In [1]:
```
import tensorflow as t
from tensorflow.keras import models,layers
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.compat.v1.losses import sparse_softmax_cross_entropy
from tensorflow.compat.v1.ragged import RaggedTensorValue
```

```
WARNING:tensorflow:From C:\Users\Guruji\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [2]:
```
IMAGE_SIZE=256
BATCH_SIZE=32
CHANNELS=3
EPOCHS=50 # its a trail and error
```

In [3]:
```
dataset=t.keras.preprocessing.image_dataset_from_directory(
    "plantvillage",
    shuffle=True,
    image_size=( IMAGE_SIZE,IMAGE_SIZE),
     batch_size=BATCH_SIZE
)
```

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

In [4]:
```
class_names = dataset.class_names
class_names
```

Out[4]:
```
['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']
```

In [5]:
```
len(dataset) #each batch contain 32 images 32*68 =2176 approx(near to the 2152)
```

Out[5]:
```
68
```

In [6]:
```
for image_batch,label_batch in dataset.take(1):
    print(image_batch.shape)
    print(label_batch.numpy())
    #32 images in 1 batch
    #256 x and y
    # 3 is RGB
```

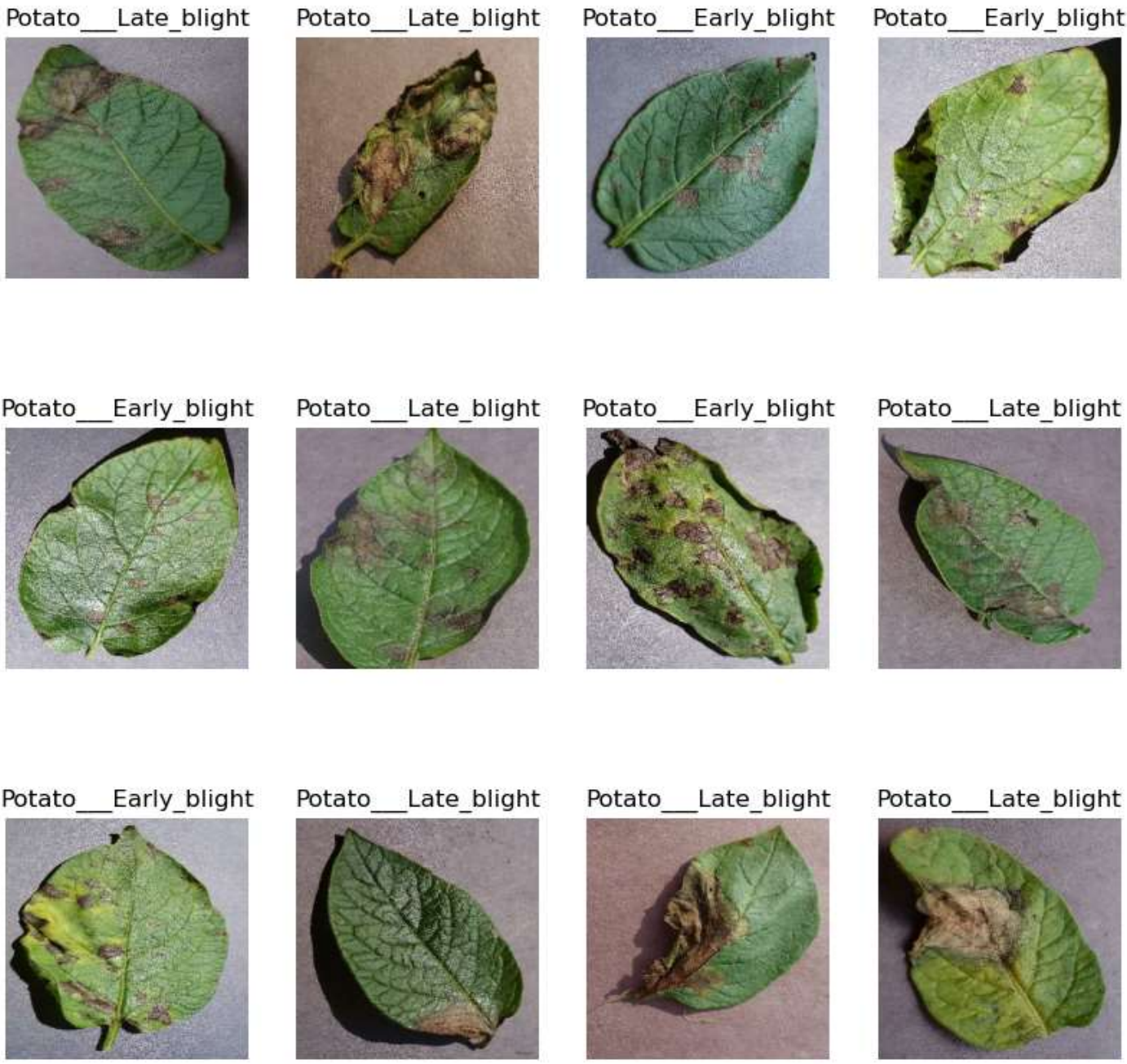
```
(32, 256, 256, 3)
[0 0 0 1 0 0 1 2 0 0 1 0 2 0 1 1 1 1 0 0 1 1 0 1 0 0 0 1 2 1 0 0]
```

In [7]:
```
for image_batch,label_batch in dataset.take(1):
    print(image_batch[0].shape)
```

