

**“E-KETHA”: ENRICHING RICE FARMER’S QUALITY
OF LIFE THROUGH A MOBILE APPLICATION.**

2022-81

Final Report

Salika Madhushanka W.J

B.Sc. (Hons) Degree in Information Technology

Department of Computer Science and
Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

October 2022

**“E-KETHA”: ENRICHING RICE FARMER'S QUALITY
OF LIFE THROUGH A MOBILE APPLICATION.**

2022-81

Final Report

Salika Madhushanka W.J – IT19101620

Supervisor: Mr. Adeepa Gunarathna

Co Supervisor: Ms. Amali Upeka Gunasinghe

B.Sc. (Hons) Degree in Information Technology

Department of Computer Science and
Software Engineering


Sri Lanka Institute of Information Technology

Sri Lanka

October 2022

DECLARATION, COPYRIGHT STATEMENT, AND THE STATEMENT OF THE SUPERVISOR

We declare that this is our own work and this proposal does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgment is made in the text.

Name	Student ID	Signature
Salika Madhushanka W.J	IT19101620	

The supervisor/s should certify the proposal report with the following declaration.

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor:

Date:

Signature of the supervisor:

Date:

ABSTRACT

In our country of Sri Lanka, rice is the most common type of food that is consumed on a daily basis. Due to that rice farmers face a huge amount of stress to supply according to the massive demand. One of the main problems rice farmers are currently facing is the abundance of pests and diseases that affect rice crops. Due to some of these being highly transmittable, proper action has to be taken quickly and efficiently. Since diseases and pests come in various types, identifying and treating them can be difficult for the common farmer. The aim is to develop a mobile application that will help farmers solve this particular problem. The application will use images to conduct image processing to analyze and identify the type of disease, and pests, and video processing can use for pest detection. This will finally allow machine learning and deep learning to provide the proper solution.

Keywords:- machine learning, image processing, deep learning, video processing, rice crop

TABLE OF CONTENTS

DECLARATION, COPYRIGHT STATEMENT, AND THE STATEMENT OF THE SUPERVISOR.....	iii
ABSTRACT	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vii
LIST OF TABLES	vii
1. INTRODUCTION.....	1
1.1 Background	1
1.2 Literature Survey	3
1.2.1 Agrio Mobile Application	3
1.2.2 Crop Farmers Mobile Application	3
1.2.3 Detection of leaf diseases and Wheat Using Image Processing	4
1.2.4 Pest Identification using Image Processing using Neural Network	4
1.2.5 Pest & Crops Mobile application	4
1.3 Research Gap	5
2. RESEARCH PROBLEM	6
3. OBJECTIVES	6
2.1 Main Objectives	6
2.2 Specific Objectives	6
4. METHODOLOGY	8
4.1 Methodology	8
4.1.1 Research Area	9
4.1.3 Design	10
4.1.4 Tools and Technologies	11
4.1.5 Data acquisition	12
4.2 Commercialization aspects of the product	12
4.2.1 Target audience	12
4.2.2 Design of the app	12
4.2.3 Gap in the market	13
4.2.4 Marketing plan	13
4.2.5 Pricing	13

4.2.6 Budget	14
4.2.7 WBS.....	14
4.2.8 Gantt chart	14
5. Testing & Implementation	15
5.1 Implementation	15
5.1.1 Pre-processing	16
5.1.2 Modal	16
5.2 Testing and Maintenance	18
6. RESULTS AND DISCUSSION	20
6.1 Results	20
6.1.1 Detection of pests and providing solutions.....	20
6.1.2 Detection of Diseases and providing solutions:	22
6.2 Research Findings.....	23
6.2.1 Detection of pests and providing solutions.....	23
6.2.2 Detection of Diseases and providing solutions:	23
6.3 Discussion.....	24
6.4 Summary of Each Student's contribution	24
Conclusions.....	25
REFERENCE LIST	26
Glossary	27
Appendices.....	28

LIST OF FIGURES

Figure 1: Where rice pests and diseases do the most damage	1
Figure 2:Production lost.....	2
Figure 3:Pest and disease detection overview.....	8
Figure 4:WBS chart	14
Figure 5:gantt chart	15
Figure 6:Test accuracy	20
Figure 7:Pests prediction output	21
Figure 8:Training accuracy chart.....	22
Figure 9:Diseases prediction output	23

LIST OF TABLES

Table 1:Comparing existing application and our application features.....	5
Table 2:Model accuracy	21

1. INTRODUCTION

1.1 Background

As the most common food in Sri Lanka rice holds a special place in all Sri Lankans' hearts. Especially, for this reason, the rice demand is massive. One of the reasons this demand is not currently met is the destruction of rice crops by diseases and pests.

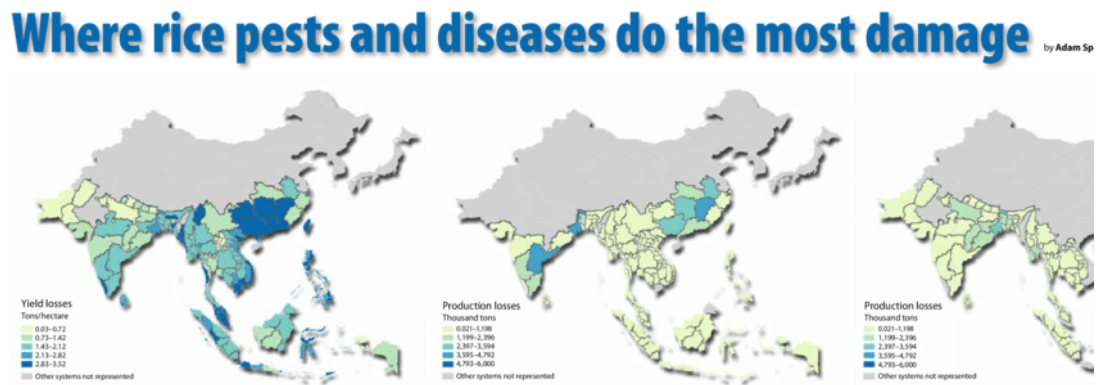


Figure 1: Where rice pests and diseases do the most damage

As it is shown in the above diagram [6] in the country of Sri Lanka, the yield losses due to diseases and pests are mostly in the red zone. Because of this annually 35% of the yield goes to waste.

