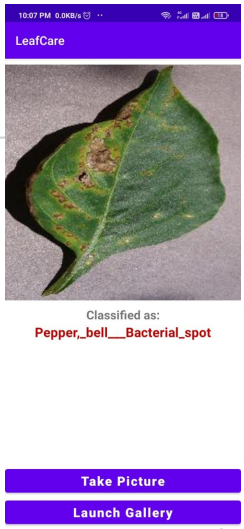
Development of a Deep Learning Model for Automated Detection and Diagnosis of Plant Diseases



In [ ]:

```
import pandas as pd
import pathlib
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import glob
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Flatten, Dense,
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.regularizers import l2
import os

%matplotlib inline
from glob import glob
import seaborn as sns
from PIL import Image
np.random.seed(11)
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, KFold, cross_val_score, GridSea
from sklearn.metrics import accuracy_score
import itertools

import keras
from keras.utils.np_utils import to_categorical
from keras.models import Sequential, Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras import backend as K
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.optimizers import Adam, RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau
from keras.wrappers.scikit_learn import KerasClassifier
from keras.applications.inception_v3 import InceptionV3
from keras import backend as K
import random
import urllib.request
import matplotlib.image as mpimg

from skimage.filters import rank, threshold_otsu
from skimage import io
from skimage.color import rgb2gray
from sklearn.cluster import KMeans
from skimage.morphology import closing, square, disk
```

Loading Data

In [2]:

```
data_dir = pathlib.Path("../input/plantvillage-dataset/color")
train='../input/plantvillage-dataset/color'
```

In [3]:

```
dataset_path_train = os.listdir(data_dir)
print (dataset_path_train)
print("Types of classes labels found: ", len(dataset_path_train))
```

```
['Tomato___Late_blight', 'Tomato___healthy', 'Grape___healthy', 'Orange___
Haunglongbing_(Citrus_greening)', 'Soybean___healthy', 'Squash___Powdery_m
ildew', 'Potato___healthy', 'Corn_(maize)___Northern_Leaf_Blight', 'Tomato
___Early_blight', 'Tomato___Septoria_leaf_spot', 'Corn_(maize)___Cercospor
a_leaf_spot Gray_leaf_spot', 'Strawberry___Leaf_scorch', 'Peach___health
y', 'Apple___Apple_scab', 'Tomato___Tomato_Yellow_Leaf_Curl_Virus', 'Tomat
o___Bacterial_spot', 'Apple___Black_rot', 'Blueberry___healthy', 'Cherry_
(including_sour)___Powdery_mildew', 'Peach___Bacterial_spot', 'Apple___Ced
ar_apple_rust', 'Tomato___Target_Spot', 'Pepper,_bell___healthy', 'Grape__
_Leaf_blight_(Isariopsis_Leaf_Spot)', 'Potato___Late_blight', 'Tomato___To
mato_mosaic_virus', 'Strawberry___healthy', 'Apple___healthy', 'Grape___Bl
ack_rot', 'Potato___Early_blight', 'Cherry_(including_sour)___healthy', 'C
orn_(maize)___Common_rust_', 'Grape___Esca_(Black_Measles)', 'Raspberry___
healthy', 'Tomato___Leaf_Mold', 'Tomato___Spider_mites Two-spotted_spider_
mite', 'Pepper,_bell___Bacterial_spot', 'Corn_(maize)___healthy']
Types of classes labels found:  38
```

In [4]:

```
image_count_train = len(list(data_dir.glob('*/*.JPG')))
print("The number of Train data:",image_count_train)
```

```
The number of Train data: 52803
```

In [5]:

```
# This Parameter we can use it in the network and model
batch_size = 32
img_height = 224
img_width = 224
```

In [6]:

```
#train data set
train_ds = image_dataset_from_directory(data_dir,
                                        seed = 123,
                                        image_size=(img_height, img_width),
                                        validation_split=0.2,
                                        subset='training')
```

```
Found 54305 files belonging to 38 classes.
Using 43444 files for training.
```

