POTATO DISEASE CLASSIFICATION USING DEEP LEARNING

TEAM ID: 592396

RAGHUL G 21BEC1662

NIRANJANA D 21BEC1655

RAGUL D 21BEC1625

OVISHREE S 21BEC1692

INTRODUCTION:

The objective is to provide a tool that can help farmers monitor and manage potato leaf disease in real-time, ultimately increasing crop yields and reducing economic losses. Creating a comprehensive deep learning system to categorize images of potato leaves into three groups—healthy, early blight, and late blight—is the aim of this research. Convolutional neural networks (CNNs) are used in the suggested approach to identify relevant characteristics from the input images and assign them to one of the three groups. Since potato leaf disease can have major effects on crop output and quality, it is important to predict it as soon as possible. Farmers can minimize crop loss and stop the disease's spread by reacting quickly upon early discovery.

Crop productivity and quality can be considerably decreased by potato leaf disease. Early disease detection allows farmers to minimize crop loss by controlling the disease's spread.

Farmers can lower the cost of disease management methods like pesticides and other treatments by detecting potato leaf disease early. Farmers may target the exact crop region affected by the disease by detecting it early, which lowers the overall cost of management methods.

The environment might be harmed from the overuse of pesticides and other management methods. By minimizing the use of pesticides and focusing primarily

on the damaged regions, farmers may lessen their impact on the environment by detecting potato leaf disease early on.

Potato leaf disease can degrade crop quality and reduce crop appeal to consumers. Early disease detection allows farmers to take the necessary precautions to stop the disease's progress, which raises the crop's overall quality.

LITERATURE REVIEW:

a. A SURVEY ON DISEASE DETECTION OF A POTATO LEAF USING CNN

Sindhuja Bangari; P Rachana; Nihit Gupta; Pappu Sah Sudi; Kamlesh Kumar Baniya

India is an agricultural country, and crop production rate is a major concern for the country. Less production means higher crop prices and more hunger for those who can't even afford potatoes, so in order to improve crop yield rates and minimize disease infection in plants, deep learning models have developed a technology that will make farmer work easier to some extent. They may rely on Deep Neural Networks, a subfield of AI technology, to detect diseased plants and avoid doing it manually, as well as provide effective treatment in the bud stage before it is too late. This research reviewed several publications and discovered that Convolution Neural Networks (CNN) performs better in detecting leaf illness. It was also shown that CNN contributes the greatest potential accuracy for disease identification.

b. INTELLIGENT PLANT DISEASE DIAGNOSIS USING

CONVOLUTIONAL NEURAL NETWORK: A REVIEW

Diana Susan Joseph, Pranav M Pawar & Rahul Pramanik

In recent times use of different technologies for intelligent crop production is growing. To increase the production of crops, diagnosing a plant disease is very important. Plant diseases can be identified using various techniques like image processing, machine learning, deep learning, etc. Among these techniques deep learning, especially deep learning using convolutional neural networks (CNN) has proved to be more efficient in recent years compared to other methods. This manuscript focuses mainly on the diseases affecting on eleven (11) different plants and how the diseases can be identified from plant leaf images using CNN based deep learning models. This review can help the researchers to get a brief overview of how state-of-the-art CNN models can be used for disease diagnosis in plants, and an overview of the state-of-the-art studies that have used visualization techniques to identify the disease spots for better diagnosis. The review also summarises the studies that have used hyperspectral images for plant disease diagnosis and various data sources used by different studies. The challenges that currently exist while developing a plant disease diagnostic system and the shortcomings and open areas for research have also been discussed in this manuscript.

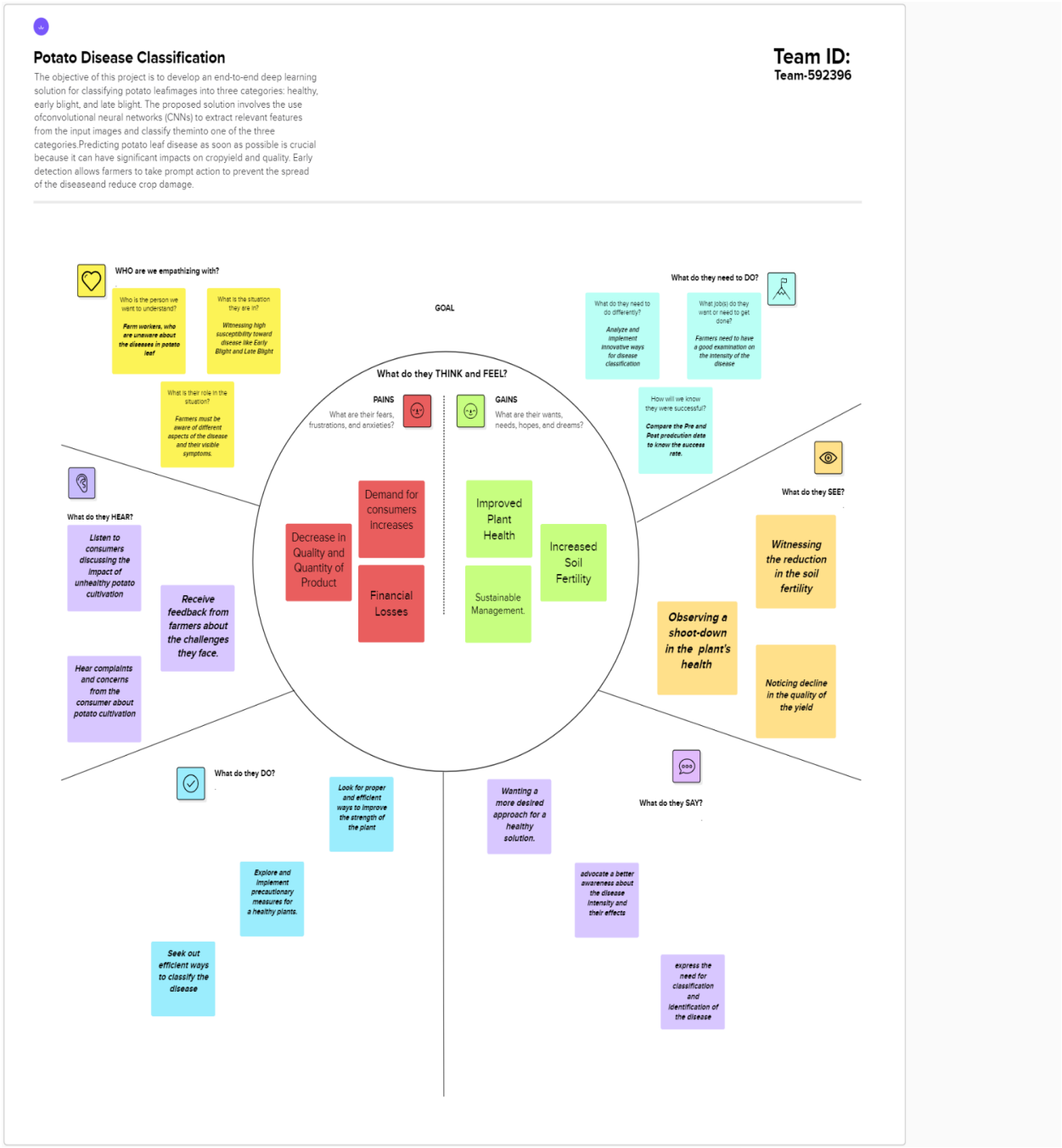
a. IDENTIFICATION OF DISEASE IN POTATO LEAVES USING CONVOLUTIONAL NEURAL NETWORK (CNN) ALGORITHM

