In [5]:

```
!pip install tensorflow
```

```
...
```

In [6]:

```
!pip install keras
```

```
Requirement already satisfied: keras in c:\users\ramir\anaconda3\lib\site-
packages (2.13.1)
```

In [134]:

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

In [135]:

```
IMAGE_SIZE = 256
BATCH_SIZE = 32
CHANNELS = 6
EPOCHS = 20
```

In [136]:

```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    r"C:\Users\ramir\Downloads\FAST TRACK FALL SEM(2023-24)\DL\DL Project\PlantVillage",
    shuffle = True,
    image_size = (IMAGE_SIZE,IMAGE_SIZE),
    batch_size = BATCH_SIZE
)
```

```
Found 6243 files belonging to 6 classes.
```

In [138]:

```
class_names = dataset.class_names
class_names
```

Out[138]:

```
['Potato___Early_blight',
 'Potato___Late_blight',
 'Potato___healthy',
 'Tomato_Bacterial_spot',
 'Tomato__Tomato_mosaic_virus',
 'Tomato_healthy']
```

In [139]:

```
len(dataset)
```

Out[139]:

```
196
```

In [140]:

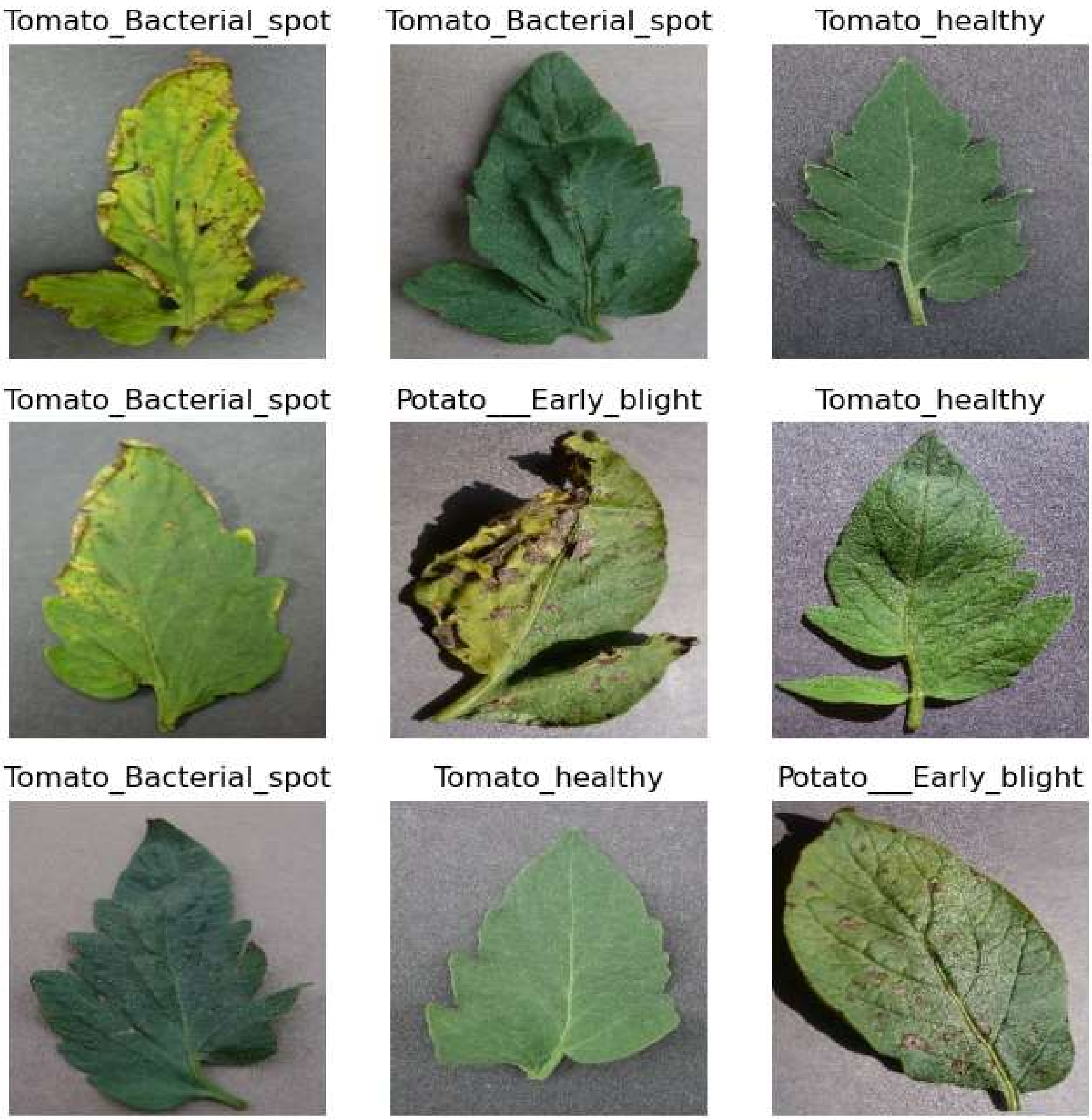
```
for image_batch,labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())
    break
```

```
(32, 256, 256, 3)
[3 3 3 1 5 3 0 5 0 0 5 0 0 1 2 1 1 0 5 5 0 5 5 4 0 0 3 0 5 3 0 5]
```

