POTATO DISEASE CLASSIFICATION USING DEEP LEARNING

Team ID: 592396

Objective:

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.

Why predicting potato leaf disease early is essential?

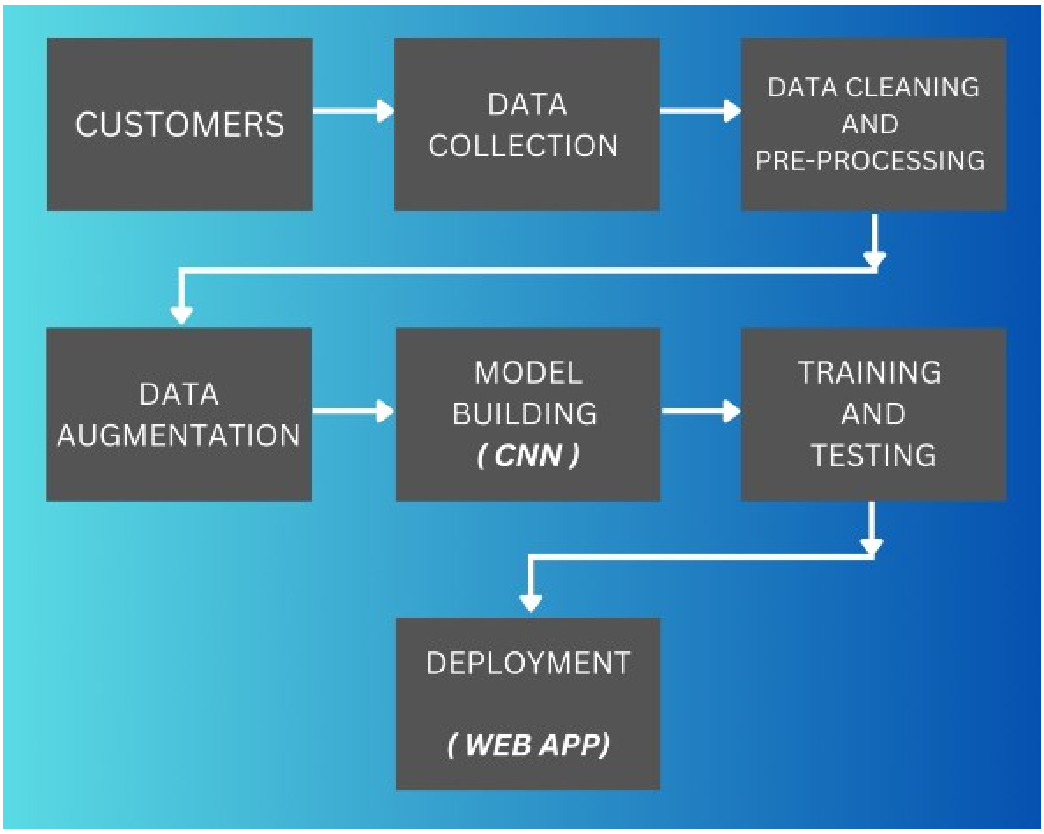
- Crop productivity and quality can be considerably decreased by potato leaf disease. Early disease detection allows farmers to minimize crop loss by controlling the disease's spread.

- 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.

Technical Architecture:



Project Flow:

Problem understanding:

The problem is to find an effective and sustainable solution to manage and control the spread of potato leaf disease while optimizing potato production without crop loss, increase in cost, reduce in crop quality and environmental damage.

Solution:

1. Identifying and classifying the healthy, early-blight and late-blight of potato plants.
2. The user interacts with the UI of web app to upload the image.

3. Uploaded input is analysed by the model below.
    - Data gathering and preprocessing
    - Data augmentation
    - Building CNN architecture
    - Training the model
    - Compile the model
    - Evaluation
    - Deployment
4. Once the model analyses the input the prediction is showcased on the UI.

Prior Knowledge:

1. VS Code
2. Deep learning concepts:
    - CNN (Convolutional Neural Network)
3. Web concepts:
    - Flask

Define Problem / Problem Understanding

Specify the business problem

The problem is to find an effective and sustainable solution to manage and control the spread of potato leaf disease while optimizing potato production without crop loss, increase in cost, reduce in crop quality and environmental damage.