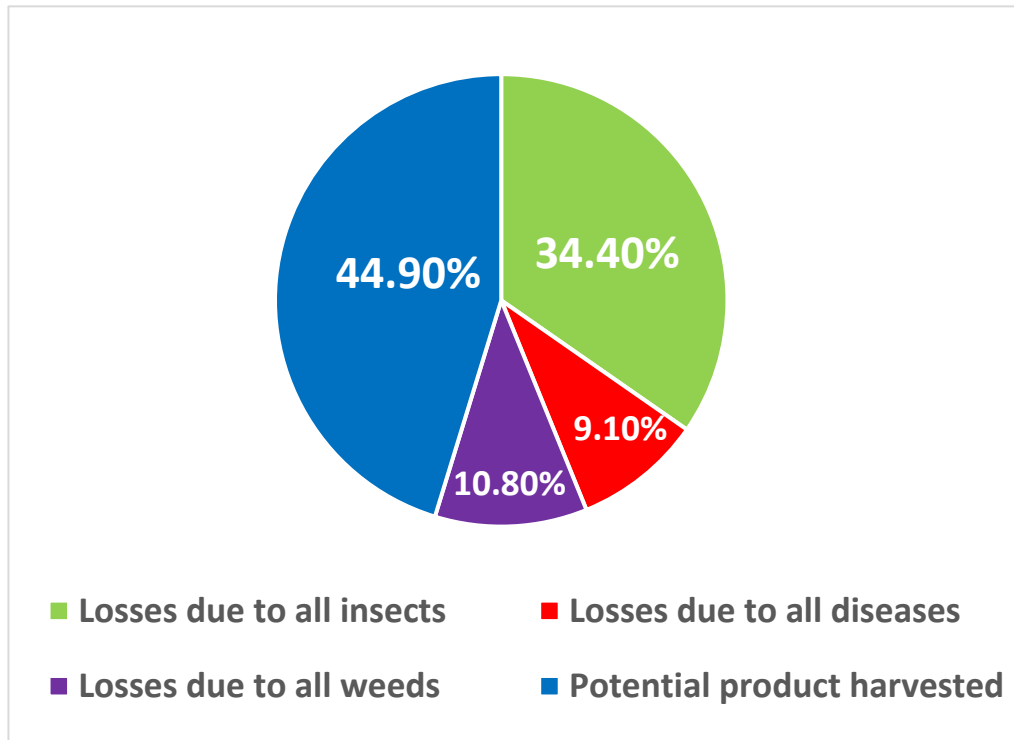


Figure 2: Production lost

By looking at figure 2 [10] it is apparent that 34% of all harvests in Asia are lost due to insects as well as 9.1% of it being lost due to diseases. All of these considered only 45% of the harvest is produced.

If these diseases or pests are not treated properly they can spread quickly to other crops making them unhealthy as well. Since there are many types of diseases and pests is virtually impossible for common farmers to have knowledge about proper pesticides and treatments.

As a solution for all these issues, a mobile application will be proposed.

1.2 Literature Survey

1.2.1 Agrio Mobile Application

This is a widely used plant disease identification application worldwide. He determined the number of diseases related to fruits, vegetables, etc. Agrio is an artificial intelligence-based solution that helps you to identify and treat plant diseases and pests in your fields, farms, and gardens. We offer comprehensive Integrated Pest Management (IPM) protocols to optimize results and reduce treatment costs. It also gives you forecasts and sends you alerts when problems arise in your area. This application allows you to protect vegetables, fruits, flowers, trees, and other plants from damage. When you are uncertain about the results of artificial intelligence algorithms, share your images with a group of agronomists and extensionists you may have. If you have a conclusion, an automatic response will be sent to you in a few seconds. If you choose to present to a team of experts, they will share their opinion based on the photos they upload and give you suggestions to solve the problem. [1]

1.2.2 Crop Farmers Mobile Application

Help the farmer with summary information about crops, fruits, and vegetables. Climatic and Soil Requirements, Avocado, Banana, Beans, Carrot, Jackfruit, Cucumber, Garlic, Irish Potato, Lettuce, Sorghum, Watermelon, Onion, Bell Peppers and Peppers, Pineapple, and Eggplant sour Info explains how. The app also describes the most common causes of pests and diseases, symptoms, how they spread, and prevention and control measures. Where possible, the app will advise on suitable farming methods to control crop pests. This app can be used as a guide for new farmers, or anyone involved in farming around the world. Learn new farming techniques/methods to avoid attacking your crops. It also provides information on best practices to follow to improve farmers' performance in growing these crops. [2]

1.2.3 Detection of leaf diseases and Wheat Using Image Processing

This study, mainly carried out by Rittika Raichaudhuri and Rashmi Sharma, aims to use image processing as the primary technology to detect foliar diseases in the field of wheat production. The former algorithm is used to extract important information from the leaf and the second algorithm is used to detect the disease it is infected with. For image processing and segmentation, the use of the k-means algorithm and convenience filter has been proposed. Pattern recognition is performed by PCA or GLCM and classification by SVM or ANN. [3]

1.2.4 Pest Identification using Image Processing using Neural Network

This study is done by Johnny L. Miranda, B. Gerardo, Bartolome T. Tanguilig, and Sajad Sabzi with the goal of classifying pests in crops. Pest infestation in rice production is a challenging task for crop technicians and farmers. A pest infestation can cause serious losses and also affect the income of farmers. Decisions for pest predictions can be made by estimating the density of farmers. Existing detection techniques for these species involve the use of various traps to detect their presence. In this study, an identification system was developed for the automatic detection of field insect pests. Continuous monitoring by a wireless camera for video recording is done by catching the insect with a sticky trap. Various imaging techniques are used to identify and extract the captured insect. A neural network was used to identify the extracted insect pests. The new automated detection system developed in this study provides reliable detection. [4]

1.2.5 Pest & Crops Mobile application

This application is also widely used for identifying pests and managing crops. Mainly target to identify the wheat pest. Also, have used image processing algorithms to identify the pests. Also, this application provides pest-forecasting information to the user. As well as this application has three main categories for pests identifications. There are Bees in North America, Insect in Canola, and Insects in Wheat [5]

1.3 Research Gap

	Agrio	Crop Farmers	Diseases and Wheat	Pest Identification	Pest & Crop	E-Ketha
Diagnosis of paddy cultivation.	No	Yes	No	No	No	Yes
Identify diseases that are harmful to the cultivation.	Yes	No	Yes	No	No	Yes
Give a brief description of the current situation of the disease.	No	No	Yes	No	No	Yes
Give Details about the treatment of the disease.	Yes	Yes	No	No	No	Yes
Identify pests that are harmful to the cultivation.	No	No	No	Yes	Yes	Yes
Give Details about the treatment of the Pests.	No	No	No	No	Yes	Yes

Table 1: Comparing existing applications and our application features

2. RESEARCH PROBLEM

Rice productivity is something on which the Economy highly depends in Sri Lanka and South Asia. nowadays rice growers are moving away from rice cultivation. The main reason for this is that they cannot afford a reasonable income. This is one of the major problems in the rice cultivation field. The first major problem with paddy is the prevalence of rice diseases. With the discovery of new diseases and ailments day by day, it is difficult for the average farmer to diagnose and treat them. Pests and other unwanted insects that are closely related to diseases also attract crops. These pests can cause disease in the plant in the first place as well as spread the disease. Pests, while not spreading diseases, can contaminate crops for human consumption [7][8][9].

3. OBJECTIVES

2.1 Main Objectives

Introduction of a mobile application to identify diseases and pests in paddy cultivation. The main objective of identifying potential diseases and pests in paddy plants is to create an Android-based mobile application that can analyze leaf changes using the image processing method and detect possible diseases in paddy cultivation. This Android-based mobile application is used to detect leaf diseases with the help of automated algorithms and those pests can detect automated algorithms. This uses the database to identify the most vulnerable sheets and present the most accurate result. In addition, the application provides treatment for diseases and pests.

2.2 Specific Objectives

1. Application Identifies possible diseases by analyzing changes in the rice plant leaf. Rice farmers take an image of the rice leaf and upload it through the app using image processing technology. The application compares the color changes of the

rice leaf with the data uploaded to the database and identifies whether the leaf could be a diseased leaf or not.

2. Application Classifies rice plants according to their disease stage. Once the disease is successfully diagnosed, the leaflet is classified according to its effect. There is a normal, risk, and so on stage.