Visualize the data

In [143]:

```
plt.figure(figsize = (8,8))
for images,labels in dataset.take(1):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```



In [144]:

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

Out[144]:

```
156.8
```

In [ ]:

```

```

In [145]:

```
train_ds = dataset.take(156)
len(train_ds)
```

Out[145]:

```
156
```

In [146]:

```
test_ds = dataset.skip(156)
len(test_ds)
```

Out[146]:

```
40
```

In [147]:

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

Out[147]:

```
19.6
```

In [148]:

```
val_ds = test_ds.take(19)
len(val_ds)
```

Out[148]:

```
19
```

In [149]:

```
test_ds = test_ds.skip(19)
len(test_ds)
```

Out[149]:

```
21
```

In [151]:

```
def datset_partitions(ds,train_split = 0.8,
                      val_split=0.1,test_ds = 0.1,
                     shuffle = True,shuffle_size=10000):
    ds_size = len(ds)

    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
```

In [152]:

```
train_ds,val_ds,test_ds = datset_partitions(dataset)
```

Pre-processing

In [153]:

```
resize_and_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Resizing(256,256),
    layers.experimental.preprocessing.Rescaling(1.0/255)
])
```

In [154]:

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

In [ ]:

```

```

In [ ]:

```

```

Building a model

In [155]:

```
input_shape = (32,256,256,3)
n_classes = 6

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

model.build(input_shape=input_shape)
```

Model Architecture

In [156]:

```
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 sequential_10 (Sequential)  (32, 256, 256, 3)         0         
                                                                 
 sequential_11 (Sequential)  (32, 256, 256, 3)         0         
                                                                 
 conv2d_15 (Conv2D)          (32, 254, 254, 32)        896       
                                                                 
 max_pooling2d_15 (MaxPooli  (32, 127, 127, 32)        0         
 ng2D)                                                           
                                                                 
 conv2d_16 (Conv2D)          (32, 125, 125, 64)        18496     
                                                                 
 max_pooling2d_16 (MaxPooli  (32, 62, 62, 64)          0         
 ng2D)                                                           
                                                                 
 conv2d_17 (Conv2D)          (32, 60, 60, 64)          36928     
                                                                 
 max_pooling2d_17 (MaxPooli  (32, 30, 30, 64)          0         
 ng2D)                                                           
                                                                 
 conv2d_18 (Conv2D)          (32, 28, 28, 64)          36928     
                                                                 
 max_pooling2d_18 (MaxPooli  (32, 14, 14, 64)          0         
 ng2D)                                                           
                                                                 
 conv2d_19 (Conv2D)          (32, 12, 12, 64)          36928     
                                                                 
 max_pooling2d_19 (MaxPooli  (32, 6, 6, 64)            0         
 ng2D)                                                           
                                                                 
 flatten_3 (Flatten)         (32, 2304)                0         
                                                                 
 dense_6 (Dense)             (32, 64)                  147520    
                                                                 
 dense_7 (Dense)             (32, 6)                   390       
                                                                 
=================================================================
Total params: 278086 (1.06 MB)
Trainable params: 278086 (1.06 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [157]:

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

In [158]:

```
EPOCHS = 20
BATCH_SIZE=32
history = model.fit(
    train_ds,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    verbose=1,
    validation_data = val_ds
)
```

```
Epoch 1/20
156/156 [==============================] - 418s 3s/step - loss: 1.3016 - a
ccuracy: 0.4960 - val_loss: 1.3751 - val_accuracy: 0.5493
Epoch 2/20
156/156 [==============================] - 399s 3s/step - loss: 0.5260 - a
ccuracy: 0.8049 - val_loss: 1.1042 - val_accuracy: 0.6941
Epoch 3/20
156/156 [==============================] - 403s 3s/step - loss: 0.3079 - a
ccuracy: 0.8960 - val_loss: 0.7685 - val_accuracy: 0.7681
Epoch 4/20
156/156 [==============================] - 399s 3s/step - loss: 0.2476 - a
ccuracy: 0.9095 - val_loss: 0.4666 - val_accuracy: 0.8355
Epoch 5/20
156/156 [==============================] - 406s 3s/step - loss: 0.1866 - a
ccuracy: 0.9365 - val_loss: 0.9686 - val_accuracy: 0.7336
Epoch 6/20
156/156 [==============================] - 407s 3s/step - loss: 0.1647 - a
ccuracy: 0.9385 - val_loss: 0.5156 - val_accuracy: 0.8487
Epoch 7/20
156/156 [==============================] - 405s 3s/step - loss: 0.1466 - a
ccuracy: 0.9425 - val_loss: 0.4900 - val_accuracy: 0.8355
Epoch 8/20
156/156 [==============================] - 406s 3s/step - loss: 0.1161 - a
ccuracy: 0.9581 - val_loss: 0.4227 - val_accuracy: 0.8635
Epoch 9/20
156/156 [==============================] - 419s 3s/step - loss: 0.1142 - a
ccuracy: 0.9599 - val_loss: 0.1787 - val_accuracy: 0.9359
Epoch 10/20
156/156 [==============================] - 411s 3s/step - loss: 0.1278 - a
ccuracy: 0.9549 - val_loss: 0.3867 - val_accuracy: 0.8668
Epoch 11/20
156/156 [==============================] - 408s 3s/step - loss: 0.1203 - a
ccuracy: 0.9557 - val_loss: 0.3291 - val_accuracy: 0.8997
Epoch 12/20
156/156 [==============================] - 407s 3s/step - loss: 0.0800 - a
ccuracy: 0.9696 - val_loss: 0.5682 - val_accuracy: 0.8421
Epoch 13/20
156/156 [==============================] - 365s 2s/step - loss: 0.0694 - a
ccuracy: 0.9772 - val_loss: 0.4114 - val_accuracy: 0.8799
Epoch 14/20
156/156 [==============================] - 198s 1s/step - loss: 0.0642 - a
ccuracy: 0.9760 - val_loss: 0.3116 - val_accuracy: 0.8997
Epoch 15/20
156/156 [==============================] - 195s 1s/step - loss: 0.1020 - a
ccuracy: 0.9647 - val_loss: 0.2040 - val_accuracy: 0.9391
Epoch 16/20
156/156 [==============================] - 200s 1s/step - loss: 0.0738 - a
ccuracy: 0.9756 - val_loss: 0.3502 - val_accuracy: 0.9095
Epoch 17/20
156/156 [==============================] - 201s 1s/step - loss: 0.0618 - a
ccuracy: 0.9790 - val_loss: 0.2452 - val_accuracy: 0.9326
Epoch 18/20
156/156 [==============================] - 199s 1s/step - loss: 0.0690 - a
ccuracy: 0.9758 - val_loss: 0.1730 - val_accuracy: 0.9424
Epoch 19/20
156/156 [==============================] - 205s 1s/step - loss: 0.0589 - a
ccuracy: 0.9790 - val_loss: 0.1829 - val_accuracy: 0.9457
Epoch 20/20
156/156 [==============================] - 204s 1s/step - loss: 0.0426 - a
ccuracy: 0.9840 - val_loss: 0.1530 - val_accuracy: 0.9539
```

In [159]:

```
scores = model.evaluate(test_ds)
```

```
21/21 [==============================] - 18s 610ms/step - loss: 0.1638 - a
ccuracy: 0.9549
```