Business requirements

1. Accurate prediction: The degree of leaf degradation must be precisely predicted by the predictor. For farmers, agribusinesses, and other stakeholders to make educated decisions about production, the prediction's accuracy is essential.

2. User-friendly interface: A user-friendly interface that is simple to use and comprehend is essential for the predictor. Farmer and other stakeholders should be able to make well-informed decisions by having easy access to the predictor's data through a user-friendly interface.

3. Scalability: Based on the forecast from our product, the predictor must be able to scale up. The model should be able to handle data of any scale without affecting accuracy or efficiency.

Literature Survey :

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.

Social or Business Impact:

- Improved potato crop yields.
- Reduction in production costs.
- Protect environment.
- Increase crop quality.

Data gathering and preprocessing:

This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Dataset: Link

..
.config
Potato_Dataset
sample_data
kaggle.json
poroject1.h5
projrct-cnn.zip

Disk 80.59 GB available

Potato Disease Classification

Getting the DATASET from Kaggle

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

```
[2] !mkdir ~/.kaggle
```

```
[3] !cp kaggle.json ~/.kaggle
```

```
[4] !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 r
Downloading projrct-cnn.zip to /content
 53% 20.0M/37.8M [00:00<00:00, 64.4MB/s]
100% 37.8M/37.8M [00:00<00:00, 87.9MB/s]
```

Unziping the DATASET

```
[5] !unzip /content/projrct-cnn.zip
```

```
  inflating: Potato_Dataset/Late_Blight/e48ffe4c-d920-4559-83ab-62d9d881d787___RS_LB 4129.JPG
  inflating: Potato_Dataset/Late_Blight/e4a264b9-fbae-4f38-86ce-d0a7d70e92e1___RS_LB 5211.JPG
  inflating: Potato_Dataset/Late_Blight/e4d454e4-c1e4-4597-8e76-77a93465b41d___RS_LB 2868.JPG
  inflating: Potato_Dataset/Late_Blight/e575421c-a9d4-4c0e-8027-b12fca1e3ac9___RS_LB 2806.JPG
  inflating: Potato_Dataset/Late_Blight/e6d7f5aa-d176-4e43-9e0c-afc2901442f6___RS_LB 3066.JPG
  inflating: Potato_Dataset/Late_Blight/e71a27b3-20d5-40e8-ad01-4c1942d46654___RS_LB 4085.JPG
  inflating: Potato_Dataset/Late_Blight/e749122f-0dcc-4613-87a9-5347a51a7b58___RS_LB 4409.JPG
  inflating: Potato_Dataset/Late_Blight/e766a563-c87a-4d14-ae00-898ce62e15d4___RS_LB 5123.JPG
  inflating: Potato_Dataset/Late_Blight/e7e39ba3-b9af-4d5f-afb1-aea45e13446a___RS_LB 2876.JPG
  inflating: Potato_Dataset/Late_Blight/e8040f51-18ed-43ec-9fa3-b5eb972f4fec___RS_LB 2527.JPG
```

✓ 0s completed at 14:22

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

\+ Code \+ Text

```
[7] BATCH_SIZE = 32
    IMAGE_SIZE = 256
    CHANNELS=3
    EPOCHS=10
```

Importing the DATASET into Tensorflow

```
[8] dataset = tf.keras.preprocessing.image_dataset_from_directory(
        "Potato_Dataset",
        seed=123,
        shuffle=True,
        image_size=(IMAGE_SIZE,IMAGE_SIZE),
        batch_size=BATCH_SIZE
    )
```