3. The application will suggest treatments according to the type of disease and the current status of the disease.

4. App will identify the possible pests by analyzing the rice plant. Rice farmers can capture a video/image of the pests and upload it through the app using video processing techniques. The application classifies pests using uploaded data to a database.

5. Once the pests have been successfully determined, the pests are classified according to the database data

6. The application will provide treatment suggestions for identified pests.

4. METHODOLOGY

4.1 Methodology

This section includes detailed descriptions of the techniques and mechanisms employed to correctly plant rice plants. The descriptions include how the software implementation of the project is carried out, what materials and data are needed, and how they will be collected.

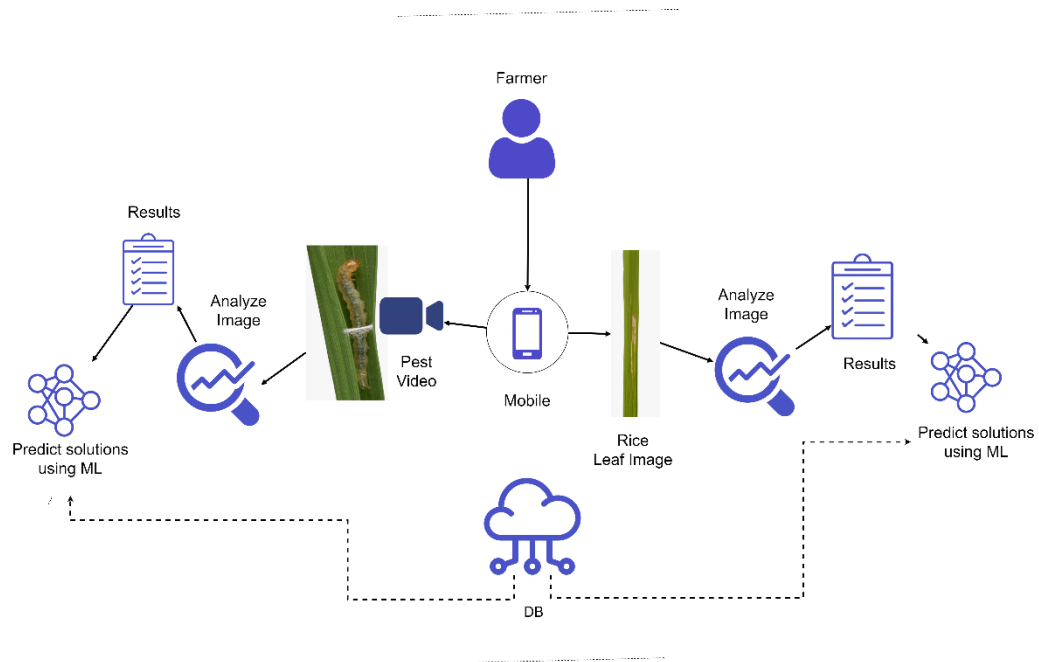


Figure 3: Pest and disease detection overview

Many diseases and pests affected the rice plants. Some are a black blight, leaf spot, leaf scab, gray spot, etc. Many farmers lack the knowledge to identify diseases and pests. So, using data mining techniques and analyzing models prepare a database about the harmful diseases and pests to the cultivation. The training database is retrieved with the information provided by the user. When the user uploads the image of the leaf to the mobile application, the system will detect whether it is healthy or diseased. In the case of pests, the user can upload a video of the pest for it to be identified. If it is identified as a diseases leaf mobile application will show the treatments that are wanting to heal that disease or if it is pests removal methods such as pesticides will be suggested.

For this process identification of the disease must be done in an accurate way. Image processing techniques such as Grayscale, threshold, median filters, and binary images are used for this process to give accurate outcomes. Segmentation is the separation or division of an image into different parts. There are various methods of image resolution, ranging from simple apertures to high-color image segmentation methods. Computers cannot detect objects intelligently, and various techniques for image segmentation have been developed. The process of segmentation is based on different elements in the image. This can be color information, a border, or part of an image. To implement this application, must use generic algorithms for color image segments.

4.1.1 Research Area

When it comes to the research area, four features were identified. Such as Image processing activities, Classification activities, Detection activities, and finally solution prediction. In order to conduct the research, deep learning technology has been taken as the core foundation.

4.1.2 Requirement Gathering and Analyzing

Due to the importance of requirement gathering and analysis, major emphasis was put on this section. Since there is a need for this process to be strictly on the "disease and pests identification and solution finding" part below mentioned approaches were used.

- Reading research papers relevant to the research problem.
- Studying existing systems related to our research area.
- Contacted experts at Rice Research and Development Institute(RRDI),
Bathalagoda.
- Met with Sri Lankan paddy farmers.

To get an idea about the research problem, studying related research papers is a must. The next step was to understand what types of systems already exist, and to see what is lacking and needs improvements. Finally, to see if the proposed solution is viable in the current environment, specialists in the field and traditional farmers were contacted.

4.1.2.1 Functional requirements

- Ability to upload disease imagery.
- Identify disease type.
- Ability to upload pest's video.
- Identify pest type.
- Propose solutions.
- Show proposed solutions.

4.1.2.2 Non-functional requirements

- Reliability
- Accuracy
- Availability
- Performance
- User friendly

4.1.3 Design

The design phase encompasses what is needed for the estimation of hardware and system requirements by the creation of a system architecture, due to the needs and specifications being included. The architecture will entail the components separated into manageable levels according to the respective research project member. In this case, it will be the "identification of pests and diseases then proposing solutions" component.

4.1.4 Tools and Technologies

4.1.4.1 Tools

- Android Studio
 - This is chosen due to it being the primary IDE recommended for Java mobile application development. The user-friendliness coupled with the performance, security, and feature richness also makes this the most suitable option.
- Google collab
 - Since some of the deep learning models require high amounts of computational resources a virtual environment like google collab is most appropriate.
- Google Drive
 - Since google collab is used for the model training, the dataset cannot be stored on personal computers thus google drive is needed for storing the dataset.

4.1.4.2 Technologies

- Deep learning
 - Deep learning is the only solution for image classification and identification tasks such as this. Due to there being no similar prior work, a model has to be created and trained from scratch.
- Models
 - Alexie
 - Alexie was identified to be the best for the mapping of pests.
 - CNN
 - A custom CNN model was identified to be the best for the identification of weeds.

The evidence for this is provided below within the methodology.
- Android Java
 - Since the application is initially targeted toward android devices, to provide the smoothest experience possible native android java is used.

- **Firestore**
 - Due to the application requiring a real-time online connection to the database firestore is chosen as the primary database. Since the data set mostly consists of images the need for a document-based database is further insinuated.
- **Python**
 - For the machine learning and deep learning parts of the application, python is used due to the wide range of libraries and frameworks available for such tasks compared to other languages. The simplicity and consistency with the large community are also a benefit.

4.1.5 Data acquisition

The Kaggle dataset containing 4081 images split into 4 different classes and they are "Blast", "Browspot", "Tungro" and "Bacterialblight" [7] used for the detection of diseases and providing solutions.

The Kaggle dataset containing 1951 images split into 2 different classes and they are "Larva", and "little tiny red spiders" used for the detection of pests and providing solutions.

4.2 Commercialization aspects of the product

4.2.1 Target audience

The primary target audience for this application will be rice farmers with rice suppliers, researchers, buyers, sellers, and any person who is connected to the rice farming process being the secondary audience.

