International Research Journal of Advanced Engineering and Science
ISSN (Online): 2455-9024

Leaf Disease Detection Web Design using Django Framework for CNN Models

Raihan Haidar Arkan[1], Dr. Nur Sultan Salahuddin, SKom, MT[2]
[1]Master of Electrical Engineering, Gunadarma University, Depok, West Java, Indonesia-16424
[2]Faculty of Computer Science and Information Technology, Gunadarma University, Depok, West Java, Indonesia-16424

Abstract— *Agriculture is one of the most important sectors of the economy as it provides food for the world's growing population. Plant diseases are a serious problem in agriculture and can lead to reduced harvests and financial losses. The development of artificial intelligence technology, especially Convolutional Neural Network (CNN) has opened up opportunities to effectively detect plant diseases through image analysis. This research aims to build a system that can classify diseases and treatments on web-based plants with the Django framework for CNN model. Based on data from 30 respondents taken using a Likert scale, in this research the model that has been created and integrated on the website with the Django framework can classify well and this web can provide recommendations for handling diseases on plants based on the classification results obtained. This system is expected to help identify diseases in plants so that they can take preventive measures and can be a tool of education for the community.*

I. Introduction

Indonesia is an agricultural country in Southeast Asia with most of its population working in the agricultural sector. One of the plant products grown in Indonesia is potatoes, rice and grapes. Agriculture is one of the economic fields that are very important for the survival of the Indonesian people. Agriculture serves as a provider of food needs for Indonesia's growing population. Even so, the agricultural sector faces many problems, one of which is diseases experienced by plants such as vines, potatoes and rice. In biology, leaves are used as indicators to measure plant health by observing the color of the leaves, most of the symptoms of disease in plants can be known on the leaves [1]. Because such diseases can be identified by leaf color, it allows computers to be able to detect such diseases to help humans. Diseases of these crops can reduce crop quality, cause financial losses, and even threaten the country's food security [2]. Therefore, the detection of disease symptoms on plant leaves is expected to help the process of handling and controlling diseases in plants.

Grapes with the scientific name Vitis Vinifera, is one of the plants consumed by the people of Indonesia. Based on data from the Central Statistics Agency (BPS), Bali dominates wine production in Indonesia with a volume of 11,938 tons in 2022. This amount is equivalent to 88.32% of the total wine production of 13,516 tons. Meanwhile, West Nusa Tenggara occupies the second position with wine production of 384 tons in 2022[3]. One of the factors affecting wine production is diseases of the plant. Common diseases of grapes include black measles, black rot and leaf blight. The disease in grapes is difficult to distinguish with the naked eye, leading to inaccurate results [4].

Potatoes (Solanum tuberosum) are the most recognized staple food throughout the world and are foods that can be processed into various processed forms. Based on data from the Central Statistics Agency (BPS), East Java is the region with the most potato production in Indonesia, which is 324,338 tons. This figure is equivalent to 23.83% of the national total[5]. Diseases of potato plants, especially potato leaves, are generally caused by fungi, for example, late blight and early blight. The disease can spread throughout the potato plant parts.

Governments and experts have recognized how important it is to detect diseases in plants early as a prelude to preventive measures. Plant diseases often go undiagnosed until they reach a severe stage, causing crop failure[6]. About 40% of the world's crop is lost to disease and pest infestation[7]. In the past, farmers or agricultural experts used visual methods to detect diseases in plants. However, this method has limitations due to inaccuracies when identifying diseases in plants, especially in cases where plant growing conditions are affected by other variables. In addition, the manual identification process requires a lot of time and effort, making it inefficient if used on a larger scale. The introduction of diseases in early-stage crops can enable farmers to take effective preventive measures, such as the use of appropriate pesticides and more efficient disease management, These plant disease detection technologies can also help reduce excessive pesticide use, which can have a negative impact on the environment and human health.

Technological developments in the field of artificial intelligence, especially in the field of machine learning and deep learning, have opened up new ways to be able to detect diseases in plants. In machine learning, the Convolutional Nerual Network (CNN) technique has proven to be very effective in pattern recognition on image data. Using image analysis of disease-infected plants, CNNs can recognize relevant characteristics in images and learn complex patterns that allow CNNs to accurately identify different types of diseases in plants [8].