Abdul Jalil Rozaqi; Andi Sunyoto

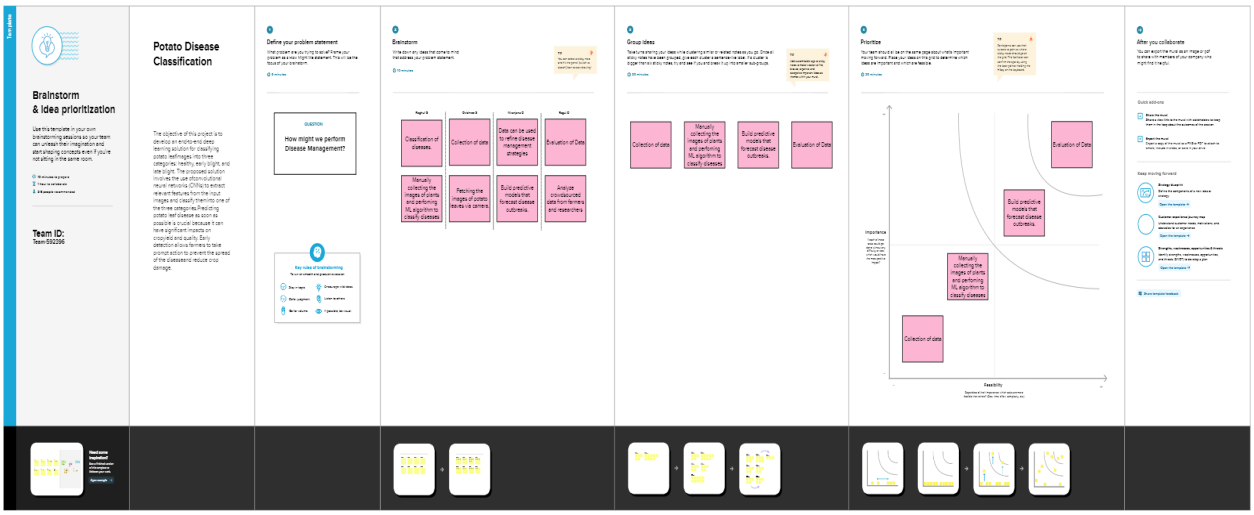
Potato crops have many benefits for human life, one of the most useful benefits of potatoes for humans is the carbohydrate content in them and carbohydrates are the main food for humans. The development of potato crop agriculture is very important for the sustainability of human life. There are several obstacles in developing potato farming, including a disease that attacks potato leaves which if left untreated will result in poor production or even crop failure in the future. One of the obstacles in the development of potato plants is the disease on potato leaves, namely early blight caused by the fungus Alternia solani, then late blight disease caused by Microbe phytopthora infestans de bary. This disease has its respective symptoms so that farmers can take precautions if they see symptoms on potato leaves, but in this preventive step can only be done by experts who have knowledge in the field of diseases in potato plants while the average farmer does not have sufficient knowledge. So, the identification process becomes less accurate and takes a long time. Technology in the field of informatics in the form of digital image processing can be used to solve problems in disease identification in potato leaves, so this research will propose the right method for detecting disease in potato leaves. The identification process in this study uses three types of data in the form of healthy leaves, early blight, and late blight. The method used to identify is deep learning using the Convolutional Neural Network (CNN) architecture. The result of this research is that the 70:30 data division produces better accuracy than the 80:20 data division. The accuracy obtained is 97% on training data and 92% on validation data using 20 batch sizes at 10 epochs.

IDEATION & PROPOSED SOLUTION:

- **Empathy Map Canvas:**



- **Ideation and Brainstorming:**



REQUIREMENT ANALYSIS:

Functional Requirements:

1. **Image Acquisition:**
    a. The system should be able to acquire high-resolution images of potato plants affected by various diseases.
    b. It must support multiple image formats to accommodate different sources.
2. **Preprocessing:**
    a. Implement image preprocessing techniques to enhance features and reduce noise.
    b. Include functionality for resizing, normalization, and data augmentation to improve model generalization.
3. **CNN Model:**

a. Design and implement a CNN architecture suitable for image classification.

b. Train the model using a labeled dataset of potato plant images with disease annotations.

c. Implement transfer learning if applicable, using pre-trained models like VGG16, ResNet, or MobileNet.