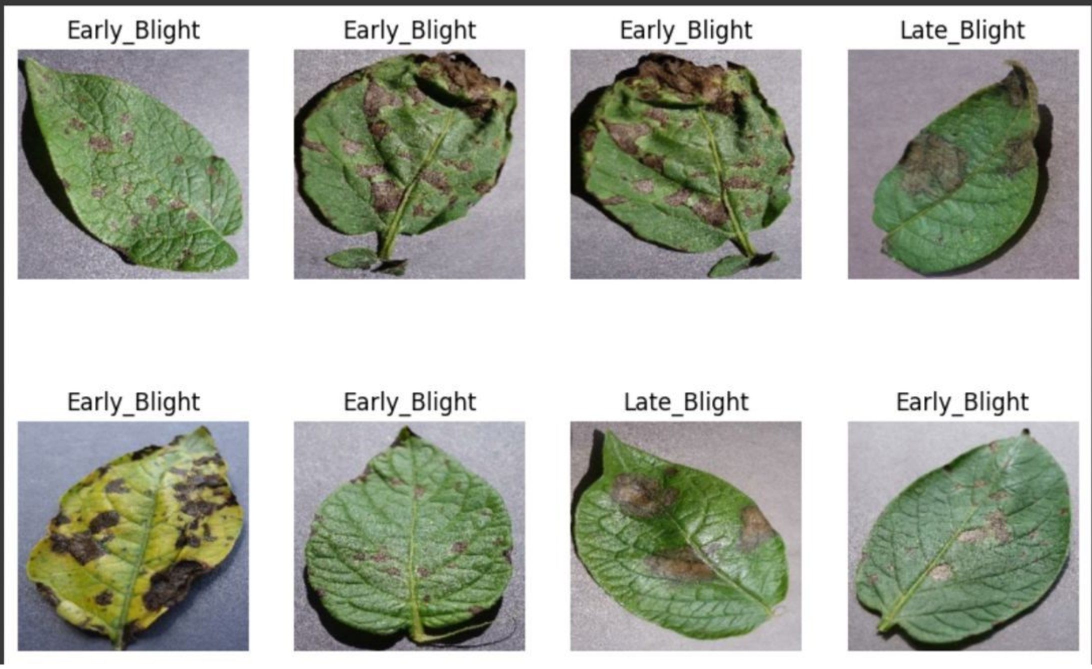
```
Found 2152 files belonging to 3 classes.
```

```
[9] class_names = dataset.class_names
    class_names
```

```
['Early_Blight', 'Healthy', 'Late_Blight']
```

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

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

```
68
```

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

```
54.400000000000006
```

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

```
54
```

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

```
14
```

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

```
6.800000000000001
```

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

```
6
```

```
[18] 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
```

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

```
[21] len(train_ds)

54
```

```
[22] len(val_ds)

6
```

Data Augmentation:

Cache, Shuffle, and Prefetch the Dataset

```
[24] train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
     val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
     test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
```

Model Building

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

Data Augmentation

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

Applying Data Augmentation to Train Dataset

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

Building CNN architecture:

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)
```

```
[29] model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (32, 256, 256, 3)         0

 conv2d (Conv2D)             (32, 254, 254, 32)        896
```

Training and compiling the model:

Compiling the Model

```
[30] 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 [==============================] - 213s 4s/step - loss: 0.8340 - accuracy: 0.5469 - val_loss: 0.6884 - val_accuracy: 0.6771
Epoch 2/10
54/54 [==============================] - 204s 4s/step - loss: 0.5991 - accuracy: 0.7407 - val_loss: 0.5507 - val_accuracy: 0.7448
Epoch 3/10
54/54 [==============================] - 203s 4s/step - loss: 0.4981 - accuracy: 0.7894 - val_loss: 0.4891 - val_accuracy: 0.7760
Epoch 4/10
54/54 [==============================] - 204s 4s/step - loss: 0.4068 - accuracy: 0.8304 - val_loss: 0.5418 - val_accuracy: 0.8021
Epoch 5/10
54/54 [==============================] - 204s 4s/step - loss: 0.3689 - accuracy: 0.8495 - val_loss: 0.3736 - val_accuracy: 0.8698
Epoch 6/10
54/54 [==============================] - 206s 4s/step - loss: 0.2505 - accuracy: 0.9010 - val_loss: 0.6769 - val_accuracy: 0.7240
Epoch 7/10
54/54 [==============================] - 199s 4s/step - loss: 0.1994 - accuracy: 0.9294 - val_loss: 0.2079 - val_accuracy: 0.9115
Epoch 8/10
54/54 [==============================] - 202s 4s/step - loss: 0.1489 - accuracy: 0.9473 - val_loss: 0.2834 - val_accuracy: 0.8854
Epoch 9/10
54/54 [==============================] - 203s 4s/step - loss: 0.1326 - accuracy: 0.9531 - val_loss: 0.1793 - val_accuracy: 0.9323
Epoch 10/10
54/54 [==============================] - 198s 4s/step - loss: 0.1210 - accuracy: 0.9560 - val_loss: 0.2076 - val_accuracy: 0.9219
```

Evaluation:

Plotting the Accuracy and Loss Curves