Although a lot of research has been done on the use of machine learning to detect plant diseases, there are still some issues that need to be addressed such as lack of verified datasets, machine learning model development requires large and varied datasets to train. However, the lack of a representative dataset on a wide range of diseases in plants is an obstacle to the development of accurate models. The next problem faced is the level of accuracy, accuracy in the disease detection system in plants is very important. If the disease is misdiagnosed, farmers may suffer greater losses and improper disease control due to inaccurate recognition.

Based on the issue described earlier, to overcome these various obstacles, this research aims to create a classification system for diseases in plant leaves and how to treat plants based on the web by utilizing the Django framework that can help build websites that can identify diseases in plant leaves and provide treatment recommendation. The system uses the Django framework that allows a web to perform classification using CNN technology which can train disease detection models on plant leaves with diverse datasets. This web-based system aims to quickly and accurately identify plant leaf diseases, while also serving as an educational tool for controlling and preventing them.

II. Research Method

2.1. *Design Stage*

At this stage, the entire structure of the website that will be built is designed. In the program design, this research uses Google Colab to conduct classification training on the model to be used. At the website development stage, this research is assisted by the Django framework which can facilitate website development because the Django framework allows users to create back-end and front-end parts in one environment.

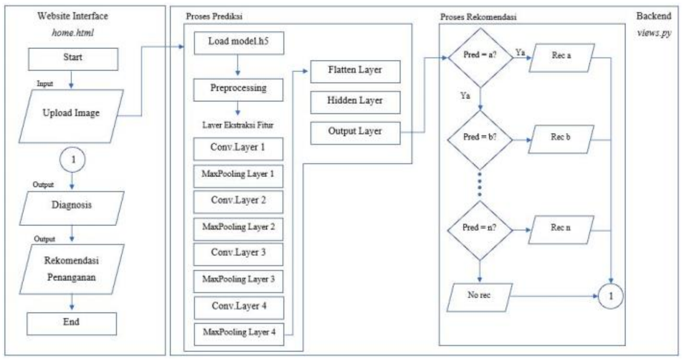


Figure 1. Flowchart

Flowchart display in figure 1. Shows how a web-based disease classification system works. The way this system works starts when the user uploaded or inputs image on the main display of the website, after the upload process, the Django framework precisely the *views.py* file will preprocess the image according to the image settings when the model is created, after the preprocessing process, the image features that have been inputted will be extracted so that the model can make predictions based on the features possessed by the image. After the prediction process is complete, the model will provide the prediction results with the largest percentage and will be displayed on the main display of the web in the classification results section.

When the prediction results are generated, the Django framework will also send treatment recommendations according to the program that has been created. The program to give these treatment recommendations uses a condition system, where the results of recommendations will be displayed according to the results of the prediction.

In the design process there are several points that need to be designed to describe the system, the following are the points to build this research model:

1. The purpose of the system, the system built on this research is a web that can identify a disease in plants and provide a treatment recommendation to deal with or prevent the disease from being too severe based on images uploaded to the web.
2. The way the system works, the system in this research uses a web base with the Django framework. This system will provide input identification results in the form of images of plant's leaf that have a disease, after identifying the disease from the input in the form of images, the system will provide treatment recommendations according to the identification results.
3. Framework, a web that can identify diseases, uses Django as a framework that allows the web to use CNN model to identify diseases and provide treatment recommendations for handling identified plant diseases.
4. *Datasets*, datasets used each amount to approximately 1000 to 2000 images for training all types of plants and their diseases. In this *dataset*, image data has been separated into 3 (three parts), namely, *data*, *val*, and *train*.
5. Model, this system uses models that are trained using the CNN or *Convolutional Neural Network* method which is usually widely used to recognize and classify images.
6. Type of disease, in this system can detect types of plants namely grapes, potatoes and rice. Diseases that can be detected by this system are:
   a. Grape
      i. *Black Rot*
      ii. *Esca/Black Measley*
      iii. *Leaf Blight*
   b. Potato
      i. *Early Blight*
      ii. *Late Blight*
   c. Rice
      i. *Brown Spot Disease*
      ii. *Hispa*
      iii. *Leaf Blight*

2.2. *Block Diagram*

The block diagram system is important in the development of a research that produces applications or tools, because block diagrams provide how an application works as a whole.