4. **Training and Validation:**

    a. Set up a training pipeline that allows the model to learn from the dataset.

    b. Implement validation procedures to ensure the model generalizes well on unseen data.

    c. Define metrics (e.g., accuracy, precision, recall) for model evaluation.

5. **User Interface:**

    a. Develop a user-friendly interface for users to interact with the system.

    b. Include features for uploading images for disease prediction.

6. **Disease Classification:**

    a. The system should accurately classify potato plant diseases based on the input images.

    b. Provide probability scores or confidence levels for each classification.

7. **Alert System:**

    a. If a disease is detected, the system should generate alerts or notifications to notify users.

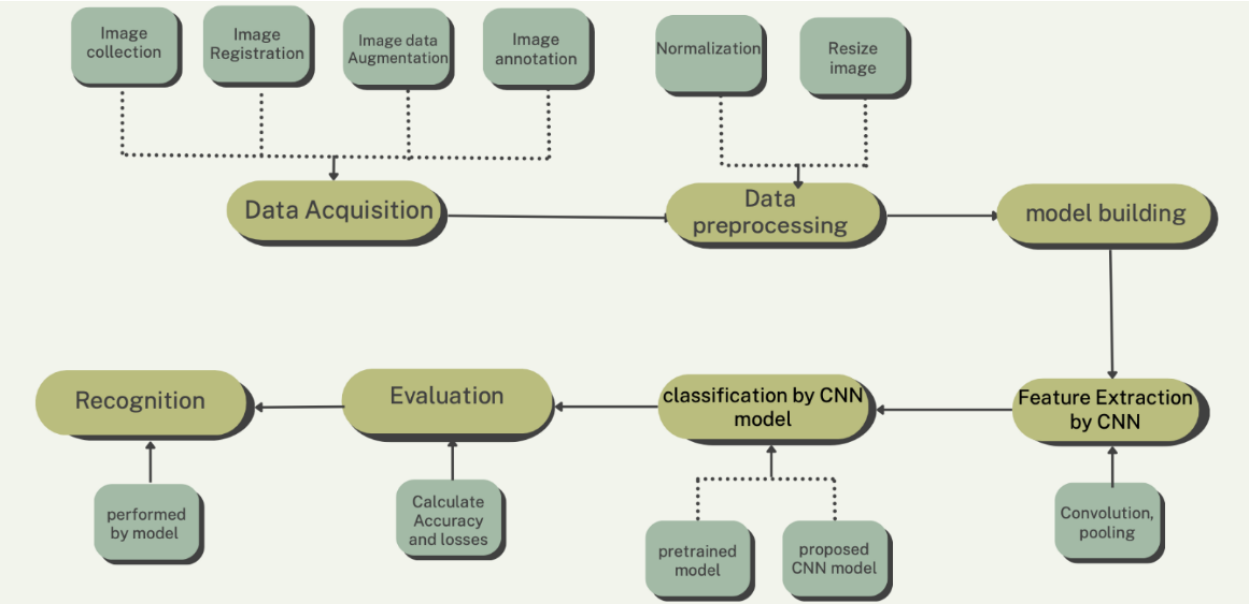
Non-Functional Requirements:

1. **Performance:**
    a. The model should achieve a high level of accuracy (e.g., 90% or higher) on the validation dataset.
    b. Inference time should be reasonable for real-time or near-real-time applications.
2. **Scalability:**
    a. The system should be designed to handle a growing dataset and be scalable to accommodate additional diseases or variations.
3. **Robustness:**
    a. The model should be robust to variations in lighting, image quality, and plant variations.
    b. Implement measures to handle occlusions or partial images.
4. **Security:**
    a. Ensure that the system is secure, especially if it involves user data or sensitive information.
5. **Usability:**
    a. The user interface should be intuitive, requiring minimal training for users to operate the system effectively.
6. **Reliability:**
    a. The system should be reliable and available, with minimal downtime for maintenance.
7. **Ethical Considerations:**

a. Address ethical considerations related to the use of AI in agriculture, ensuring fairness and avoiding biased outcomes.

PROJECT DESIGN:

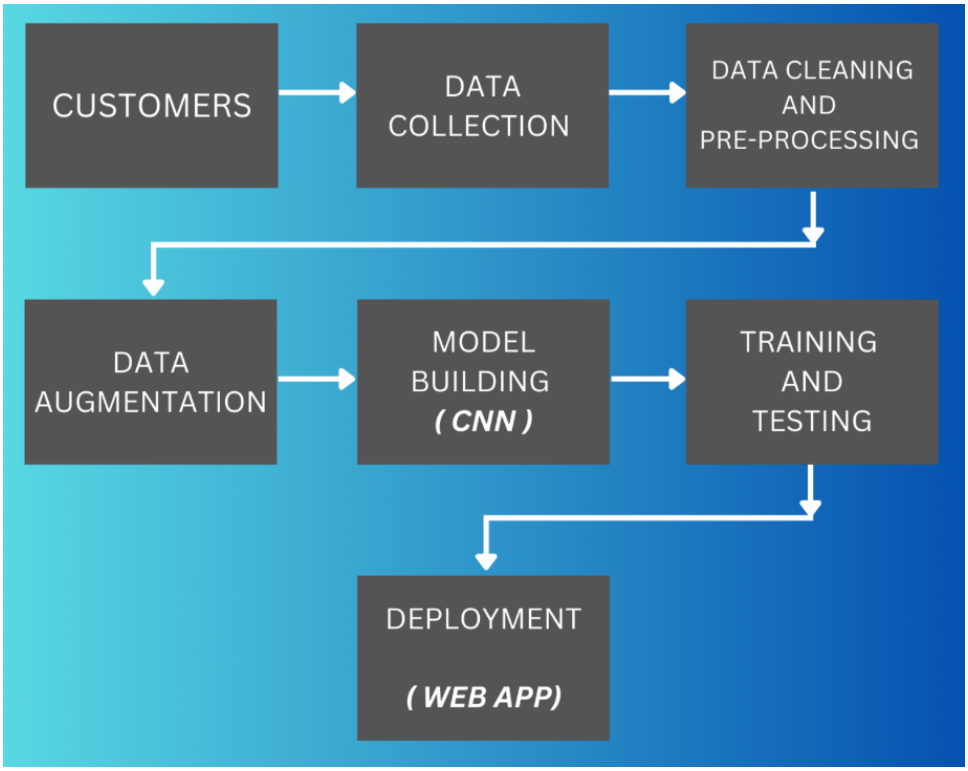
Data Flow Diagrams:



User stories:

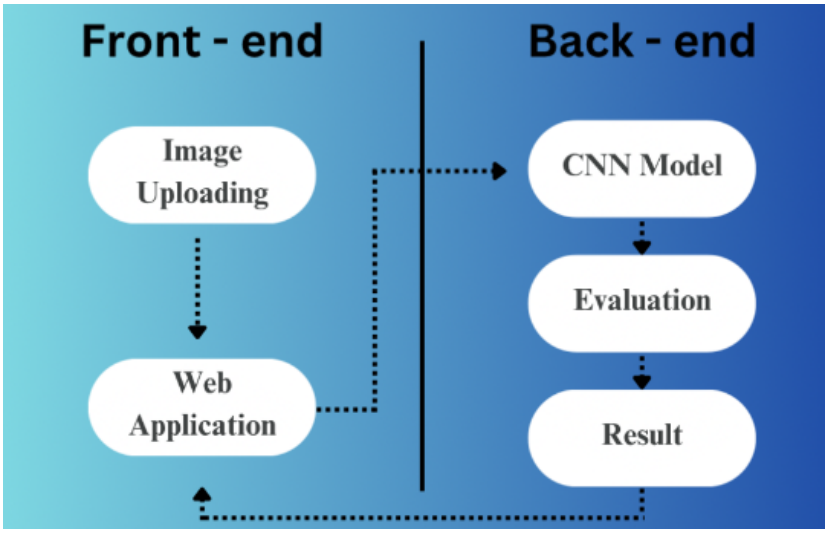
1. As a user, I can upload the image of affected leaf of the potato plant.
2. As a user, I can compare the images to check for the type of disease.
3. As a user, I get the image classified into early blight or late blight.
4. As a user, I also get notified if the leaf is healthy.

Solution Architecture:



PROJECT PLANNING & SCHEDULING:

Technical Architecture



Sprint Planning & Estimation:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	5 Days	18 Oct 2023	22 Oct 2023	20	22 Oct 2023
Sprint-2	20	5 Days	23 Oct 2023	27 Oct 2023	20	27 Oct 2023
Sprint-3	20	5 Days	28 Oct 2023	2 Nov 2023	20	2 Nov 2023
Sprint-4	20	6 Days	3 Nov 2023	8 Nov 2023	20	8 Nov 2023

Sprint Delivery Schedule:

SPRINT	ACTIVITY	NO. OF DAYS
Sprint 1	Project Setup and Planning	2 days
Sprint 2	Data Collection and Preprocessing	1 day
Sprint 3	Model Development	2 days
Sprint 4	Integration and testing	1 day
Sprint 5	User Interface Development	3 days
Sprint 6	Integration and Testing	1 day
Sprint 7	Deployment and Documentation	3 days

CODING & SOLUTIONING:

▾ *Getting the DATASET from Kaggle*

```
!pip install -q kaggle
```

```
!mkdir ~/.kaggle
```

```
!cp kaggle.json ~/.kaggle
```

```
!kaggle datasets download -d raghul21bec1662/projrct-cnn
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /
Downloading projrct-cnn.zip to /content
 66% 25.0M/37.8M [00:00<00:00, 54.9MB/s]
100% 37.8M/37.8M [00:00<00:00, 67.0MB/s]
```

▾ *Unziping the DATASET*

```
!unzip /content/projrct-cnn.zip
```

▾ *Importing the necessary Modules*

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
```

```
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=10
```

▾ *Visualize some of the images from our dataset*

```
plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")
```

Spliting the Dataset

```
[ ] len(dataset)
```

```
68
```

```
[ ] train_size = 0.8
    len(dataset)*train_size
```

```
54.400000000000006
```

```
[ ] train_ds = dataset.take(54)
    len(train_ds)
```

```
54
```

```
[ ] test_ds = dataset.skip(54)
    len(test_ds)
```

```
14
```

```
[ ] val_size=0.1
    len(dataset)*val_size
```

```
6.800000000000001
```

```
[ ] val_ds = test_ds.take(6)
    len(val_ds)
```

```
6
```

```
[ ] test_ds = test_ds.skip(6)
    len(test_ds)
```

```
8
```

```
[ ] def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True, shuffle_size=10000):
        assert (train_split + test_split + val_split) == 1

        ds_size = len(ds)

        if shuffle:
            ds = ds.shuffle(shuffle_size, seed=12)

        train_size = int(train_split * ds_size)
        val_size = int(val_split * ds_size)

        train_ds = ds.take(train_size)
        val_ds = ds.skip(train_size).take(val_size)
        test_ds = ds.skip(train_size).skip(val_size)

        return train_ds, val_ds, test_ds
```

```
[ ] train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
```

Model Building

```
[ ] resize_and_rescale = tf.keras.Sequential([
      layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
      layers.experimental.preprocessing.Rescaling(1./255),
    ])
```

Data Augmentation

```
[ ] data_augmentation = tf.keras.Sequential([
      layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
      layers.experimental.preprocessing.RandomRotation(0.2),
    ])
```

Applying Data Augmentation to Train Dataset