```
history
```

```
<keras.src.callbacks.History at 0x78260e66e1d0>
```

```
history.params
```

```
{'verbose': 1, 'epochs': 10, 'steps': 54}
```

```
history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
type(history.history['loss'])
```

```
list
```

```
len(history.history['loss'])
```

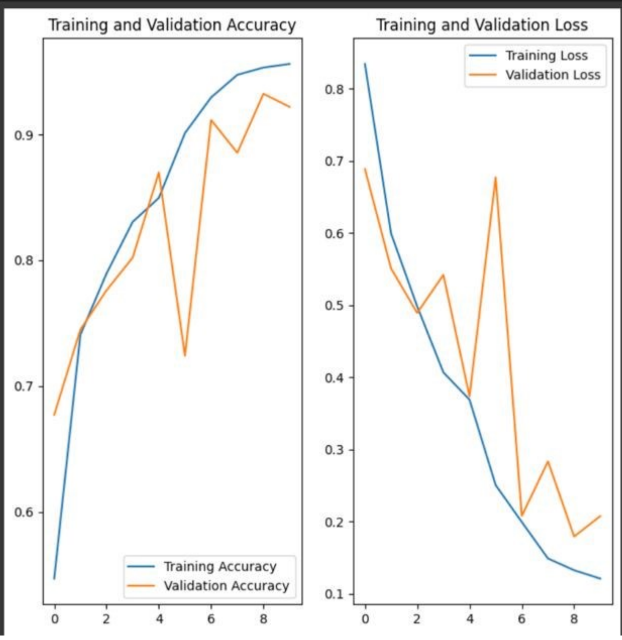
```
10
```