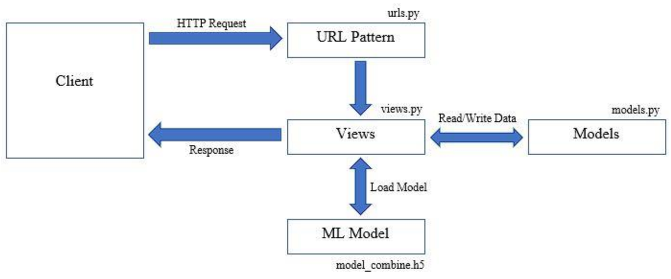


Figure 2. Block Diagram

2.3. *Image Upload Process*

This process occurs on the *client side* or the main display of the classification website. In this process, the user will choose the type of plant that he wants to identify the disease.

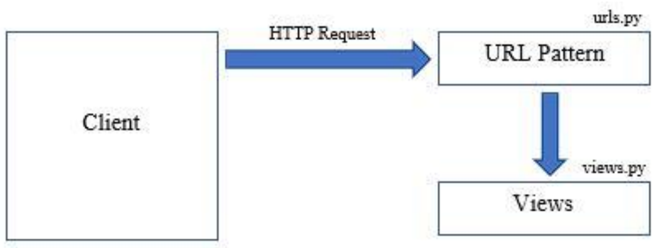


Figure 3. Upload Process

This process is commonly referred to as the input block in the diagram block. The following is an explanation during the image upload process:

1. This process involves 3 files namely, home.html, urls, py, and views.py. On the website display (*home.html*) the image upload process will be triggered when the user presses the "*classify*" button.
2. When the classify button is triggered, the browser sends an HTTP POST request to the URL specified in Django's form attribute. Then *urls.py* is tasked with matching the requested URL to the *view* function.

2.4. Home.html Program

In this section will explain about programs related to the process of uploading images on the website display precisely in *home.html*.

```
    <Form method="Post"  action="{% URL 'classify' %}"enctype="multipart/form-data">
    {% csrf_token %}
    <label for="model">Select Plant Type</label>
    <br>
    <select name="model"  id="model">
        <option value="model_potato">Potato</option>
        <option value="model_rice">Rice</option>
        <option value="model_grape">Grape</option>
    </select>
    <br>
    <br>
    <input type="file"  name="image"  accept="image/*">
    <br>
    <br>
    <input type="submit" name="upload_button" value="Classify">

    </form>
```

This program functions as a <form></form> used to access Django's backend. In this form there is a function to select a model with the model name and a function to upload an image with the image name and a button value = "Classify" to send a *request*.

2.5. urls.py Program

This file serves to redirect each request from HTTP POST to the specified URL

```
urlpatterns = [
path('', views.index, name='index'),
]
```

The urlpatterns function is used to store any kind of URL pattern needed in a Django project. Each URL pattern is defined using the path function. Path ('', views.index, name='index') The program defines a URL pattern that connects URLs with the home display function in the *views* module so that the URL generated by default is *https://localhost:8000/*.

2.6. views.py Program

The program in *the views.py* file is used to receive requests from forms on the main display or *client side*. In this case, the request received is in the form of a request model used to select the type of plant and a request image used to receive and send the inputted image.

```
# Get the model and image
selected_model = request. POST['model']
uploaded_image = request. FILES['image']

# Get the uploaded image from the form
uploaded_image = form.instance.image
```

The function of the request program. POST['model'] is to receive requests from the main view in the form of plant types, in this case it will later be used to determine the cnn model to be used for training, requests for this plant type will be stored in the selected_model variable, while request. FILES['image'] is a program to receive digital images that have been uploaded by the user, these digital images will be stored in uploaded_image variables which will later be used as input for *Image Preprocessing*.

2.7. Prediction Process on the Web

This process involves *views.py* and a trained cnn model. In this process, when a request from the form on the main display of the website is received by the *views.py* in the previous program, then the image and plant type request that has been received will be processed. In this process *views.py* is tasked with loading cnn models that have been trained with the hdf5 extension (.h5).