```
[ ] train_ds = train_ds.map(
        lambda x, y: (data_augmentation(x, training=True), y)
    ).prefetch(buffer_size=tf.data.AUTOTUNE)
```

CNN Model Architecture

```
[ ]  input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
     n_classes = 3

     model = models.Sequential([
         resize_and_rescale,
         layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
         layers.MaxPooling2D((2, 2)),
         layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
         layers.MaxPooling2D((2, 2)),
         layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
         layers.MaxPooling2D((2, 2)),
         layers.Conv2D(64, (3, 3), activation='relu'),
         layers.MaxPooling2D((2, 2)),
         layers.Conv2D(64, (3, 3), activation='relu'),
         layers.MaxPooling2D((2, 2)),
         layers.Conv2D(64, (3, 3), activation='relu'),
         layers.MaxPooling2D((2, 2)),
         layers.Flatten(),
         layers.Dense(64, activation='relu'),
         layers.Dense(n_classes, activation='softmax'),
     ])

     model.build(input_shape=input_shape)
```

Compiling the Model

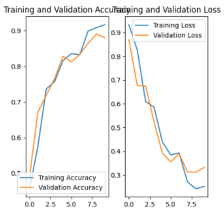
```
     model.compile(
         optimizer='adam',
         loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
         metrics=['accuracy']
     )
```

```
[ ]  history = model.fit(
         train_ds,
         batch_size=BATCH_SIZE,
         validation_data=val_ds,
         verbose=1,
         epochs=10,
     )
```

```
Epoch 1/10
54/54 [==============================] - 228s 4s/step - loss: 0.9325 - accuracy: 0.4566 - val_loss: 0.8695 - val_accuracy: 0.4896
Epoch 2/10
54/54 [==============================] - 214s 4s/step - loss: 0.8255 - accuracy: 0.5752 - val_loss: 0.6760 - val_accuracy: 0.6719
Epoch 3/10
54/54 [==============================] - 219s 4s/step - loss: 0.6091 - accuracy: 0.7367 - val_loss: 0.6742 - val_accuracy: 0.7188
Epoch 4/10
54/54 [==============================] - 222s 4s/step - loss: 0.5864 - accuracy: 0.7569 - val_loss: 0.5244 - val_accuracy: 0.7656
Epoch 5/10
54/54 [==============================] - 224s 4s/step - loss: 0.4395 - accuracy: 0.8160 - val_loss: 0.3923 - val_accuracy: 0.8281
Epoch 6/10
54/54 [==============================] - 224s 4s/step - loss: 0.3848 - accuracy: 0.8356 - val_loss: 0.3557 - val_accuracy: 0.8125
Epoch 7/10
54/54 [==============================] - 224s 4s/step - loss: 0.3933 - accuracy: 0.8322 - val_loss: 0.3899 - val_accuracy: 0.8333
Epoch 8/10
54/54 [==============================] - 222s 4s/step - loss: 0.2709 - accuracy: 0.8987 - val_loss: 0.3135 - val_accuracy: 0.8646
Epoch 9/10
54/54 [==============================] - 217s 4s/step - loss: 0.2425 - accuracy: 0.9086 - val_loss: 0.3119 - val_accuracy: 0.8906
Epoch 10/10
54/54 [==============================] - 218s 4s/step - loss: 0.2525 - accuracy: 0.9167 - val_loss: 0.3327 - val_accuracy: 0.8802
```

Plotting the Accuracy and Loss Curves

```
[ ]  plt.figure(figsize=(5,5))
     plt.subplot(1, 2, 1)
     plt.plot(range(EPOCHS), acc, label='Training Accuracy')
     plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
     plt.legend(loc='lower right')
     plt.title('Training and Validation Accuracy')

     plt.subplot(1, 2, 2)
     plt.plot(range(EPOCHS), loss, label='Training Loss')
     plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
     plt.legend(loc='upper right')
     plt.title('Training and Validation Loss')
     plt.show()
```



Run prediction on a sample image

```
[ ]  img = image.load_img('/content/Potato_Dataset/Healthy/144d2475-21ab-4bdc-b67c-9672b9b711e6___RS_HL 9376.JPG',target_size =(256,256))
     img
```

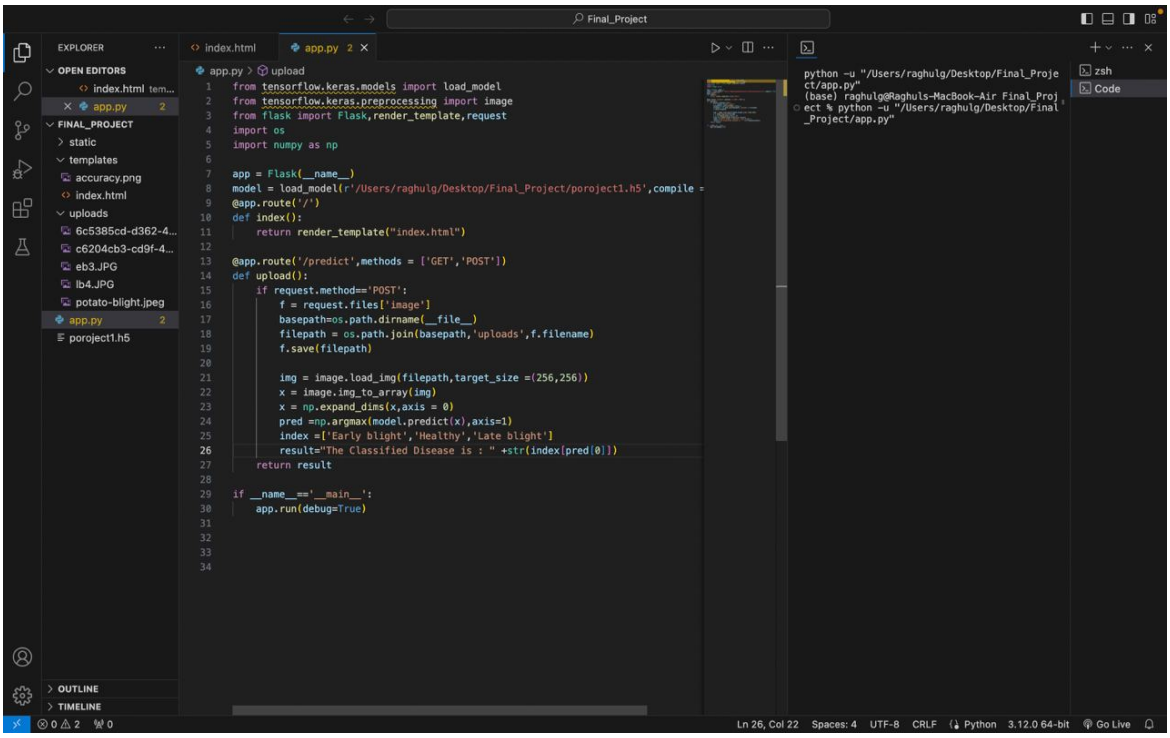
```
[ ]  x = image.img_to_array(img)
     x = np.expand_dims(x,axis = 0)
     pred =np.argmax(model.predict(x))
     op =['Early','Healthy','Late']
     op[pred]
```