4.2.2 Design of the app

A comprehensive and easily understandable UI and UX are created so that even nontech-savvy users will not be confused while using the application. This will make sure that the application will reach a wide audience.

4.2.3 Gap in the market

Currently, in the play store, there is no other similar application to be found. This already makes the application unique and stand out.

4.2.4 Marketing Plan

The initial incentive will be to introduce this application to the farmers themselves. This will enable us to get feedback directly from the primary target audience which will then make it easier to enhance and optimize the application further, thus making a better product.

This application will be promoted by famous influencers and through social media platforms.

4.2.5 Pricing

Most of the functionality will be provided for free with them including,

- Pest identification
- Disease identification
- Weed identification
- Fertilizer type identification
- Weed mapping
- Growth deficiency identification

This will be with a free application.

In order for the solution providing functionality associated with the above-mentioned features to be enabled, a small price will have to be paid on a monthly basis.

4.2.6 Budget

A price will have to be allocated for the influencer and social media promotions. To publish the application another, sum also has been assigned. Finally, the database will also be required a monthly payment.

4.2.7 WBS

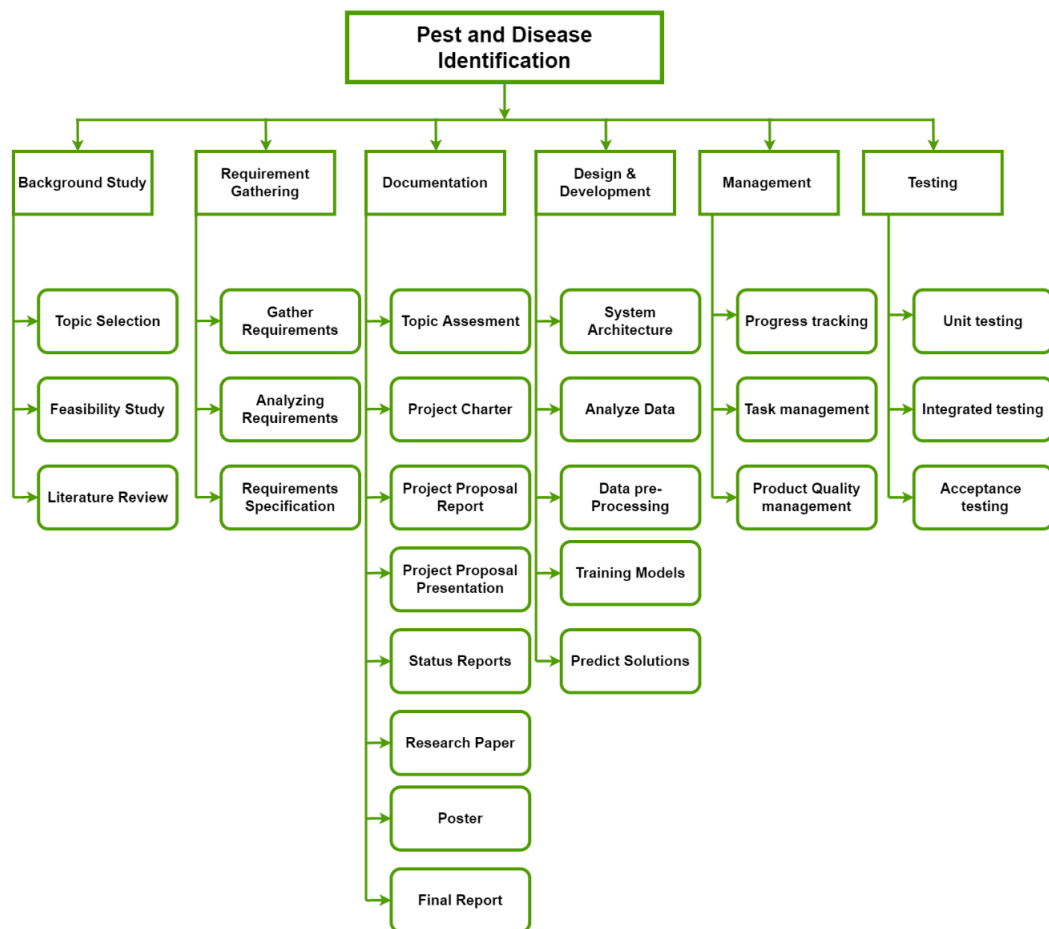


Figure 4:WBS chart

4.2.8 Gantt chart

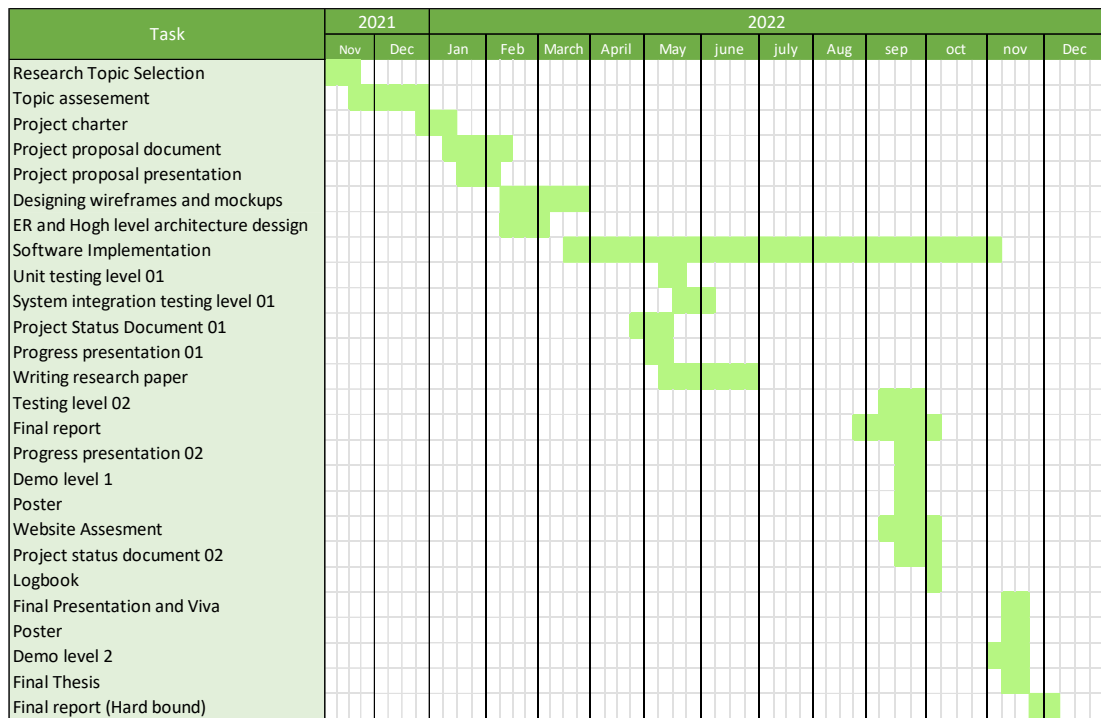


Figure 5: gantt chart

5. Testing & Implementation

5.1 Implementation

In this stage of the project, the implantation of the system will be started. This will be in accordance with the system architecture proposed in the previous design phase. The “Identification pests and diseases and proposing solutions” component will be further split into three subcomponents, with them being

- Identification of diseases using imagery
- Identification of pests using videos
- Proposing solutions.