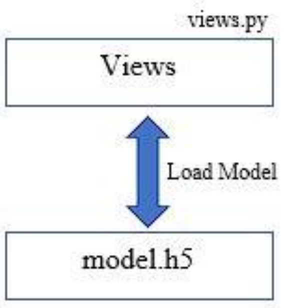


Figure 4. Prediction Process

2.7.1. Defining Disease Classes

Before carrying out the process involving prediction, a class name is defined to name the prediction results according to the index.

```
class_names_potato = {
0:  'Potato Early Blight',
1:  'Potato Healthy',
2:  'Potato Late Blight'
}
```

The above program is an example of defining a disease class name, because without a defined class name, the result of a prediction is only a number without any class name or disease name.

2.7.2. Preprocessing Image

In this program, *Preprocess Image* is carried out before making predictions using the model that has been trained. The purpose of the program is to prepare a digital image uploaded by the user to match the image format used by the model.

```
img_upload = image.load_img(uploaded_image.path, target_size=(256, 256))
img_array = image.img_to_array(img_upload)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.0
```

1. img_upload. In this program, the image uploaded by the user will be read through the path that has been written on the previous variable, namely *uploaded_image* and the image will be resized to 256x256 *pixels* to match the size used by the model during the training process.
2. image.img_to_array(img_upload). After the image is loaded, it will then be converted to an array or numpy array. Thus the image to be represented by an array of numbers can be understood by the model as input.
3. Np. expand_dims(img_array, axis=0). Deep *learning* models accept input in batches, therefore it is necessary to add batch dimensions to the image array in order for the model to process them. After this operation the images will have dimensions (1,256,256,3), where 1 indicates that there is 1 image in the batch.
4. img_array /= 255.0. This step is normalized in the image array to change the value from 0 to 255 to 0 to 1. This normalization is necessary because *deep learning* models tend to be better at coping with data that has a more uniform scale of values, while a range of 0 to 1 makes it easier for the model to process.

2.7.3. Determining Plant Type

After the digital image that has been uploaded passes through *Image Preprocessing*, the next step is to load the cnn model according to the type of plant chosen by the user. Because each type of plant selected by the user has a different cnn model.

```
if selected_model ==  'model_potato':
model = tf.keras.models.load_model('model_potato_v2.h5')
current_class_names = class_names_potato
Elif selected_model ==  'model_rice':
model = tf.keras.models.load_model('model_rice_v1.h5')
current_class_names = class_names_rice
Elif selected_model ==  'model_grape':
model = tf.keras.models.load_model('model_grape_v2.h5')
current_class_names = class_names_grape
```

The if selected_model program is a condition program that if the type of plant selected by the user will affect the model that will be used to make predictions. Program tf.keras. models. load_model used to load trained cnn models. And current_class_names is a program used to determine the class of diseases by index.

2.7.4. Making Prediction

The program in this section is made to make predictions on uploaded digital images, this prediction process will use a cnn model selected based on the choice of plant types that have been selected by the user. This prediction program will also convert the softmax output into percentage form to make it easier to see.

```
predictions = model.predict(img_array)
predicted_class_index = np.argmax(predictions[0])
predicted_class = current_class_names[predicted_class_index]
probability = "{:.2f}".format(Predictions[0][predicted_class_index] * 100)
```

The above program is an implementation of the prediction process using *a pre-trained deep learning* model. The goal of this program is to predict classes from input in the form of images (img_array) based on pretrained models.

1. model.predict(img_array). In this program model.predict is used to predict the image brought by the img_array. When the image is inputted, the model will generate probability predictions. The results of these predictions will be stored in the predictions variable.
2. np.argmax(predictions[0]). After getting the probability prediction results from the model, the next step is to determine the class that has the highest probability, therefore the np.argmax command is used.
3. class_names[predicted_class_index]. After obtaining the class index with the highest probability, the program searches for class names corresponding to that index based on class_names variables containing class names according to the number of classes in the dataset.
4. probability. The program contained in this variable serves to calculate the probability of prediction in percentage form.

2.7.5. Prediction Percentage Formula

The method used by *Conlutional Neural Networks* to generate probability values is from the value of each neuron which indicates the extent to which the digital image input matches the class concerned, then this score will be converted into probability through the softmax activation function.

The softmax activation function is responsible for converting *the output layer score* into probability with a maximum value of 1. The formula used by softmax activation to determine probability can be seen in the equation below:

$$P(y_i|x) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

Information:

$P(y_i|x)$ = The probability of the input image x belongs to class i.

z_i = Score generated by the I-th neuron

N = Total number of classes

To get a score value on *the Convolutional Neural Network* can be obtained by a combination of *weight* and input given to the neuron. The process involves the addition of *weighted sum* and the application of the activation function.