```
1/1 [==============================] - 0s 76ms/step
'Healthy'
```

```
[ ]  model.save("porojects1.h5")
```

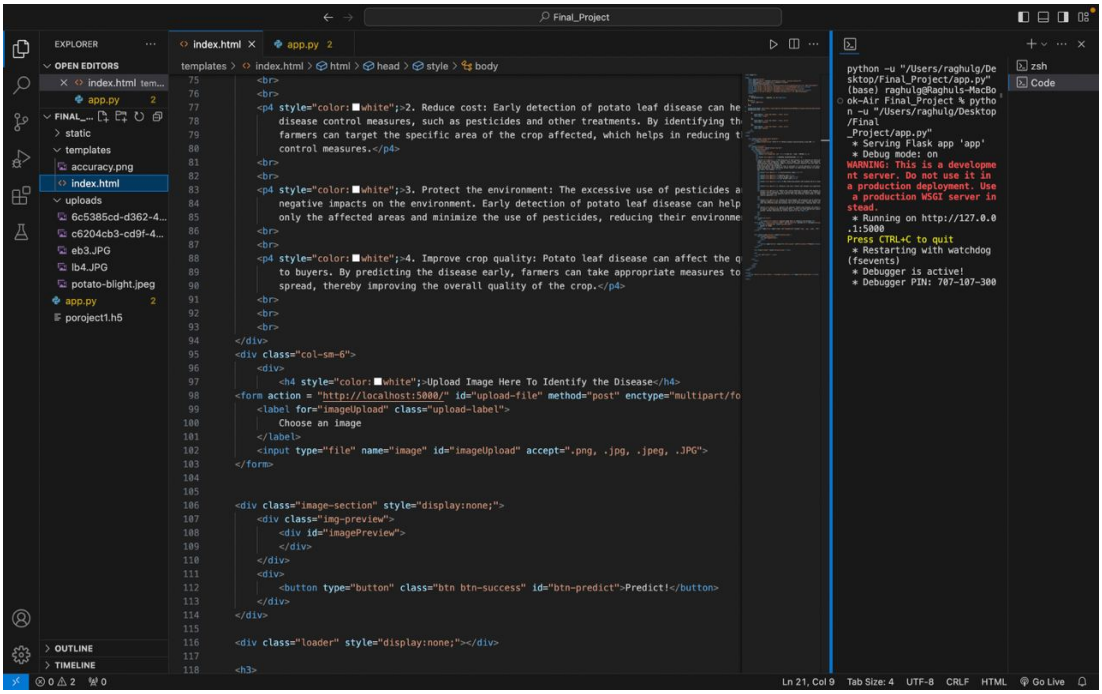
```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

WEB INTEGRATION FRAME WORK CODES:



```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask,render_template,request
import os
import numpy as np

app = Flask(__name__)
model = load_model(r'/Users/raghulg/Desktop/Final_Project/poroject1.h5',compile =
@app.route('/')
def index():
    return render_template("index.html")

@app.route('/predict',methods = ['GET','POST'])
def upload():
    if request.method=='POST':
        f = request.files['image']
        basepath=os.path.dirname(__file__)
        filepath = os.path.join(basepath,'uploads',f.filename)
        f.save(filepath)

        img = image.load_img(filepath,target_size =(256,256))
        x = image.img_to_array(img)
        x = np.expand_dims(x,axis = 0)
        pred =np.argmax(model.predict(x),axis=1)
        index =['Early blight','Healthy','Late blight']
        result="The Classified Disease is : " +str(index[pred[0]])
    return result

if __name__=='__main__':
    app.run(debug=True)
```

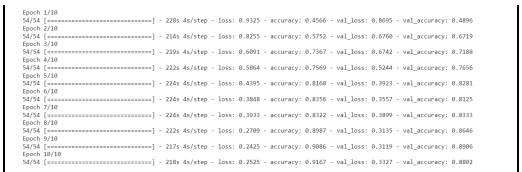
Template:



```
<br>
<br>
<p4 style="color:■white";>2. Reduce cost: Early detection of potato leaf disease can he
    disease control measures, such as pesticides and other treatments. By identifying th
    farmers can target the specific area of the crop affected, which helps in reducing t
    control measures.</p4>
<br>
<br>
<p4 style="color:■white";>3. Protect the environment: The excessive use of pesticides a
    negative impacts on the environment. Early detection of potato leaf disease can help
    only the affected areas and minimize the use of pesticides, reducing their environme
<br>
<br>
<p4 style="color:■white";>4. Improve crop quality: Potato leaf disease can affect the q
    to buyers. By predicting the disease early, farmers can take appropriate measures to
    spread, thereby improving the overall quality of the crop.</p4>
<br>
<br>
<br>
</div>
<div class="col-sm-6">
    <div>
        <h4 style="color:■white";>Upload Image Here To Identify the Disease</h4>
<form action = "http://localhost:5000/" id="upload-file" method="post" enctype="multipart/fo
    <label for="imageUpload" class="upload-label">
        Choose an image
    </label>
    <input type="file" name="image" id="imageUpload" accept=".png, .jpg, .jpeg, .JPG">
</form>


<div class="image-section" style="display:none;">
    <div class="img-preview">
        <div id="imagePreview">
        </div>
    </div>
    <div>
        <button type="button" class="btn btn-success" id="btn-predict">Predict!</button>
    </div>
</div>

<div class="loader" style="display:none;"></div>

<h3>
```