```
history.history['loss'][:5] # show loss for first 5 epochs
```

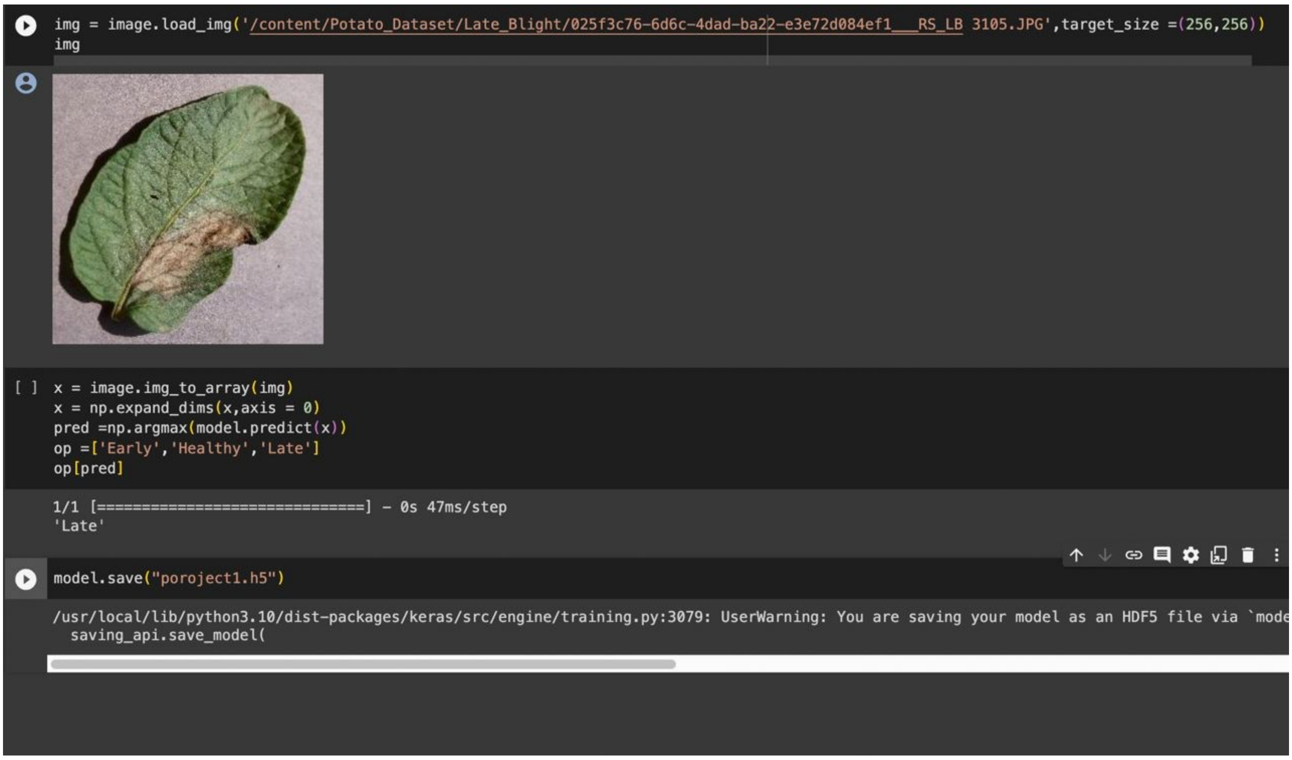
```
[0.8340210914611816,
 0.5991488695144653,
 0.49809300899505615,
 0.4067521393299103,
 0.3688788115978241]
```