1. *Weighted Sum*. Each neuron input is multiplied by the corresponding *weight* after which the results of multiplication are summed. In equation, x is the input of the neuron and w is the weight.
$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n$$
2. Application of activation function. After *Weighted Sum* is performed, the z value is usually used as input for the activation function. This function will convert the z value into an output that has a non-linear nature. For example in ReLU activation, the α output of the neuron can be calculated as:
$$\alpha = \max(0, z)$$

2.8. Recommendation Program

The recommendation process also takes place on the *views.py* and uses a condition system to determine which recommendations to display. The Recommendation System will provide handling recommendations based on predictions issued by the model.

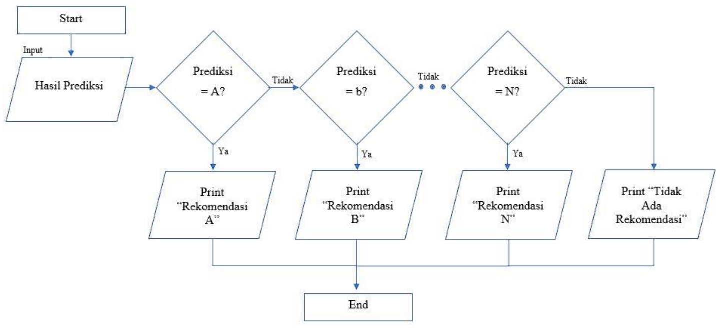


Figure 5. Flowchart Recommendation

```
if predicted_class == 'Disease A':
treatment = 'Recommendation A'
trsource = 'Source A'
elif predicted_class == 'Disease B':
treatment = 'Recommendation B'
trsource = 'Source B'
.
.
.
elif predicted_class == 'Disease N':
treatment = 'Recommendation N'
trsource = 'Source N'
```

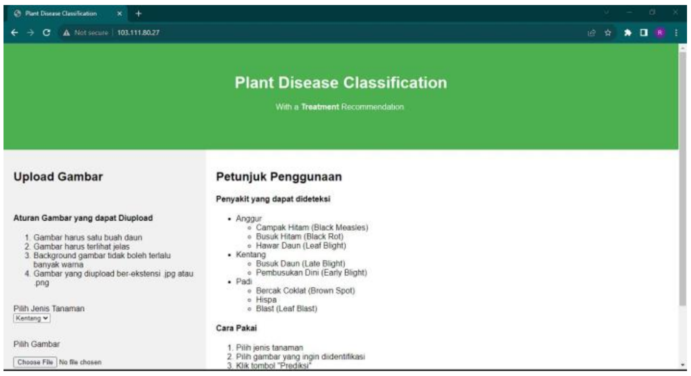
In the recommendation program, the results of the recommendations to be displayed will match the disease predicted by the model. This recommendation program utilizes the condition system to provide treatment recommendations based on identified diseases. The results displayed by this recommendation program also include the handling resources used to recommend the treatment.

III. Results And Discussion

3.1. Web Display

This stage is testing the website to see if the web that has been built can be accessed and test if the website in this research can classify properly and provide treatment recommendations on how to prevent diseases in plants.

The web is built using a mix of various languages such as HTML, Python, JavaScript, and CSS. The web built on this research uses a framework provided by Django, making it easier for researchers to create a web classification of diseases in plants.


Plant Disease Classification
Not secure | 103.111.80.27

Plant Disease Classification
With a **Treatment** Recommendation

Upload Gambar

Aturan Gambar yang dapat Diupload
1. Gambar harus satu buah daun
2. Gambar harus terlihat jelas
3. Background gambar tidak boleh terlalu banyak warna
4. Gambar yang diupload ber-ekstensi .jpg atau .png

Pilih Jenis Tanaman
Kentang

Pilih Gambar
Choose File No file chosen

Petunjuk Penggunaan

Penyakit yang dapat dideteksi
- Anggur
  - Campak Hitam (Black Measles)
  - Busuk Hitam (Black Rot)
  - Hawar Daun (Leaf Blight)
- Kentang
  - Busuk Daun (Late Blight)
  - Pembusukan Dini (Early Blight)
- Padi
  - Bercak Coklat (Brown Spot)
  - Hispa
  - Blast (Leaf Blast)