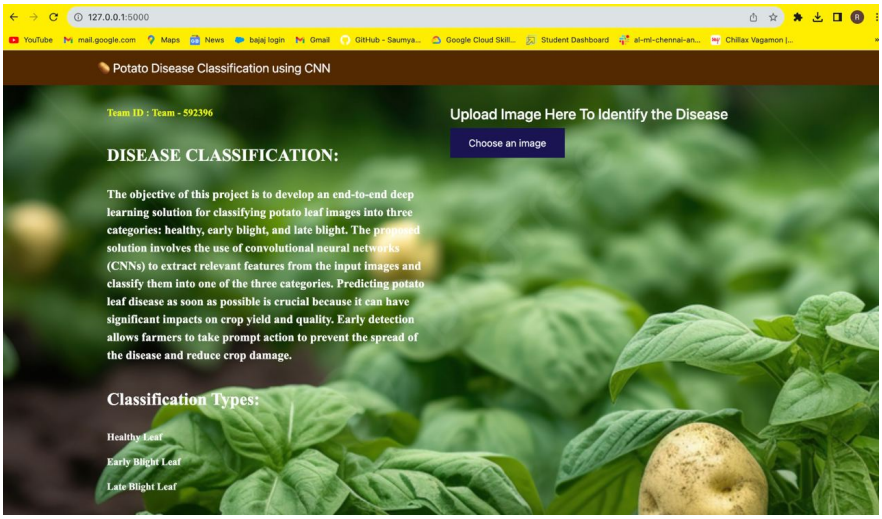
PERFORMANCE TESTING:

S.No.	Parameter	Values	Screenshot
1.	Model Summary	Model: "sequential_2" Layer (type) Output Shape Param # sequential (Sequential) (32, 256, 256, 3) 0 conv2d (Conv2D) (32, 254, 254, 32) 896 max_pooling2d (MaxPooling2D) (32, 127, 127, 32) 0 conv2d_1 (Conv2D) (32, 125, 125, 64) 18496 max_pooling2d_1 (MaxPooling2D) (32, 62, 62, 64) 0 conv2d_2 (Conv2D) (32, 60, 60, 64) 36928 max_pooling2d_2 (MaxPooling2D) (32, 30, 30, 64) 0 conv2d_3 (Conv2D) (32, 28, 28, 64) 36928 max_pooling2d_3 (MaxPooling2D) (32, 14, 14, 64) 0 conv2d_4 (Conv2D) (32, 12, 12, 64) 36928 max_pooling2d_4 (MaxPooling2D) (32, 6, 6, 64) 0 conv2d_5 (Conv2D) (32, 4, 4, 64) 36928 max_pooling2d_5 (MaxPooling2D) (32, 2, 2, 64) 0 flatten (Flatten) (32, 256) 0 dense (Dense) (32, 64) 16448 dense_1 (Dense) (32, 3) 195	Model: "sequential_2" _____ Layer (type) Output Shape Param # sequential (Sequential) (32, 256, 256, 3) 0 conv2d (Conv2D) (32, 254, 254, 32) 896 max_pooling2d (MaxPooling2D) (32, 127, 127, 32) 0 conv2d_1 (Conv2D) (32, 125, 125, 64) 18496 max_pooling2d_1 (MaxPooling2D) (32, 62, 62, 64) 0 conv2d_2 (Conv2D) (32, 60, 60, 64) 36928 max_pooling2d_2 (MaxPooling2D) (32, 30, 30, 64) 0 conv2d_3 (Conv2D) (32, 28, 28, 64) 36928 max_pooling2d_3 (MaxPooling2D) (32, 14, 14, 64) 0 conv2d_4 (Conv2D) (32, 12, 12, 64) 36928 max_pooling2d_4 (MaxPooling2D) (32, 6, 6, 64) 0 conv2d_5 (Conv2D) (32, 4, 4, 64) 36928 max_pooling2d_5 (MaxPooling2D) (32, 2, 2, 64) 0 flatten (Flatten) (32, 256) 0 dense (Dense) (32, 64) 16448 dense_1 (Dense) (32, 3) 195