In [160]:

```
scores
```

Out[160]:

```
[0.1637960970401764, 0.9548988938331604]
```

In [161]:

```
history
```

Out[161]:

```
<keras.src.callbacks.History at 0x2a467ba38e0>
```

In [162]:

```
history.params
```

Out[162]:

```
{'verbose': 1, 'epochs': 20, 'steps': 156}
```

In [163]:

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

Out[163]:

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

In [164]:

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

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

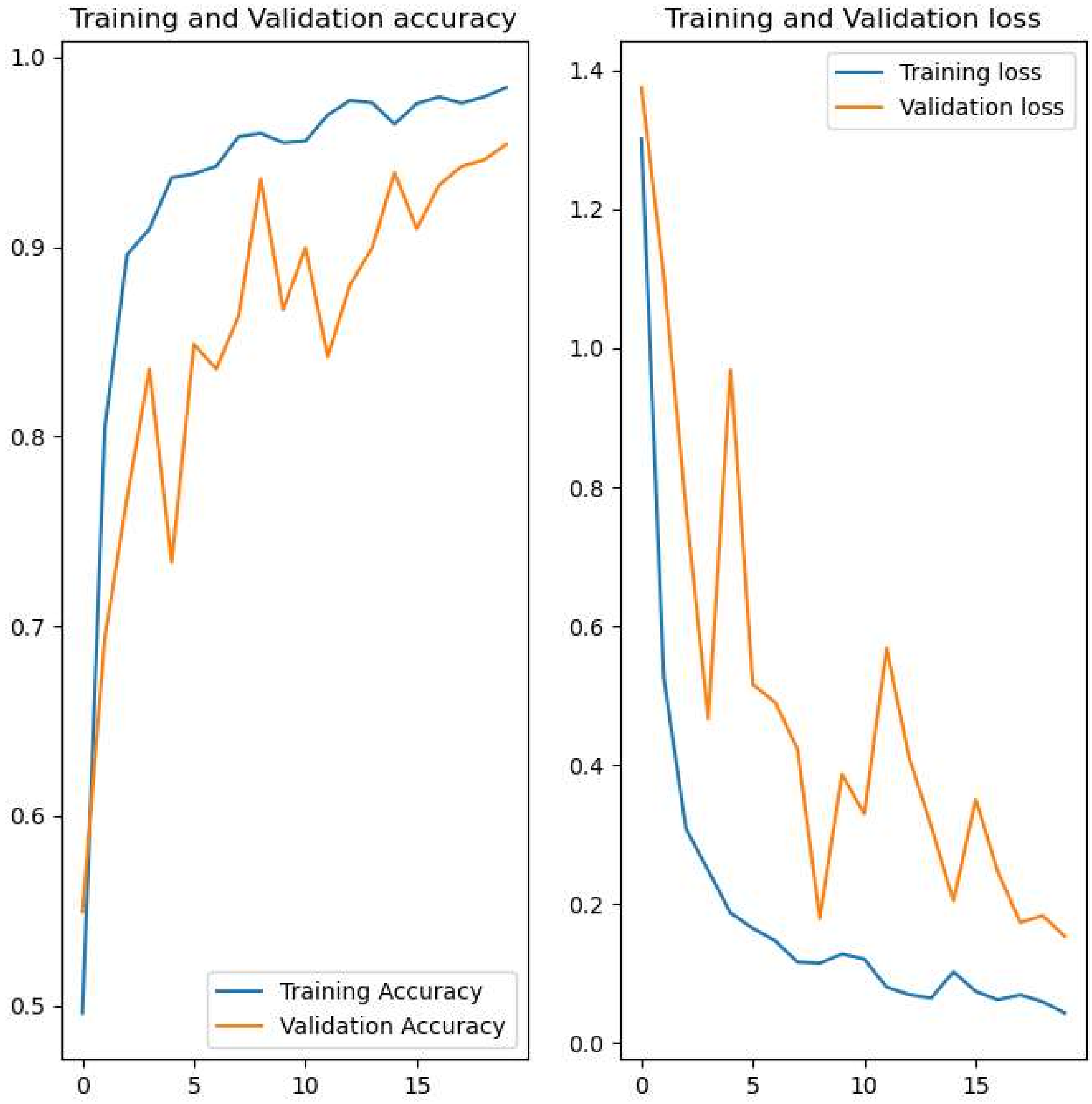
In [165]:

```
plt.figure(figsize=(8,8))
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')
```