Cara Pakai
1. Pilih jenis tanaman
2. Pilih gambar yang ingin diidentifikasi
3. Klik tombol "Prediksi"

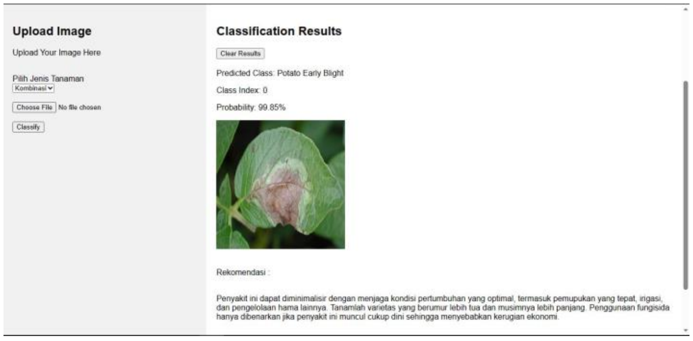
Figure 5. Website Display

Figure 5, is an image of the appearance of the website built in this research. This website has several features such as:

1. Choosing the type of plant. This feature allows users to select the desired type of plant, this feature will use a model that matches the type of plant selected.
2. Image upload feature. It is the main feature of this website as input to run the identification process by pressing the "*classify*" button after selecting the image you want to classify.

3.2. Display of Classification Result

After identifying the disease in the image uploaded by the user. The results will be displayed in the "Classification Results" section. The results displayed are images that have been uploaded or that become inputs, disease predictions, and probabilities. Then in the "Recommendations" section there are recommendations on how to treat or prevent diseases that have been identified by the model along with the source of recommendations for handling the disease used.



Classification Results

Predicted Class: Potato Early Blight

Class Index: 0

Probability: 99.85%

Rekomendasi :

Penyakit ini dapat diminimalisir dengan menjaga kondisi pertumbuhan yang optimal, termasuk pemupukan yang tepat, irigasi, dan pengelolaan hama lainnya. Tanamlah varietas yang berumur lebih tua dan musimnya lebih panjang. Penggunaan fungisida hanya dibenarkan jika penyakit ini muncul cukup dini sehingga menyebabkan kerugian ekonomi.

Figure 6. Display of Classification Result

3.3. Website Test Result according Respondent

This research uses the Likert Scale and the Technology Acceptance Model (TAM) as the basis for determining the questionnaire, where there are several criteria for determining the level of technology acceptance to users.

Table 1. Perceived Easy of Use

Perceived Easy of Use	
1	Django-based Leaf Disease Detection Website is easy to learn
2	Django-based Leaf Disease Detection Website is easy to use
3	Django-based Leaf Disease Detection Website is easy to access

Table 2. Perceived Usefulness

Perceived Usefulness	
1	The use of the Django-based Leaf Disease Detection Website makes it easy to determine the disease on the leaves.
2	The use of Django-based Leaf Disease Detection Website helps to speed up the determination of plant disease treatment.
3	Django-based Leaf Disease Detection Website can be an educational tool in recognizing and treating plant diseases.
4	I found it helpful to use the Django-based Leaf Disease Detection Website when identifying leaf diseases.

Table 3. Attitude Toward Using

Attitude Toward Using	
1	I use the Django-based Leaf Disease Detection Website because it suits my needs.
2	I like using the Django-based Leaf Disease Detection Website because there is a guide to using it

Table 4. Actual Use

Actual Use	
1	Overall, I am satisfied with the leaf disease classification website.
2	Django-based Leaf Disease Detection Website has good accuracy
3	I will use the Django-based Leaf Disease Detection Website to identify plant diseases.

The assessment carried out in this research using a Likert scale has provisions based on points from strongly agree to strongly disagree.

Table 5. Context Value

Context	Value
Strongly Agree (SA)	5
Agree (A)	4
Neutral (N)	3
Disagree (D)	2
Strongly Disagree (SD)	1

The questionnaire in this research utilizes the Google Form feature to collect respondent data and is distributed online on the internet. Respondents in this study had 30 respondents. Based on the answers and the number of respondents, it can be seen the percentage index of acceptance of the Django-based Plant Leaf Disease Classification Web to users.

