farmers and sellers use it for a specific time period and get an idea about allowing the app to use by the farmers and sellers all around Sri Lanka as well.

7. CONCLUSION

This research component is to develop an agricultural intelligent platform to address healthiness of some specific crops. The platform is developed for the farmers and sellers of the Mahaweli market and project. The requirement is implemented as a mobile application using Android with java in android studio. The requirements to the research is gathered from the Mahaweli authority.

Sri Lanka is an agricultural country since the ancient times. From that times, farmers and seller have measured the healthiness of their cultivation by just human eye level. But that method brings lots of problem towards selling their crops to a correct pricing. They cannot even categorize their cultivated crops according to different healthiness levels. That may become a huge disadvantage towards farmers and sellers when selling their crops with low prices to customers. That is the problem and reason to develop a platform to address the healthiness levels for specific crops with a very easy manner. Further, this component of the application is developed to use with ease to the users. Farmers or sellers who uses this application simply need to hold their camera phone by their desired crop and the application will generate a prediction of healthiness with the name of the crop in real-time.

In the development process, the iterative waterfall model has been followed. It is because, at the beginning there isn't any idea about the technologies and the way to achieve the final object. So in case of wrong path, I would be able to correct it and continue the process. The overall system overview is consist with four sub components. Those are mobile frontend to upload and show results, Java (android) based backend, deep learning algorithms and the database component. The technologies I have used are android with java for frontend, Python, Tensorflow, Tensorflow lite, Keras for the purpose of training the deep learning model and firebase for database purposes. And I have used android studio as environment to develop the android part and the jupyter notebook with anaconda navigator to develop the deep learning part.

There are four interface to interact In order to use the healthiness measuring component. Those are the home page that has main access to the healthiness component, the navigation interface to the camera which predicts the healthiness with the crop and the list view which listed out the saved information to future reference and these mentioned two interfaces to measure healthiness and view the list view.

For the deep learning model, an image classification model has been trained. The dataset used for the training purpose is a pure original dataset. To gather the images to the dataset, I have walked into shops and bought potatoes and onions with four different healthiness levels. Even the worst potatoes and onions were used to gather the dataset. However, the final dataset to train the model had more than 3000 training images. The training batch, validation batch and test batch were preprocessed using Keras image data generator. The height and the width was given as 100x100. Then the sequential model has been initialized using 2 convolutional 2D layers, 2 max pooling layers, the flatten layer and the dense layer. Then the model has been compiled including the Adam optimizer and the categorical_crossentropy for measure the loss. After compiling the model, it was pushed to train using fit function with 10 epochs. After training the sequential model, it has been use to predict the results using the test dataset in order to test the accuracy of the trained model. After testing the model with test batches, the result has been plotted using the confusion matrix as I mentioned above. Finally the trained Tensorflow model is converted into a Tflite model using the same python notebook. To use the model in the android project, the model is loaded into the assets folder of the android project folder structure.

After the development is done according the way that mentioned above, farmers and sellers can use the application component in order to measure the healthiness levels of potatoes and onions. After a specific time period, we can gather the feedbacks of the Mahaweli farmers and sellers in order to improve the application towards a more successful agricultural mobile application. Then after we can deploy this agricultural application to all Sri Lankan farmers and sellers in order to achieve their need to measure healthiness of their crops towards selling them to the customers.

8. REFERENCES

- [1] Menale Kassie, Bekele Shiferaw and Geoffrey Muricho, Agricultural Technology, Crop income, and Poverty Alleviation in Uganda. International Maize and Wheat Improvement Center, Nairobi, Kenya. Vol 39, 2011.
- [2] Duan Yan-e, Design of Intelligent Agricultural Management Information System Based on IoT. Beijing University of Agriculture, Beijing, China, 2011.
- [3] K. Koszela, M. Lukomski, W. Mueller, K. Gorna, O. Okon, P. Boniecki, M. Zaborowicz, D. Wojcieszak, Classification of Dried Vegetables using Computer image analysis and artificial neural networks. Poznan Univ. of Life Science, Poland.
- [4] K. Przybyl, K. Gorna, D. Wojcieszak, The recognition of potato varieties using of neural image analysis method. Poznan Univ. of Life Science, Poland.
- [5] Priyanka S. Mankoji, Uttam Patil, A Versatile Mobile Application for Quality Prediction of Wheat Grain using Machine Learning Algorithms. Belagavi Karnataka, India. Vol. 6, Issue 7, July 2007.
- [6] Kavindra Paranage, The Mahaweli Development Project and the ‘rendering technical’ of agrarian development in Sri Lanka. School of Social Sciences, Monash University, Wellington Road, Clayton, VIC, 3168, Australia.
- [7] T Gunda, J T Bazuin, J Nay, K L Yeung, Impact of seasonal forecast use on agricultural income in a system with varying crop costs and returns: an empirically-grounded simulation. Vanderbilt Institute, United States of America.

9. APPENDICES

Turnitin Originality Report Processed on: 14-May-2021 21:46 +0530 ID: 1586097067 Word Count: 8994 Submitted: 1					
20_21-J11_IT17167024 By Thamal Wijetunge	<table> <tr> <th>Similarity Index</th><th>Similarity by Source</th></tr> <tr> <td>5%</td><td> Internet Sources: 3% Publications: 1% Student Papers: 4% </td></tr> </table>	Similarity Index	Similarity by Source	5%	Internet Sources: 3% Publications: 1% Student Papers: 4%
Similarity Index	Similarity by Source				
5%	Internet Sources: 3% Publications: 1% Student Papers: 4%				
2% match (student papers from 09-Nov-2018) Submitted to Sri Lanka Institute of Information Technology on 2018-11-09					
1% match (student papers from 04-Apr-2021) Submitted to National School of Business Management NSBM, Sri Lanka on 2021-04-04					
1% match (student papers from 02-Oct-2020) Submitted to Middlesex University on 2020-10-02					
< 1% match (student papers from 02-Nov-2018) Submitted to Sri Lanka Institute of Information Technology on 2018-11-02					
< 1% match (student papers from 30-Aug-2011) Submitted to University of Sheffield on 2011-08-30					
< 1% match (student papers from 16-Sep-2020) Submitted to University of Sheffield on 2020-09-16					
< 1% match (student papers from 09-Nov-2011) Submitted to Multimedia University on 2011-11-09					
< 1% match () Molia, Sophie. "Report of the protocol for epidemiological surveys : LoA PR 37212 between FAO and CIRAD", s.n., 2007					
< 1% match (Internet from 09-Mar-2019) https://www.nature.com/articles/s41534-018-0118-7?code=76ea30d7-7a47-41a3-a4c9-14eab22bf98d&error=cookies_not_supported					
< 1% match (Internet from 03-Apr-2021) http://dl.lib.mrt.ac.lk/bitstream/handle/123/11521/pre-text.pdf?isAllowed=y&sequence=1					

Figure 34: Turnitin similarity report