5.1.1 Pre-processing

When it comes to pre-processing all the models that are described below went through the same process. Which includes shuffling, resizing, rescaling, and flipping horizontally and vertically. Finally, normalization was performed according to the mean and standard deviation calculated for the datasets.

5.1.2 Modal

5.1.2.1 DETECTION OF DISEASES AND PROVIDING SOLUTIONS:

Customized CNN was used as the main model for disease identification. CNN was chosen due to it being one of the most basic deep learning models which can take input images and have them differentiated. TensorFlow is primarily used here, this is due to the highly flexible array of tools, libraries, and community resources contained in it. Then Keras is used to provide abstractions and building blocks for the development of machine learning code. For plotting the outputs matplotlib was used due to its simple but detailed GUI. 80% and 20% split was made for the training and testing set. The data model starts with layers, they are 3 input layers with maxPooling2d, 3 output layers, 1 flattened layer, and 2 Dense layers. Different optimizers were attempted with the model to get the maximum results.

Modal summary – diseases

```
model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
sequential (Sequential)	(32, 250, 250, 3)	0
conv2d (Conv2D)	(32, 248, 248, 32)	896
max_pooling2d (MaxPooling2D)	(32, 124, 124, 32)	0

5.1.2.2 DETECTION OF PESTS AND PROVIDING SOLUTIONS:

For pest identification, AlexNet was used as the model with modifications being done to best fit the dataset. The reason why AlexNet was chosen is due to its relatively short training time compared to other deep learning models. This is because it allows multi-GPU usage thus making use of multiple GPUs if they are present to detect pests and provide solutions. Hyperparameter tuning [6] was performed for the parameters of batch size, learning rate, and epochs. Due to there being research showing that higher values for learning rate and batch size do not always provide Higher results, a lower number was chosen initially with it gradually going higher. As for the epochs, a brute force method was used to see which would be best.

Modal summary – pests

```
Alexnet_model.summary()

Model: "sequential"