Table 6. Frequensy of Perceived Easy of Use Data

S	SA		A		N		D		SD		A
	F	%	F	%	F	%	F	%	F	%	
1	9	30.0	11	36.6	8	26.6	2	6.6	0	0	3.9
2	9	30.0	13	43.3	7	23.3	1	3.3	0	0	4.0
3	9	30.0	11	36.6	8	26.6	2	6.6	0	0	3.9

Table 7. Frequensy of Perceived Usefulness Data

S	SA		A		N		D		SD		A
	F	%	F	%	F	%	F	%	F	%	
1	11	36.6	13	43.4	5	16.6	1	3.3	0	0	4.1
2	10	33.3	11	36.6	9	30.0	0	0	0	0	4.0
3	16	53.3	10	33.3	3	10.0	1	3.3	0	0	4.3
4	12	40.0	13	43.3	5	16.6	0	0	0	0	4.2

Table 8. Frequency of Attitude Toward Using Data

S	SA		A		N		D		SD		A
	F	%	F	%	F	%	F	%	F	%	
1	10	33.3	11	36.6	7	23.3	2	6.6	0	0	3.9
2	8	26.6	16	53.3	4	13.3	2	6.6	0	0	4.0

Table 9. Frequensy of Actual Use Data

S	SA		A		N		D		SD		A
	F	%	F	%	F	%	F	%	F	%	
1	11	36.6	13	43.3	4	13.3	2	6.6	0	0	4.1
2	5	16.6	16	53.3	7	23.3	2	6.6	0	0	3.8
3	10	33.3	16	53.3	3	10.0	1	3.3	0	0	4.1

Based on respondents regarding the Actual Usage criteria in table 6 to table 9, out of 30 respondents gave an average score of 4 (Agree) that the website built in this study can function properly in direct use by users

3.4. Grape Leaf Disease Treatment Recommendations

This stage displays the classification results of grape diseases with recommendations for handling them on the classification website that has been built in this research. There are 3 types of grape diseases that can be classified, namely Black Measles, Black Rot, and Leaf Blight. The following are the recommendations generated based on the classification results of the 3 grape diseases:

1. Black Measles : “This disease should be treated in the same way as you would Eutypa or Botryosphaeria dieback. Avoid making large cuts during pruning and do not prune during periods of high humidity. Seal wounds or spray fungicide after pruning.”
2. Black Rot : “Organic Control: As soon as it enters the blooming stage, you can spray Bacillus thuringiensis | Chemical Control: Chemical application is done in a preventive manner. Strat spraying is approximately two weeks before blooming with captan + mycobutanil or mancozeb + mycobutanil. Just before the flowers bloom, you can also use carbaryl or imidacloprid. Post-bloom spray mancozeb + mycobutanil, imidacioprid or azadirachtin. Ten days after blooming, you can also use a mixture of captan and sulfur on your vines. Since most grape varieties become immune to infection three to four

weeks after blooming, chemical sprays should be avoided at that time."

3. Leaf Blight : "The disease can be effectively tackled through the combined use of culture, sanitation, resistance and fungicide sprays. This integrated approach to disease control minimizes reliance on one type of control over another and usually results in a high percentage of quality fruit."

3.5. Potato Leaf Disease Treatment Recommendations

This stage displays the results of recommendations for handling potato diseases based on the classification results. This classification website can provide recommendations for handling 2 diseases that can be classified, namely Late Blight and Early Blight, the following are recommendations for handling each disease in potato plants:

1. Early Blight : "These diseases can be minimized by maintaining optimal growing conditions, including proper fertilization, irrigation, and other pest management. Plant older and longer-season varieties. The use of fungicides is only justified if the disease appears early enough to cause economic losses."
2. Late Blight : "Late blight is controlled by removing cull piles, using proper harvesting and storage practices, and using fungicides when necessary. Air drainage to facilitate daily drying of foliage is important."