Out[165]:

```
Text(0.5, 1.0, 'Training and Validation loss')
```



In [166]:

```
print("Predicted_label: ",batch_prediction[0])
```

```
Predicted_label:  [1.0000000e+00 1.9904942e-08 2.9574501e-08]
```

In [ ]:

```

```

In [171]:

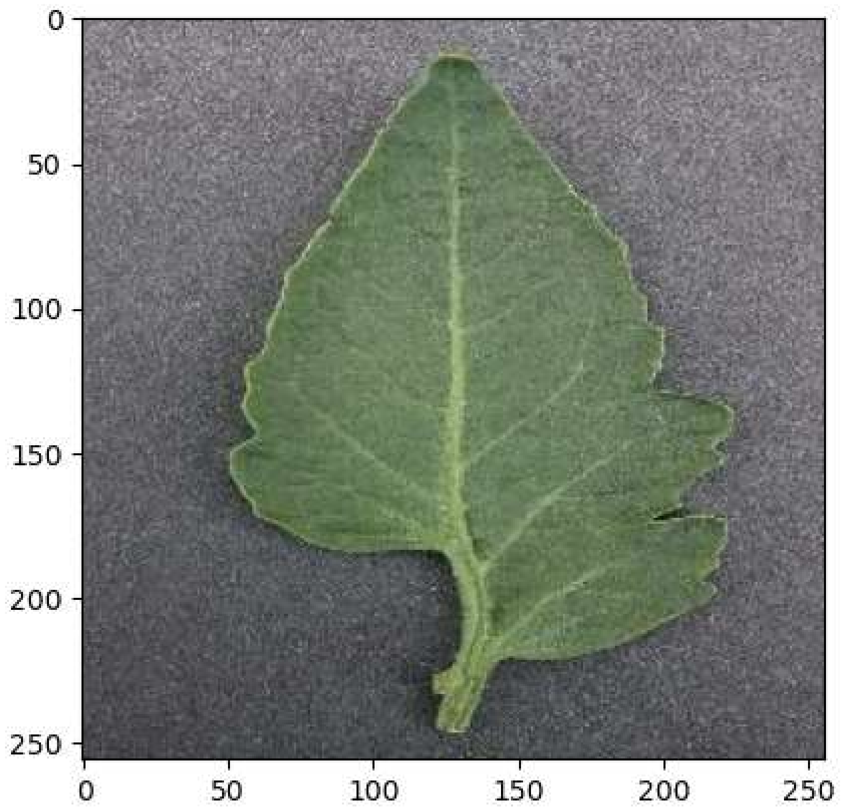
```
import numpy as np
for images_batch,labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("First image to predict")
    plt.imshow(first_image)
    print("Actual label: ",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("Predicted_label: ",class_names[np.argmax(batch_prediction[0])])
```

```
First image to predict
Actual label:  Tomato_healthy
1/1 [==============================] - 1s 693ms/step
Predicted_label:  Tomato_healthy
```



In [172]:

```
def predict(model,img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array,0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])),2)
    return predicted_class,confidence
```