In [7]:

```
#validate data
val_ds = image_dataset_from_directory(data_dir,
                                      seed = 123,
                                      image_size=(img_height, img_width),
                                      validation_split=0.2,
                                      subset='validation')
```

```
Found 54305 files belonging to 38 classes.
Using 10861 files for validation.
```

In [8]:

```
# test data
test = image_dataset_from_directory(data_dir,
                                      seed = 123,
                                      image_size=(img_height, img_width),
                                      validation_split=None)
```

```
Found 54305 files belonging to 38 classes.
```

In [9]:

```
# Here We give the name of the data the we found the Benign dataset and malignant
class_names = train_ds.class_names
print(len(class_names))
print( class_names)

```

```
38
['Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust', 'A
pple___healthy', 'Blueberry___healthy', 'Cherry_(including_sour)___Powdery
_mildew', 'Cherry_(including_sour)___healthy', 'Corn_(maize)___Cercospora_
leaf_spot Gray_leaf_spot', 'Corn_(maize)___Common_rust_', 'Corn_(maize)___
Northern_Leaf_Blight', 'Corn_(maize)___healthy', 'Grape___Black_rot', 'Gra
pe___Esca_(Black_Measles)', 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
'Grape___healthy', 'Orange___Haunglongbing_(Citrus_greening)', 'Peach___Ba
cterial_spot', 'Peach___healthy', 'Pepper,_bell___Bacterial_spot', 'Peppe
r,_bell___healthy', 'Potato___Early_blight', 'Potato___Late_blight', 'Pota
to___healthy', 'Raspberry___healthy', 'Soybean___healthy', 'Squash___Powde
ry_mildew', 'Strawberry___Leaf_scorch', 'Strawberry___healthy', 'Tomato___
Bacterial_spot', 'Tomato___Early_blight', 'Tomato___Late_blight', 'Tomato_
__Leaf_Mold', 'Tomato___Septoria_leaf_spot', 'Tomato___Spider_mites Two-sp
otted_spider_mite', 'Tomato___Target_Spot', 'Tomato___Tomato_Yellow_Leaf_C
url_Virus', 'Tomato___Tomato_mosaic_virus', 'Tomato___healthy']
```

In [10]:

```
for image_batch, labels_batch in train_ds.take(1):
    print(image_batch.shape)
    print(labels_batch.shape)
```

```
(32, 224, 224, 3)
(32,)
```

Creat Model

In [11]:

```
num_classes = 38

model = Sequential()
model.add(layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height
model.add(Conv2D(16, 3, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D())

model.add(Conv2D(32, 3, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D())

model.add(Conv2D(64, 3, padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D())
model.add(Dropout(0.15))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('Softmax'))
model.add(Dense(num_classes))
```

In [12]:

```
# Compile the model
model.compile(optimizer='adam',
              loss=SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [13]:

```
1 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling (Rescaling)        (None, 224, 224, 3)       0
_________________________________________________________________
conv2d (Conv2D)              (None, 224, 224, 16)      448
_________________________________________________________________
activation (Activation)      (None, 224, 224, 16)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      4640
_________________________________________________________________
activation_1 (Activation)    (None, 112, 112, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 64)        18496
_________________________________________________________________
activation_2 (Activation)    (None, 56, 56, 64)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
dropout (Dropout)            (None, 28, 28, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0
_________________________________________________________________
dense (Dense)                (None, 128)               6422656
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 38)                4902
=================================================================
Total params: 6,451,142
Trainable params: 6,451,142
Non-trainable params: 0
_________________________________________________________________
```

In [15]:

```
epochs = 10

history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/10
1186/1358 [=========================>....] - ETA: 1:06 - loss: 1.0078 - ac
curacy: 0.7149

Cleanup called...

1358/1358 [==============================] - 567s 417ms/step - loss: 0.947
9 - accuracy: 0.7303 - val_loss: 0.4355 - val_accuracy: 0.8683
Epoch 2/10
1186/1358 [=========================>....] - ETA: 1:04 - loss: 0.3279 - ac
curacy: 0.8966

Cleanup called...