3.6. Rice Leaf Disease Treatment Recommendations

This stage displays the results of recommendations for handling rice plant diseases based on the classification results, there are 3 diseases that can be detected on the classification website, namely Brown Spot, Hispa, and Blast, the following are the results of recommendations for handling rice diseases that can be classified:

1. Brown Spot: "Use fungicides (e.g., iprodione, propiconazole, azoxystrobin, trifloxystrobin, and carbendazim) as seed treatments. Warm water (53-54°C) for 10-12 minutes before planting to control primary infections at the seedling stage."
2. Hispa : "Organic Control: To this day, there are no effective biological controls for this disease that are commercially available. Experiments are ongoing to test the feasibility of products based on Streptomyces or Pseudomonas bacteria on fungus and disease incidence or spread. | Chemical Control: Seed treatment with thiram is effective against this disease. Fungicides containing azoxystrobin, or active ingredients from the triazole or strobilurin family can also be sprayed at the seedling, tillering, and panicle emergence stages to control blast disease. One or two fungicide applications at planting time can be effective in controlling this disease."

IV. Conclusion

Based on the data provided by 30 respondents in this research, the average value obtained on each statement submitted gets a value of 4 or tends to a value of 4 so that it falls into the Agree category. This proves that the classification website built in this study functions well, can be accepted by users, and can be a means of education for the community and help people who have a hobby of gardening or plant lovers to identify plant diseases and carry out prevention or treatment based on recommendations for handling the identified disease.

The Django framework can be an option to help build websites for classification and provide prevention recommendations based on prediction results. This is because the Django framework provides a variety of libraries, modules, and APIs that can be used freely so there is no need to do programming from scratch.

References

[1] M. M. Ali, N. A. Bachik, N. 'Atirah Muhadi, T. N. Tuan Yusof, and C. Gomes, "Non-destructive techniques of detecting plant diseases: A review," *Physiol. Mol. Plant Pathol.*, vol. 108, 2019, doi: 10.1016/j.pmpp.2019.101426.

[2] P. K. Mugithe, R. V. Mudunuri, B. Rajasekar, and S. Karthikeyan, "Image Processing Technique for Automatic Detection of Plant Diseases and Alerting System in Agricultural Farms," *Proc. 2020 IEEE Int. Conf. Commun. Signal Process. ICCSP 2020*, pp. 1603–1607, 2020, doi: 10.1109/ICCSP48568.2020.9182065.

[3] R. Mustajab, "Bali Miliki Produksi Anggur Terbesar di Indonesia pada 2022," *DataIndonesia.id*, 2023. https://dataindonesia.id/sektor-riil/detail/bali-miliki-produksi-anggur-terbesar-di-indonesia-pada-2022 (accessed Aug. 09, 2023).

[4] S. S. Simanjuntak, H. Sinaga, K. Telaumbanua, and A. Andri, "Klasifikasi Penyakit Daun Anggur Menggunakan Metode GLCM, Color Moment dan K*Tree," *J. SIFO Mikroskil*, vol. 21, no. 2, pp. 93–104, 2021, doi: 10.55601/jsm.v21i2.754.

[5] S. Widi, "Produksi Kentang Indonesia Terbanyak di Jawa Timur pada 2021," *DataIndonesia.id*, 2022. https://dataindonesia.id/agribisnis-kehutanan/detail/produksi-kentang-indonesia-terbanyak-di-jawa-timur-pada-2021 (accessed Aug. 09, 2023).

[6] E. Maria, F. Fadlin, and M. Taruk, "Diagnosis Penyakit Tanaman Padi Menggunakan Metode Promethee," *Inform. Mulawarman J. Ilm. Ilmu Komput.*, vol. 15, no. 1, pp. 27–31, 2020, [Online]. Available: https://e-journals.unmul.ac.id/index.php/JIM/article/view/2844.

[7] H. P. Angjaya, K. Gunadi, and R. Adipranata, "Pengenalan Penyakit pada Tanaman Pokok di Indonesia dengan Metode Convolutional Neural Network," *J. Infra*, 2021, [Online]. Available: http://publication.petra.ac.id/index.php/teknik-informatika/article/view/11426%0Ahttp://publication.petra.ac.id/index.php/teknik-informatika/article/viewFile/11426/10036.

[8] J. S. H. Al-Bayati and B. B. Ustundag, "Artificial Intelligence in Smart Agriculture: Modified Evolutionary Optimization Approach for Plant Disease Identification," *4th Int. Symp. Multidiscip. Stud. Innov. Technol. ISMSIT 2020 - Proc.*, 2020, doi: 10.1109/ISMSIT50672.2020.9255323.