Layer (type)                Output Shape                Param #
=====
conv2d (Conv2D)              (None, 55, 55, 96)         34944
max_pooling2d (MaxPooling2D) (None, 27, 27, 96)         0
conv2d_1 (Conv2D)             (None, 27, 27, 256)        614656
max_pooling2d_1 (MaxPooling2D) (None, 13, 13, 256)        0
conv2d_2 (Conv2D)             (None, 13, 13, 384)        885120
conv2d_3 (Conv2D)             (None, 13, 13, 384)        1327488
conv2d_4 (Conv2D)             (None, 13, 13, 256)        884992
max_pooling2d_2 (MaxPooling2D) (None, 6, 6, 256)         0
flatten (Flatten)            (None, 9216)               0
dense (Dense)                 (None, 4096)               37752832
dropout (Dropout)             (None, 4096)               0
dense_1 (Dense)                (None, 4096)               16781312
dropout_1 (Dropout)            (None, 4096)               0
dense_2 (Dense)                (None, 1000)               4097000
=====
Total params: 62,378,344
Trainable params: 62,378,344
Non-trainable params: 0
```

5.2 Testing and Maintenance

As the final phase of the Software development life cycle(SDLC) is the testing and maintenance phase which will be done under the discipline of functional and non-functional testing. The functional testing will mainly consider the functional requirements of the system and unit testing will be taken as the basis. Then in order to check the nonfunctional requirements such as performance and availability various

nonfunctional testing will be conducted. As for the maintenance of the application after the publication, various support features will be added.

6. RESULTS AND DISCUSSION

6.1 Results

6.1.1 Detection of pests and providing solutions

Adam is used as the optimizer since it gave the best outputs and it was fit for datasets such as this.

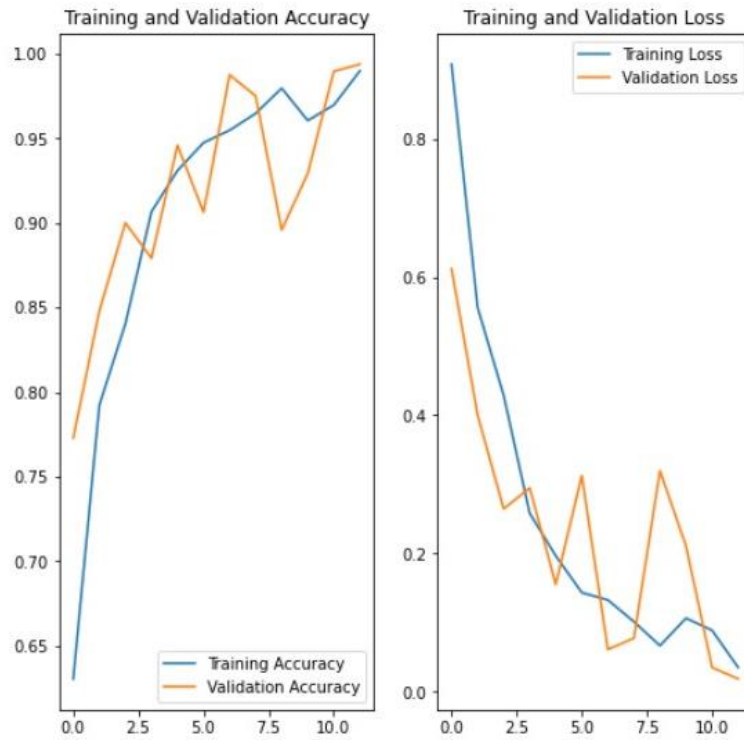


Figure 6:Test accuracy

The above figure represents training and validation accuracy with the loss. The y-axis depicts the accuracy for the first graph and the loss for the second graph. The x-axis depicts the number of epochs. As it is clear from the graphs with a large number of epochs, loss and accuracy fluctuation get higher. It is apparent that a large number of epochs does not negatively affect accuracy or loss.

Table 2: Model accuracy

Epoch	loss	accuracy	Val_loss	Val_accuracy
08/12	0.1013	0.9647	0.0775	0.9750
09/12	0.0666	0.9766	0.3196	0.8958
10/12	0.1064	0.9605	0.2108	0.9292
11/12	0.0888	0.9696	0.0351	0.9896
12/12	0.0351	0.9898	0.0186	0.9937

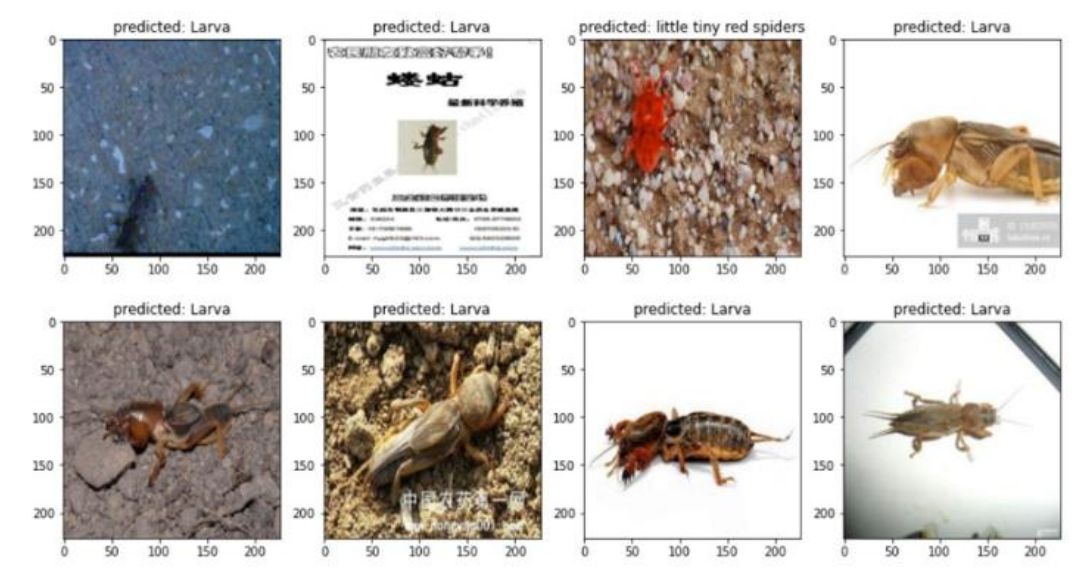


Figure 7: Pests prediction output

6.1.2 Detection of Diseases and providing solutions:

Table 3: Model accuracy

Epoch	Loss	accuracy	Val_loss	Val_accuracy
11/15	0.3942	0.8649	0.3157	0.8678
12/15	0.3645	0.8610	0.2362	0.9040
13/15	0.4016	0.8421	0.2662	0.8963
14/15	0.3234	0.8775	0.2476	0.9071
15/15	0.2875	0.8837	0.1775	0.9288

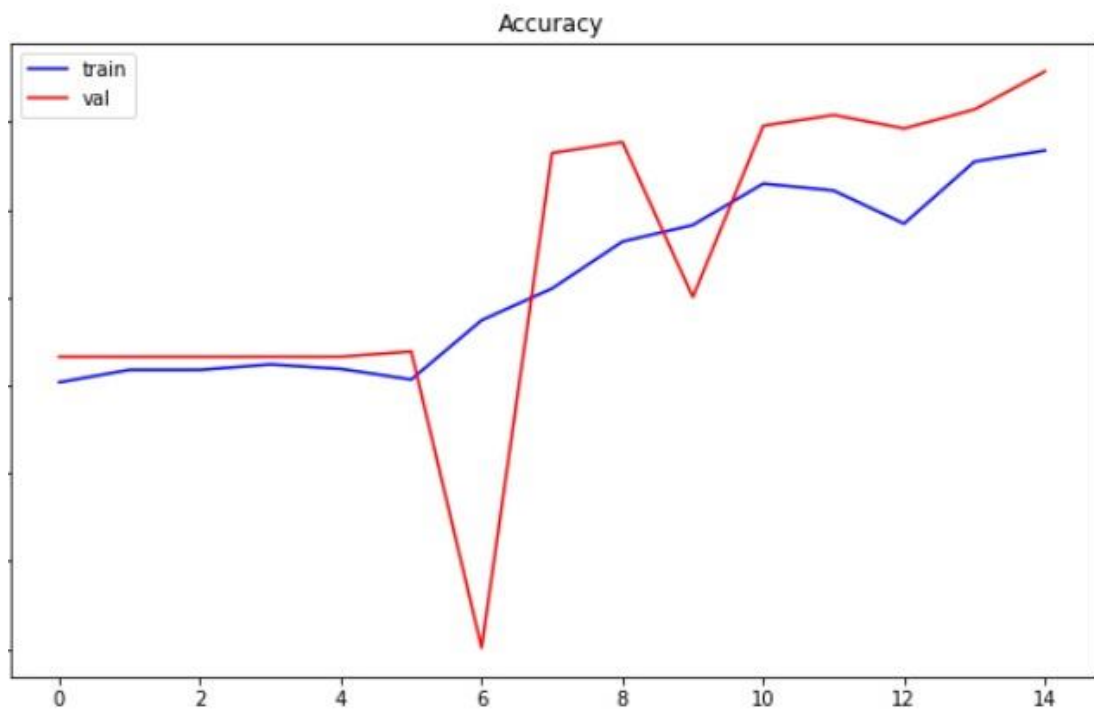


Figure 8: Training accuracy chart

The above figure represents training and validation accuracy. The y-axis depicts the accuracy and the x-axis depicts the number of epochs. Even though validation accuracy drops near the middle it quickly recovers and runs parallelly to the training accuracy. It is evident that the most favorable number of epochs is close to 15.

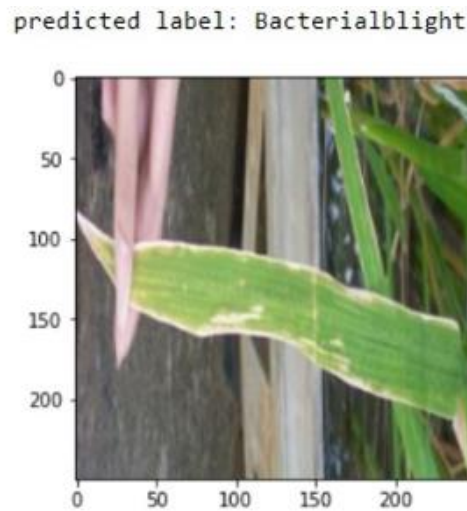


Figure 9:Diseases prediction output

6.2 Research Findings

6.2.1 Detection of pests and providing solutions

98.98% training accuracy and 97.71% test accuracy were able to be reached using this model as shown in the examples and predictions. 20 epochs and 32 batch sizes are the hyperparameters used in this model.

6.2.2 Detection of Diseases and providing solutions:

88.37% training accuracy and 92.88% test accuracy were able to be reached using this model as shown in the examples above with the predictions for the test data shown below. 15 epochs, 0.1 learning rate, and 32 batch size are the hyperparameters used in this model.

6.3 Discussion

After comparing with the other models such as customized AlexNet and custom CNN models, this ResNet model with the above-mentioned configurations was found to give the best accuracies for the weed identification task.

6.4 Summary of Each Student's contribution

- Discovering the best model for diseases identification from a pool of different CNN models
- Discovering the best configurations for that said model to acquire the best results
- Discovering the best model for pests identification from a pool of different CNN models
- Creating a mobile application for the created components
- Making the application as user-friendly as possible
- Find an appropriate dataset to train the models

Conclusions

This research paper was performed in order to provide rice farmers with solutions to the four major issues that they are currently facing which include pests, disease, weeds, fertilizers, and growth defects. In this research four CNN models are compared and contrasted in order to identify which one of them is best suited when it comes to rice and paddy farm datasets. Considering the outputs provided by four models which are used for image classification, the resnet50(modal 02) model performed best with it providing 99.43% for training accuracy and 97.04% for validation accuracy.

REFERENCE LIST

- [1] A. H. Sparks and A. Nelson, "Where rice pests and diseases do the most damage," Researchgate, 12 2012. [Online]. Available: https://www.researchgate.net/publication/270584803_Where_rice_pests_and_diseases_do_the_most_damage.
- [2] Google Books, "Crop Loss Assessment in Rice: Papers given at the International Workshop on Crop Loss Assessment to Improve Pest Management in Rice and Rice-Based Cropping Systems in South and Southeast Asia, 11-17 October 1987," *Google Books*, 1987.
- [3] Saillog Ltd, "Saillog Ltd," [Online]. Available: https://play.google.com/store/apps/details?id=com.agrio&hl=en_US. [Accessed 20 02 2020].
- [4] Private Pvt, "Crop Farmers App - Apps on Google Play," Bivatec Pvt, [Online]. Available: https://play.google.com/store/apps/details?id=com.bivatec.cropfarmersguide&hl=en_US. [Accessed 20 02 2020].
- [5] R. Raichaudhuri and R. Sharma, "Imaging-Based-Method-for-Precursors-of-Impending-Disease-from-Blood-Traces," *researchgate*.
- [6] J. L. M. and B. Gerardo, "Pest Identification using Image Processing Techniques in Detecting Image Pattern through Neural Network," *Semantic Scholar*.
- [7] UofM Mobile, "Pest & Crop Management," UofM Mobile, [Online]. Available: <https://play.google.com/store/apps/details?id=ca.umanitoba.ipm&hl=en&gl=US>.
- [8] R. A. and A., "Identification and recognition of rice diseases and pests using convolutional neural networks," ScienceDirect, [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/S1537511020300830>.

- [9] M. Eliz and M. L. Pai, "Detection of Rice Leaf Diseases Using Image Processing," IEEE Xplore, 2020. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9076527>. [Accessed 02 12 2021].
- [10] S. B. and R., "Image Processing Techniques for Diagnosing Rice Plant Disease: A Survey," IEEE Xplore, 03 2020. [Online]. Available: <https://ieeexplore.ieee.org/abstract/document/9076527>. [Accessed 2 12 2021].

Glossary

CNN – Convolutional Neural Network

FCN — Fully Convolutional Network

SDLC - Software development life cycle

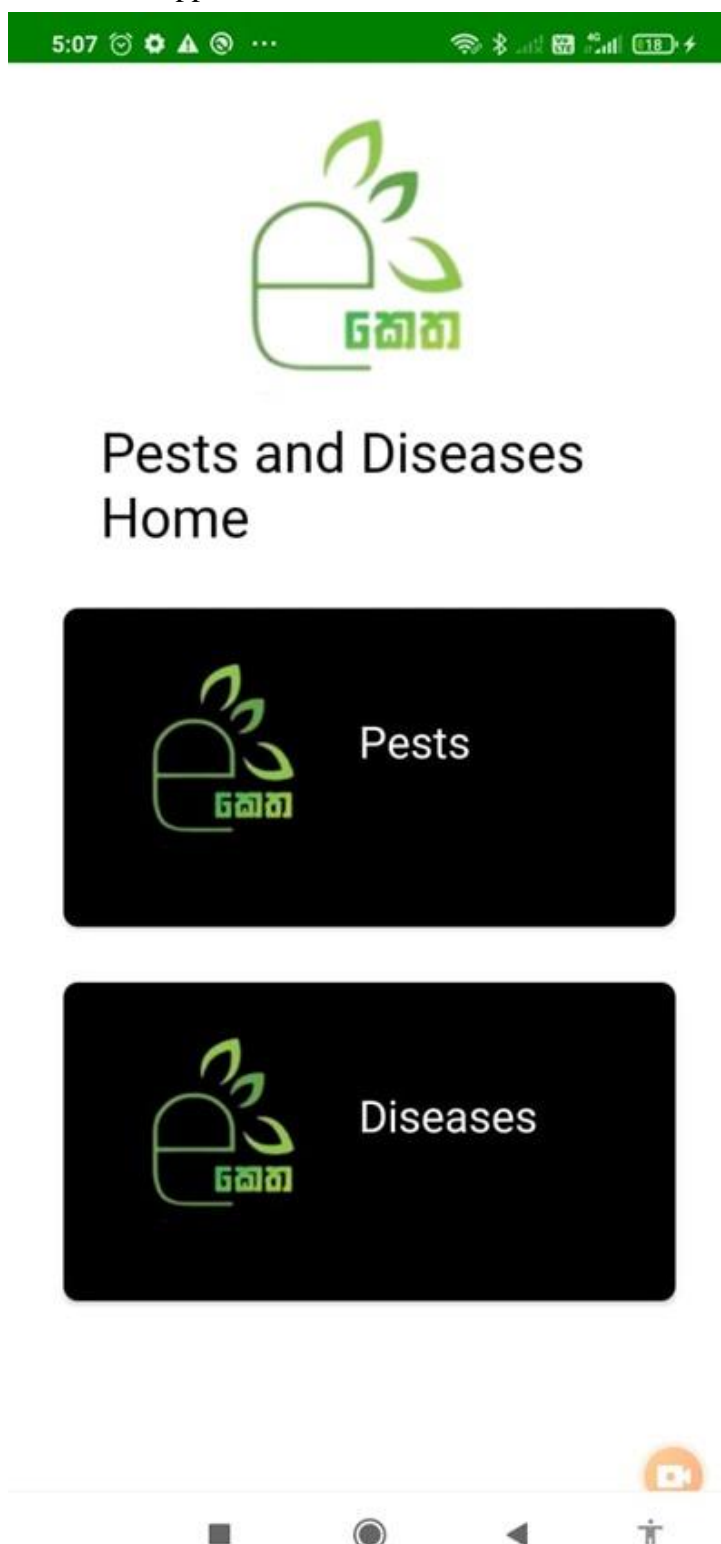
GPS - Global Positioning System

UI – User interface

UX – User experience

Appendices

Screenshots of the Application





Classified as pests:

white grubs

white grubs is caused by the bacterium *Pseudomonas syringae* pv. *syringae* (Pss), which survives in diseased stem tissue (cankers), plant debris, and soil. Pss can be spread by insects and on pruning tools, but is more commonly spread by wind and rain

Take Picture

Launch Gallery





Classified Diseases :

Tungro

Tungro is caused by the bacterium *Pseudomonas syringae* pv. *syringae* (Pss), which survives in diseased stem tissue (cankers), plant debris, and soil. Pss can be spread by insects and on pruning tools, but is more commonly spread by wind and rain

Take Picture

Launch Gallery



Training diseases classification model