1358/1358 [==============================] - 543s 400ms/step - loss: 0.320
6 - accuracy: 0.8987 - val_loss: 0.3058 - val_accuracy: 0.9056
Epoch 3/10
1186/1358 [=========================>....] - ETA: 1:04 - loss: 0.1712 - ac
curacy: 0.9454

Cleanup called...

1358/1358 [==============================] - 544s 400ms/step - loss: 0.167
8 - accuracy: 0.9460 - val_loss: 0.3885 - val_accuracy: 0.8882
Epoch 4/10
1186/1358 [=========================>....] - ETA: 1:03 - loss: 0.1138 - ac
curacy: 0.9613

Cleanup called...

1358/1358 [==============================] - 540s 398ms/step - loss: 0.109
3 - accuracy: 0.9631 - val_loss: 0.3530 - val_accuracy: 0.9100
Epoch 5/10
1186/1358 [=========================>....] - ETA: 1:04 - loss: 0.0895 - ac
curacy: 0.9697

Cleanup called...

1358/1358 [==============================] - 542s 399ms/step - loss: 0.089
9 - accuracy: 0.9698 - val_loss: 0.4175 - val_accuracy: 0.8948
Epoch 6/10
1186/1358 [=========================>....] - ETA: 1:03 - loss: 0.0747 - ac
curacy: 0.9751

Cleanup called...

1358/1358 [==============================] - 540s 397ms/step - loss: 0.075
6 - accuracy: 0.9754 - val_loss: 0.3449 - val_accuracy: 0.9175
Epoch 7/10
1186/1358 [=========================>....] - ETA: 1:04 - loss: 0.0560 - ac
curacy: 0.9820

Cleanup called...
```

```
1358/1358 [==============================] - 541s 398ms/step - loss: 0.057
4 - accuracy: 0.9816 - val_loss: 0.3413 - val_accuracy: 0.9188
Epoch 8/10
1186/1358 [========================>....] - ETA: 1:04 - loss: 0.0547 - ac
curacy: 0.9823

Cleanup called...

1358/1358 [==============================] - 546s 402ms/step - loss: 0.052
7 - accuracy: 0.9830 - val_loss: 0.3569 - val_accuracy: 0.9200
Epoch 9/10
1186/1358 [========================>....] - ETA: 1:04 - loss: 0.0501 - ac
curacy: 0.9843

Cleanup called...

1358/1358 [==============================] - 547s 403ms/step - loss: 0.051
1 - accuracy: 0.9839 - val_loss: 0.4819 - val_accuracy: 0.9010
Epoch 10/10
1186/1358 [========================>....] - ETA: 1:04 - loss: 0.0461 - ac
curacy: 0.9847

Cleanup called...

1358/1358 [==============================] - 544s 401ms/step - loss: 0.042
3 - accuracy: 0.9860 - val_loss: 0.4520 - val_accuracy: 0.9125
```

In [16]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

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

In [17]:

```
preformance = model.evaluate(val_ds)
```

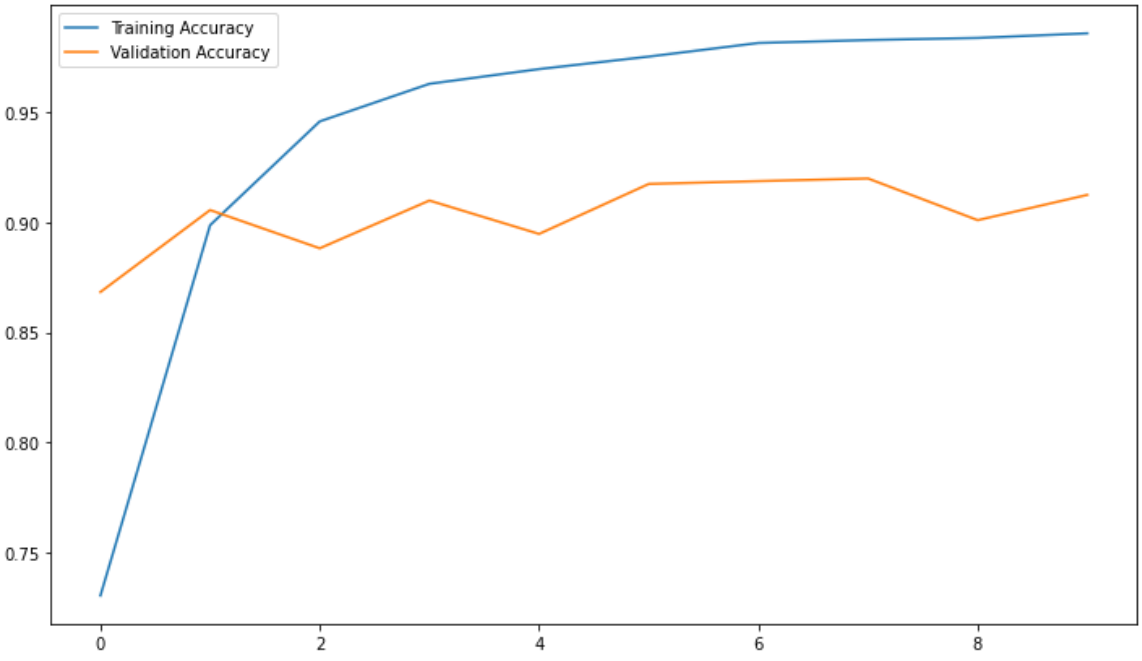
```
340/340 [==============================] - 37s 108ms/step - loss: 0.4520 -
accuracy: 0.9125
```

In [ ]:

```
preformance
```

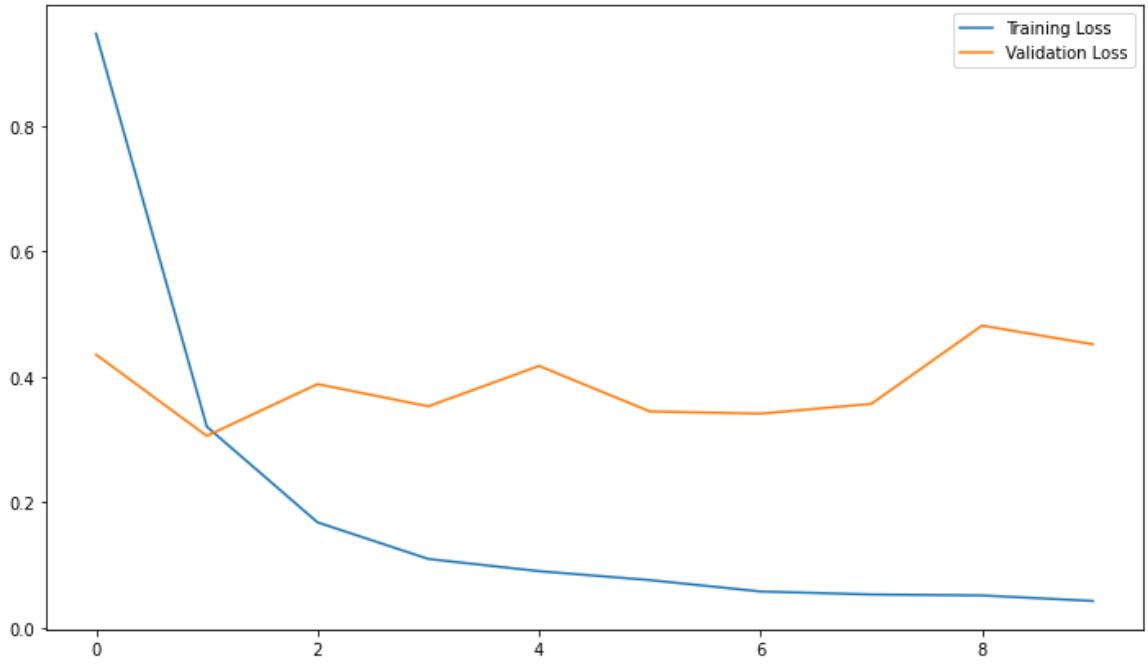
In [18]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
no_of_epochs = history.params['epochs']
len(history.history['accuracy']) == no_of_epochs
plt.figure(figsize=(12, 7))

plt.plot(range(no_of_epochs), acc, label='Training Accuracy')
plt.plot(range(no_of_epochs), val_acc, label='Validation Accuracy')

plt.legend()
plt.show()
```