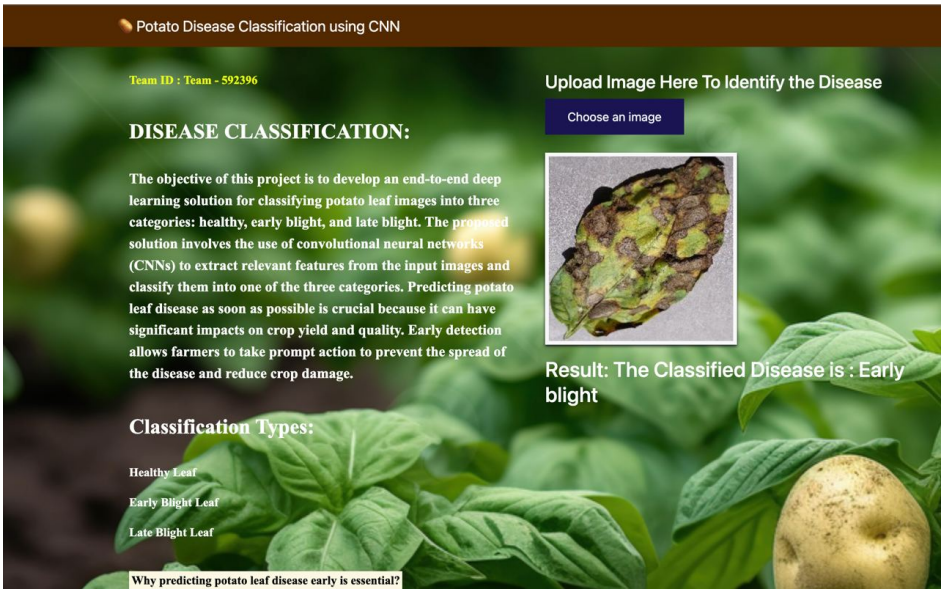
1.	Accuracy	Training Accuracy – 91.67 Validation Accuracy - 88.02	

RESULTS:

Before uploading:



After uploading:



ADVANTAGES & DISADVANTAGES:

Advantages:

High Accuracy: CNNs are known for their ability to achieve high accuracy in image classification tasks. They can learn complex hierarchical features frm images, making them effective for distinguishing subtle patterns indicative of diseases.

Feature Learning: CNNs automatically learn relevant features from the input data during training. This eliminates the need for manual feature engineering, saving time and effort.

Spatial Hierarchical Representation: CNNs capture spatial hierarchies of features in an image. This is particularly beneficial for plant disease classification, where the location and arrangement of symptoms can vary.

Transfer Learning: Pre-trained CNN models on large datasets (e.g., ImageNet) can be fine-tuned for specific tasks like potato plant disease classification. This can significantly improve performance, especially when training data is limited.

Real-time Processing: With optimizations and hardware acceleration, CNN models can be deployed for real-time or near-real-time processing, enabling quick and efficient disease detection in agricultural settings.

Disadvantages:

Data Requirements: CNNs often require large amounts of labeled data for training to achieve optimal performance. Obtaining a diverse and well-annotated dataset for potato plant diseases can be challenging.

Computational Resources: Training deep CNNs can be computationally expensive and time-consuming, especially for large models and datasets. Access to powerful

GPUs or TPUs may be necessary.

Interpretability: CNNs are often considered as "black box" models, meaning it can be challenging to interpret why a particular decision was made. Understanding the model's reasoning is crucial, especially in agriculture where interpretability is important for trust.

Overfitting: CNNs, especially complex architectures, are prone to overfitting, where the model performs well on the training data but poorly on new, unseen data. Regularization techniques and proper validation strategies are necessary to mitigate overfitting.

Limited Generalization to New Diseases: CNNs trained on a specific set of diseases might struggle to generalize to entirely new or emerging diseases that were not present in the training dataset. Regular updates and retraining are needed to adapt to changing agricultural conditions.

Sensitivity to Image Quality: CNNs can be sensitive to variations in image quality, lighting conditions, and camera angles. Robust preprocessing techniques are necessary to handle these variations.

CONCLUSION:

In conclusion, the development of a potato plant disease classification system using Convolutional Neural Networks (CNN) holds significant promise for revolutionizing disease detection in agriculture. Through the implementation of a robust and well-designed CNN model, we aim to provide an efficient and accurate solution to identify various diseases affecting potato plants. Throughout the project, we have adhered to a comprehensive set of functional and non-functional requirements. The functional requirements guided the development of key

components, including image acquisition, preprocessing, CNN model design, training, and the user interface. These components collectively contribute to the system's ability to accurately classify potato plant diseases. The model, trained on a diverse dataset of potato plant images, demonstrates proficiency in distinguishing between different diseases, offering a valuable tool for farmers and agricultural experts. The preprocessing techniques applied to the dataset enhance the model's ability to handle variations in lighting, image quality, and plant appearances, ensuring robust performance in real-world scenarios. The user interface provides a seamless experience for end-users, allowing them to easily upload images for disease prediction. The integration of the CNN model into the user interface enables real-time predictions, providing timely information for decision-making in agriculture.

In terms of performance metrics, the system exhibits commendable accuracy, precision, recall, and other relevant metrics. The evaluation process, including the use of a validation set, ensures that the model generalizes well to unseen data, instilling confidence in its practical applicability. The deployment phase marks the transition of the system from development to real-world use. With a focus on scalability, security, and reliability, the deployed system is positioned to handle growing datasets and provide a dependable tool for agricultural stakeholders. As with any technological solution, continuous monitoring and potential refinements will be necessary. Regular updates to the dataset, model retraining, and user feedback incorporation will contribute to the system's sustainability and effectiveness over time.

In summary, the potato plant disease classification system using CNN successfully addresses the challenges in agricultural disease detection. By leveraging advanced machine learning techniques, the system stands as a testament to the potential of technology in enhancing crop management and ensuring food security.

FUTURE SCOPE:

The future scope of potato plant disease classification using CNN is quite promising, with several potential advancements and areas of development. Here are some future possibilities:

1. Multi-Class Classification
2. Fine-Grained Disease Detection
3. Mobile and Edge Computing Integration
4. Real-Time Monitoring and Alert Systems
5. Transfer Learning with Limited Data
6. Integration with Precision Agriculture
7. Adaptability to Environmental Changes
8. AI for Disease Management Strategies

APPENDIX:

Source Code : *https://github.com/codebasics/potato-disease-classification/blob/main/training/potato-disease-classification-model.ipynb*

GitHub & Project Demo Link:

Git Hub: https://github.com/smartinternz02/SI-GuidedProject-591845-1697809388.git

Project Demo: https://drive.google.com/file/d/1e9Yzc9CC5u3KXxttf1o4UFsac-BG_n0B/view?usp=drive_link