```
Epoch 4/12
120/120 [=====] - 211s 2s/step - loss: 0.2581 - accuracy: 0.9066 - val_loss: 0.2948 - val_accuracy: 0.8792
Epoch 5/12
120/120 [=====] - 210s 2s/step - loss: 0.1966 - accuracy: 0.9309 - val_loss: 0.1552 - val_accuracy: 0.9458
Epoch 6/12
120/120 [=====] - 208s 2s/step - loss: 0.1434 - accuracy: 0.9474 - val_loss: 0.3126 - val_accuracy: 0.9062
Epoch 7/12
120/120 [=====] - 209s 2s/step - loss: 0.1327 - accuracy: 0.9547 - val_loss: 0.0611 - val_accuracy: 0.9875
Epoch 8/12
120/120 [=====] - 217s 2s/step - loss: 0.1013 - accuracy: 0.9647 - val_loss: 0.0775 - val_accuracy: 0.9750
Epoch 9/12
120/120 [=====] - 208s 2s/step - loss: 0.0666 - accuracy: 0.9796 - val_loss: 0.3196 - val_accuracy: 0.8958
Epoch 10/12
120/120 [=====] - 210s 2s/step - loss: 0.1064 - accuracy: 0.9605 - val_loss: 0.2108 - val_accuracy: 0.9292
Epoch 11/12
120/120 [=====] - 208s 2s/step - loss: 0.0888 - accuracy: 0.9696 - val_loss: 0.0351 - val_accuracy: 0.9896
Epoch 12/12
120/120 [=====] - 208s 2s/step - loss: 0.0351 - accuracy: 0.9898 - val_loss: 0.0186 - val_accuracy: 0.9937
```