In [19]:

```
plt.figure(figsize=(12, 7))

plt.plot(range(no_of_epochs), loss, label='Training Loss')
plt.plot(range(no_of_epochs), val_loss, label='Validation Loss')

plt.legend()
plt.show()
```



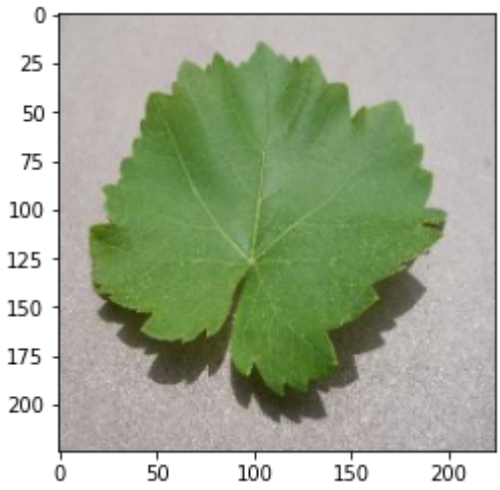
In [ ]:

```
1
```

In [20]:

```
labels = train_ds.class_names
labels
for images_batch, labels_batch in test.take(1):

    image = images_batch[0].numpy().astype('uint8')
    label = labels_batch[0].numpy()

    plt.imshow(image)
    print("actual label:", labels[label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:", labels[np.argmax(batch_prediction[0])])
```

```
actual label: Grape___healthy
predicted label: Grape___healthy
```