```
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()
```

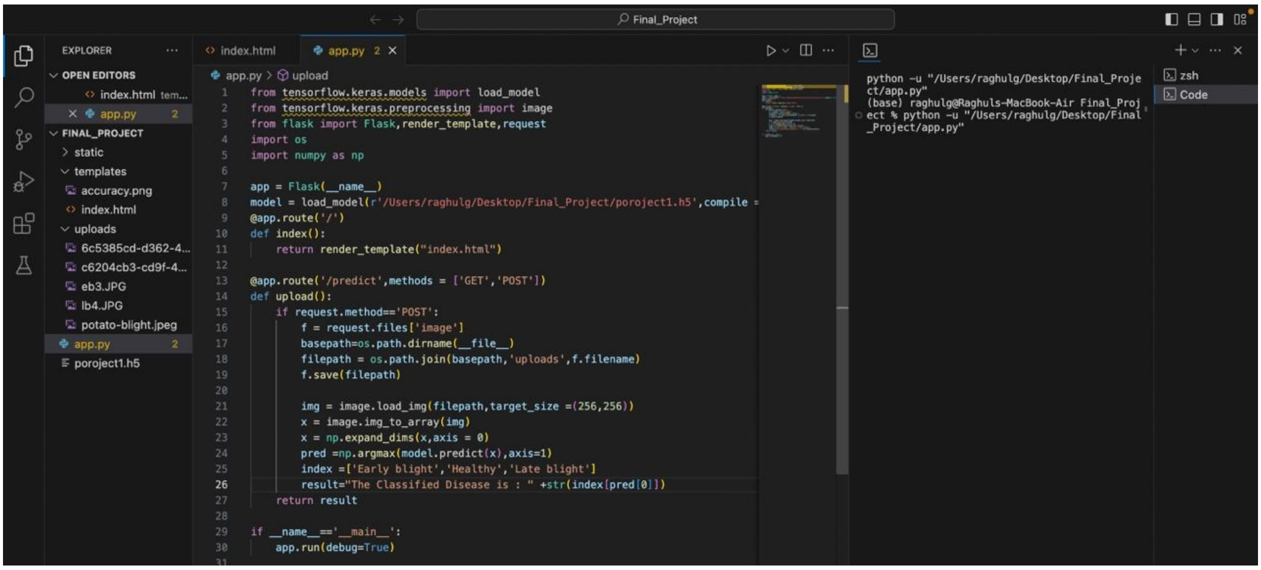


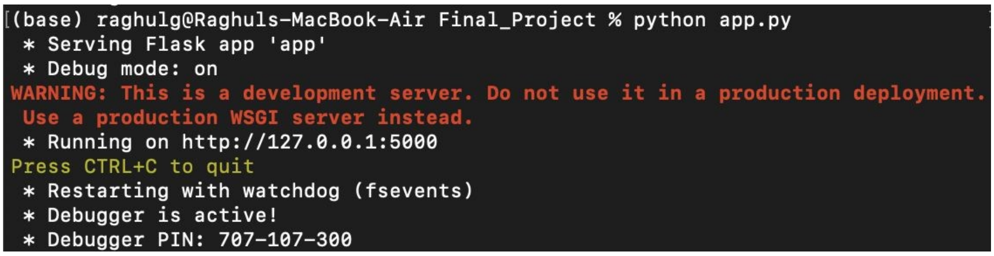
Deployment:



Integration with web frame work:

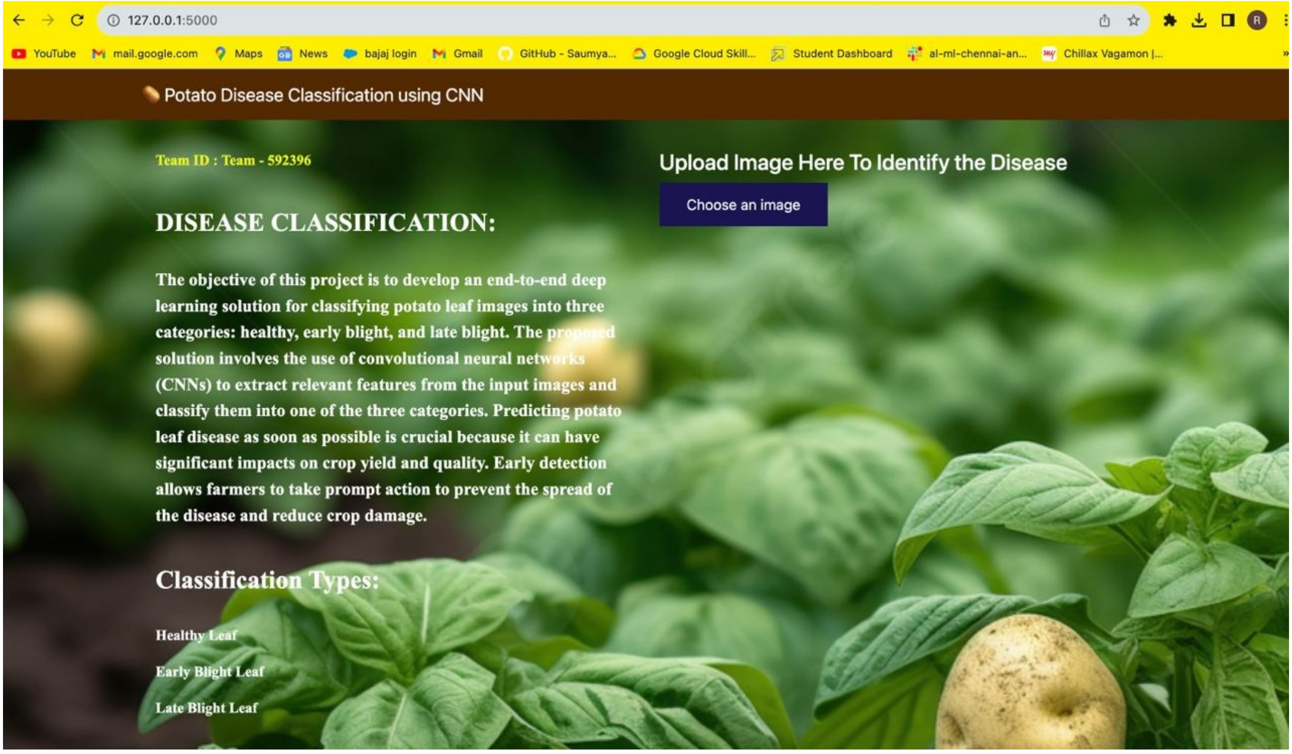
Flask file



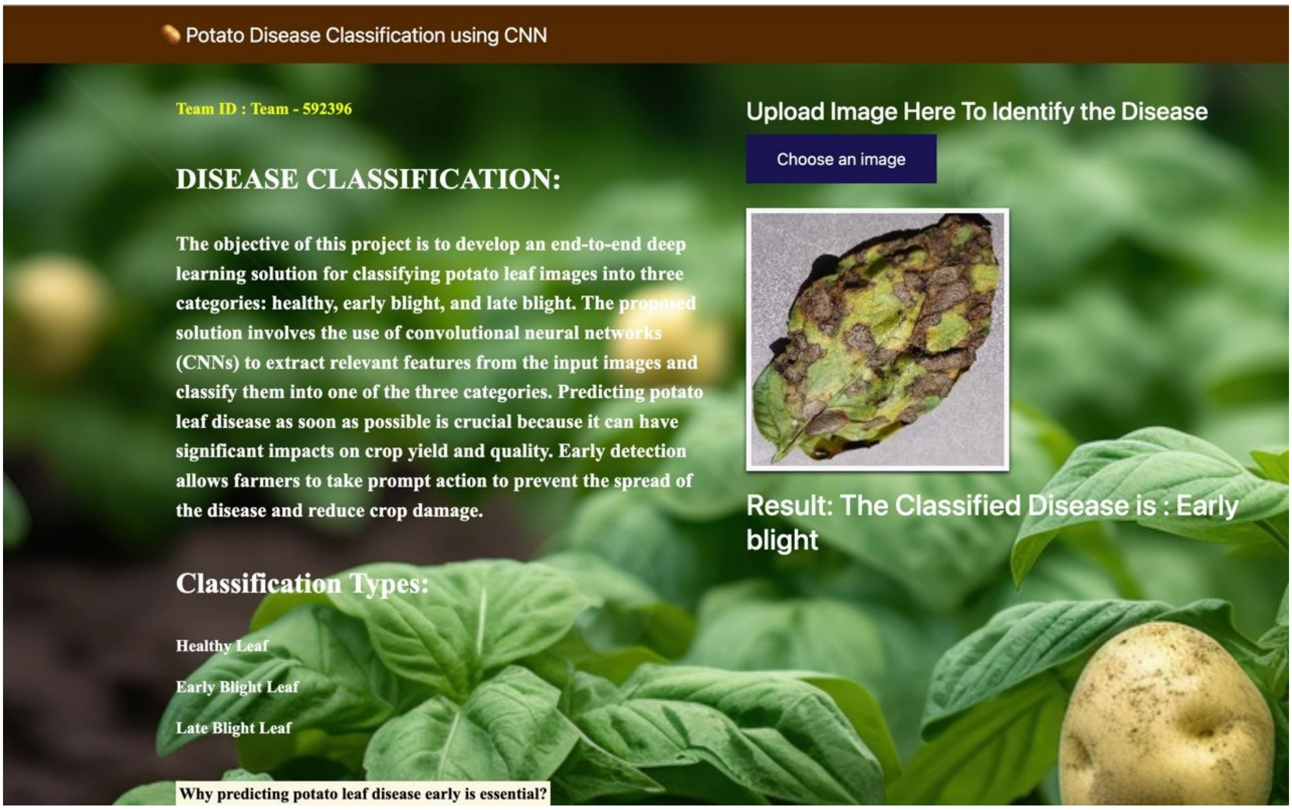


Web Application

Before uploading



After uploading



Project files:

CNN Model : Potato_Disese_Project (1).ipynb - Google Drive

Flask file: Final_Project.zip - Google Drive

Team lead:

Raghul G - 21BEC1662

Team members:

Ragul D – 21BEC1625

Niranjana D – 21BEC1655

Ovishree S – 21BEC1692

THANK YOU