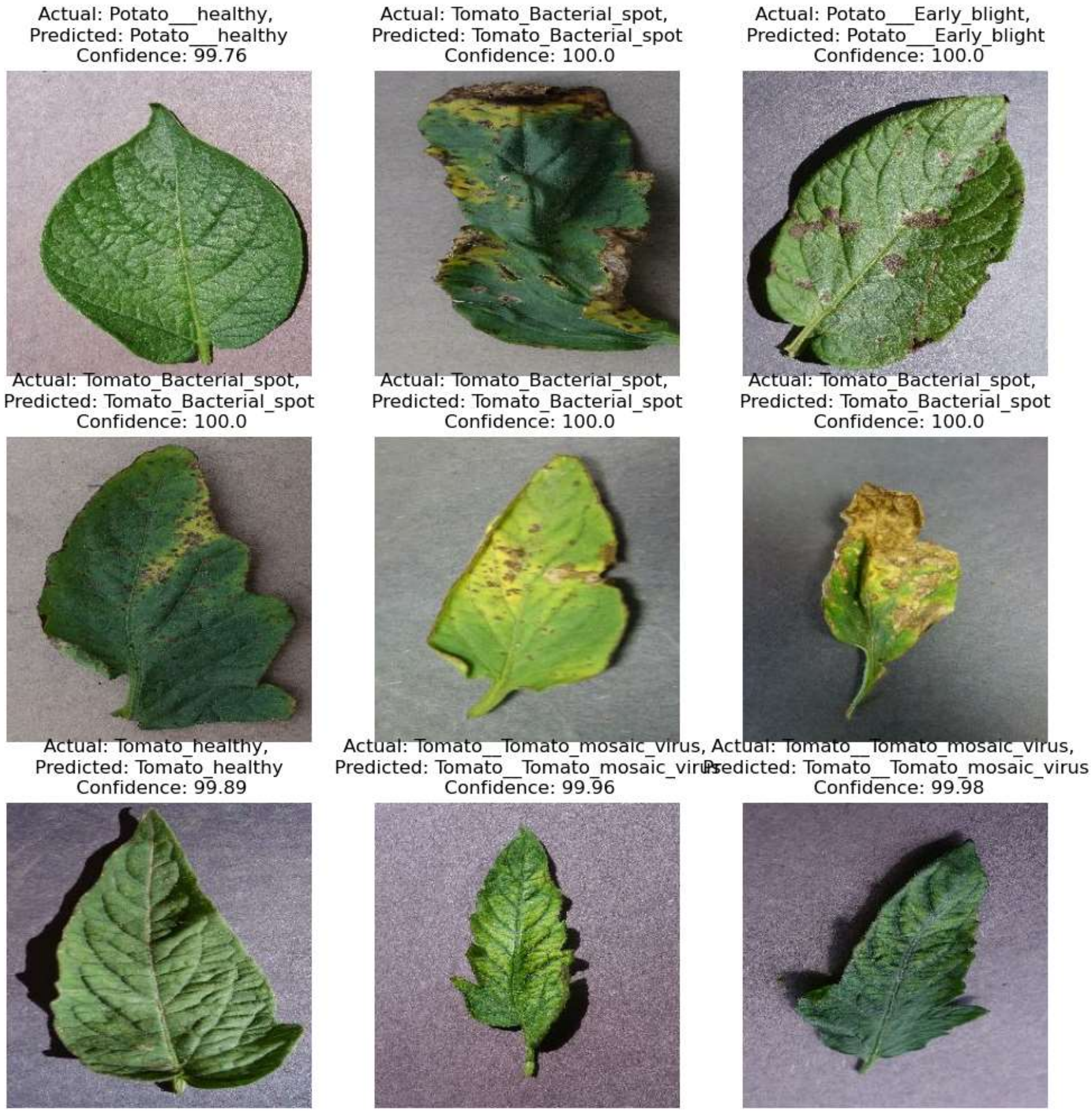
In [178]:

```
plt.figure(figsize=(12,12))
for images,labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype('uint8'))


        predicted_class,confidence = predict(model,images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class}, \n Predicted: {predicted_class} \n Confidenc
        plt.axis('off')
```

```
1/1 [==============================] - 0s 100ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 94ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 84ms/step
1/1 [==============================] - 0s 91ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 94ms/step
```



Actual: Potato___healthy,
Predicted: Potato___healthy
Confidence: 99.76
Actual: Tomato_Bacterial_spot,
Predicted: Tomato_Bacterial_spot
Confidence: 100.0
Actual: Potato___Early_blight,
Predicted: Potato___Early_blight
Confidence: 100.0
Actual: Tomato_Bacterial_spot,
Predicted: Tomato_Bacterial_spot
Confidence: 100.0
Actual: Tomato_Bacterial_spot,
Predicted: Tomato_Bacterial_spot
Confidence: 100.0
Actual: Tomato_Bacterial_spot,
Predicted: Tomato_Bacterial_spot
Confidence: 100.0
Actual: Tomato_healthy,
Predicted: Tomato_healthy
Confidence: 99.89
Actual: Tomato__Tomato_mosaic_virus,
Predicted: Tomato__Tomato_mosaic_virus
Confidence: 99.96
Actual: Tomato__Tomato_mosaic_virus,
Predicted: Tomato__Tomato_mosaic_virus
Confidence: 99.98

```
In [ ]:
```