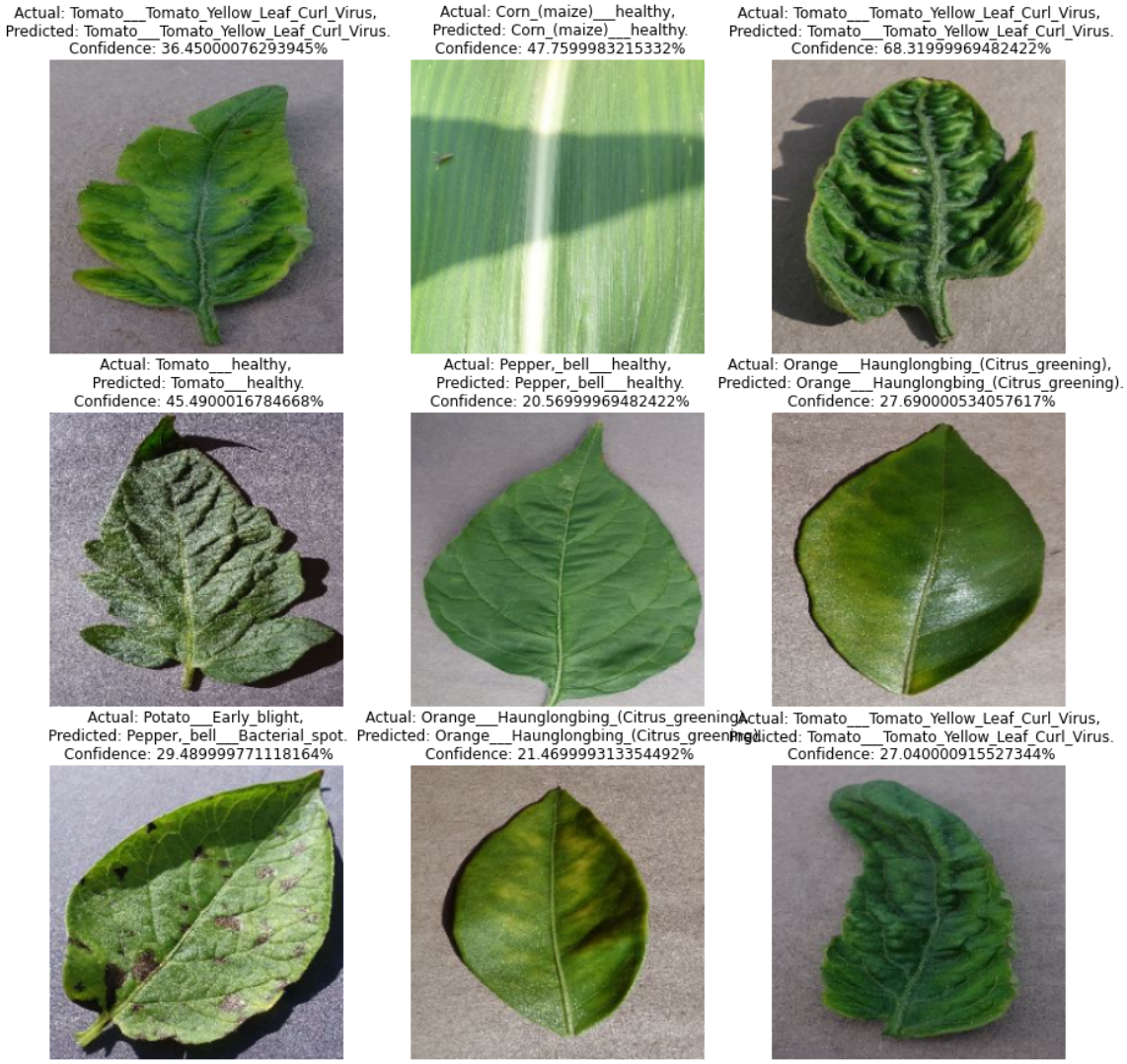


In [22]:

```
from tensorflow import expand_dims, newaxis

def predict(model, img):

    img_array = img.numpy()
    img_array = expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = labels[np.argmax(predictions[0])]
    confidence = round( (np.max(predictions[0])), 2)

    return predicted_class, confidence
```

In [23]:

```
plt.figure(figsize=(15, 15))

for images, lbs in val_ds.take(1):
    for i in range(9):

        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint32"))

        predicted_class, confidence = predict(model, images[i])
        actual_class = labels[lbs[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confide

        plt.axis("off")
```

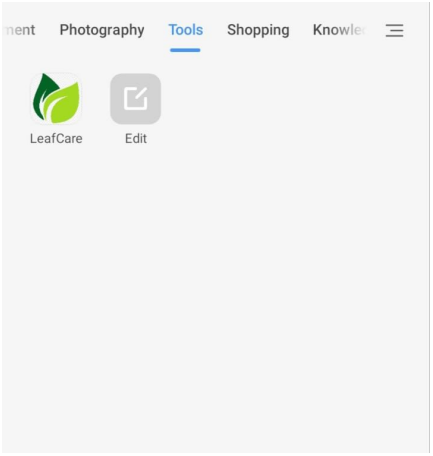


In [25]:

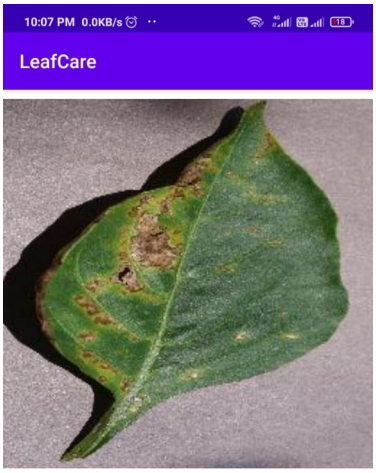
```
model.save("leafModel.h5")
```

In [ ]:

```

```



App Logo



Classified as:
Pepper,_bell___Bacterial_spot

Take Picture

Launch Gallery