```
In [29]: scores = model.evaluate(test_ds)
15/15 [=====] - 17s 477ms/step - loss: 0.0461 - accuracy: 0.9771
```

```
In [ ]:
```

Training pests classification model

```
Epoch 5/15
40/40 [=====] - 59s 1s/step - loss: 0.5266 - accuracy: 0.7594 - val_loss: 0.4901 - val_accuracy: 0.7663
Epoch 6/15
40/40 [=====] - 53s 1s/step - loss: 0.5408 - accuracy: 0.7533 - val_loss: 0.5396 - val_accuracy: 0.7693
Epoch 7/15
40/40 [=====] - 57s 1s/step - loss: 0.4958 - accuracy: 0.7871 - val_loss: 0.7090 - val_accuracy: 0.6006
Epoch 8/15
40/40 [=====] - 56s 1s/step - loss: 0.4725 - accuracy: 0.8052 - val_loss: 0.3608 - val_accuracy: 0.8824
Epoch 9/15
40/40 [=====] - 56s 1s/step - loss: 0.4419 - accuracy: 0.8319 - val_loss: 0.2999 - val_accuracy: 0.8885
Epoch 10/15
40/40 [=====] - 56s 1s/step - loss: 0.4280 - accuracy: 0.8413 - val_loss: 0.4405 - val_accuracy: 0.8003
Epoch 11/15
40/40 [=====] - 56s 1s/step - loss: 0.3942 - accuracy: 0.8649 - val_loss: 0.3157 - val_accuracy: 0.8978
Epoch 12/15
40/40 [=====] - 56s 1s/step - loss: 0.3645 - accuracy: 0.8610 - val_loss: 0.2362 - val_accuracy: 0.9040
Epoch 13/15
40/40 [=====] - 52s 1s/step - loss: 0.4016 - accuracy: 0.8421 - val_loss: 0.2662 - val_accuracy: 0.8963
Epoch 14/15
40/40 [=====] - 55s 1s/step - loss: 0.3234 - accuracy: 0.8775 - val_loss: 0.2476 - val_accuracy: 0.9071
Epoch 15/15
40/40 [=====] - 52s 1s/step - loss: 0.2875 - accuracy: 0.8837 - val_loss: 0.1775 - val_accuracy: 0.9288
21/21 [=====] - 5s 221ms/step - loss: 0.1775 - accuracy: 0.9288
92.879
```

Pest TensorFlow lite model

The screenshot shows the TensorFlow Lite Model Viewer interface. At the top, a tab is labeled 'pestmodel.tflite'. Below the tab, the 'Model' section displays the message 'No metadata found in this model' with a link 'Add metadata to your model' and a help icon. The 'Sample Code' section has tabs for 'Kotlin' and 'Java', with 'Java' selected. It contains a code block with Java code for initializing and running the model.

Model

No metadata found in this model
[Add metadata to your model](#) ⓘ

Sample Code

Kotlin Java

```
try {
    Pestmodel model = Pestmodel.newInstance(context);

    // Creates inputs for reference.
    TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 250, 250, 3}, DataType.FLOAT32);
    inputFeature0.loadBuffer(byteBuffer);

    // Runs model inference and gets result.
    Pestmodel.Outputs outputs = model.process(inputFeature0);
    TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

    // Releases model resources if no longer used.
    model.close();
} catch (IOException e) {
    // TODO Handle the exception
}
```

```

public void classifyImage2(Bitmap image){
    try {
        Pestmodel pModel = Pestmodel.newInstance(getApplicationContext());

        // Creates inputs for reference.
        TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 250, 250, 3}, DataType.FLOAT32);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * imageSize * imageSize * 3);
        byteBuffer.order(ByteOrder.nativeOrder());

        int[] intValues = new int[imageSize * imageSize];
        image.getPixels(intValues, 0, image.getWidth(), 0, 0, image.getWidth(), image.getHeight());
        int pixel = 0;
        //iterate over each pixel and extract R, G, and B values. Add those values individually to the byte buffer.
        for(int i = 0; i < imageSize; i++){
            for(int j = 0; j < imageSize; j++){
                int val = intValues[pixel++]; // RGB
                byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f / 1));
                byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f / 1));
                byteBuffer.putFloat((val & 0xFF) * (1.f / 1));
            }
        }
    }
}

```

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if(resultCode == RESULT_OK){
        if(requestCode == 3){
            Bitmap image = (Bitmap) data.getExtras().get("data");
            int dimension = Math.min(image.getWidth(), image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, filter: false);
            classifyImage2(image);
        }else{
            Uri dat = data.getData();
            Bitmap image = null;
            try {
                image = MediaStore.Images.Media.getBitmap(this.getContentResolver(), dat);
            } catch (IOException e) {
                e.printStackTrace();
            }
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, filter: false);
            classifyImage2(image);
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}

```



```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    if(resultCode == RESULT_OK){
        if(requestCode == 3){
            Bitmap image = (Bitmap) data.getExtras().get("data");
            int dimension = Math.min(image.getWidth(), image.getHeight());
            image = ThumbnailUtils.extractThumbnail(image, dimension, dimension);
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage1(image);
        }else{
            Uri dat = data.getData();
            Bitmap image = null;
            try {
                image = MediaStore.Images.Media.getBitmap(this.getContentResolver(), dat);
            } catch (IOException e) {
                e.printStackTrace();
            }
            imageView.setImageBitmap(image);

            image = Bitmap.createScaledBitmap(image, imageSize, imageSize, false);
            classifyImage1(image);
        }
    }
    super.onActivityResult(requestCode, resultCode, data);
}

```

Diseases TensorFlow lite model

model.tflite

Model

No metadata found in this model
[Add metadata to your model](#)

Sample Code

KotlinJava

```
try {
    Model model = Model.newInstance(context);

    // Creates inputs for reference.
    TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 250, 250, 3}, DataType.FLOAT32);
    inputFeature0.loadBuffer(byteBuffer);

    // Runs model inference and gets result.
    Model.Outputs outputs = model.process(inputFeature0);
    TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

    // Releases model resources if no longer used.
    model.close();
} catch (IOException e) {
    // TODO Handle the exception
}
```

```
public void classifyImage(Bitmap image){
    try {
        Model model = Model.newInstance(getApplicationContext());

        // Creates inputs for reference.
        TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 250, 250, 3}, DataType.FLOAT32);
        ByteBuffer byteBuffer = ByteBuffer.allocateDirect(4 * imageSize * imageSize * 3);
        byteBuffer.order(ByteOrder.nativeOrder());

        int[] intValues = new int[imageSize * imageSize];
        image.getPixels(intValues, 0, image.getWidth(), 0, 0, image.getWidth(), image.getHeight());
        int pixel = 0;
        //iterate over each pixel and extract R, G, and B values. Add those values individually to the byte buffer.
        for(int i = 0; i < imageSize; i++){
            for(int j = 0; j < imageSize; j++){
                int val = intValues[pixel++]; // RGB
                byteBuffer.putFloat(((val >> 16) & 0xFF) * (1.f / 255));
                byteBuffer.putFloat(((val >> 8) & 0xFF) * (1.f / 255));
                byteBuffer.putFloat((val & 0xFF) * (1.f / 255));
            }
        }

        inputFeature0.loadBuffer(byteBuffer);

        // Runs model inference and gets result.
        Model.Outputs outputs = model.process(inputFeature0);
        TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();
    }
}
```