```
(256, 256, 3)
```

In [20]:
```
plt.figure(figsize=(10,10))
for image_batch,label_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[label_batch[i]])
```

```
        plt.axis("off")
    # for every run of the current cell image will shuffle
```



80% ==> training 20% ==> 10% validation,10% test

In [8]:
```
train_size=0.8
len(dataset)*train_size
```

Out[8]: 54.400000000000006

In [9]:
```
train_ds=dataset.take(54)
len(train_ds)
```

Out[9]: 54

In [10]:
```
test_ds=dataset.skip(54)
len(test_ds)
```

Out[10]: 14

In [11]:
```
val_size=0.1
len(dataset)*val_size
```

Out[11]: 6.800000000000001

In [12]:
```
val_ds=test_ds.take(6)
len(val_ds)
```

Out[12]: 6

In [13]:
```
test_ds=test_ds.skip(6)
len(test_ds)
```

Out[13]: 8

In [14]:
```
def get_dataset_partitions_tf(ds,train_split=0.8,val_split=0.1,test_split=0.1,shuffle=True,shuff

    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
```

In [15]:
```
train_ds,val_ds,test_ds=get_dataset_partitions_tf(dataset)
```

In [16]:
```
len(train_ds)
```

Out[16]: 54

In [17]:
```
len(val_ds)
```

Out[17]: 6

In [18]:
```
len(test_ds)
```

Out[18]: 8

In [19]:
```
train_ds=train_ds.cache().shuffle(1000).prefetch(buffer_size=t.data.AUTOTUNE)
val_ds=val_ds.cache().shuffle(1000).prefetch(buffer_size=t.data.AUTOTUNE)
test_ds=test_ds.cache().shuffle(1000).prefetch(buffer_size=t.data.AUTOTUNE)
```

In [20]:
```
resize_and_rescale=t.keras.Sequential([
    layers.experimental.preprocessing.Rescaling(IMAGE_SIZE,IMAGE_SIZE),
    layers.experimental.preprocessing.Rescaling(1.0/255)
])
```

```
WARNING:tensorflow:From C:\Users\Guruji\anaconda3\Lib\site-packages\keras\src\backend.py:873: The
name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

In [21]:
```
data_augmentation=t.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2)
])
```

In [22]:
```
input_shape=(BATCH_SIZE,IMAGE_SIZE,IMAGE_SIZE,CHANNELS)
n_classes =3

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'),
    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)
```

```
WARNING:tensorflow:From C:\Users\Guruji\anaconda3\Lib\site-packages\keras\src\layers\pooling\max_
pooling2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [23]:
```
model.summary()
```

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

 sequential_1 (Sequential)   (32, 256, 256, 3)         0

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

 max_pooling2d (MaxPooling2  (32, 127, 127, 32)        0
 D)

 conv2d_1 (Conv2D)           (32, 125, 125, 64)        18496

 max_pooling2d_1 (MaxPoolin  (32, 62, 62, 64)          0
 g2D)

 conv2d_2 (Conv2D)           (32, 60, 60, 64)          36928

 max_pooling2d_2 (MaxPoolin  (32, 30, 30, 64)          0
 g2D)

 conv2d_3 (Conv2D)           (32, 28, 28, 64)          36928

 max_pooling2d_3 (MaxPoolin  (32, 14, 14, 64)          0
 g2D)

 conv2d_4 (Conv2D)           (32, 12, 12, 64)          36928

 max_pooling2d_4 (MaxPoolin  (32, 6, 6, 64)            0
 g2D)

 conv2d_5 (Conv2D)           (32, 4, 4, 64)            36928

 max_pooling2d_5 (MaxPoolin  (32, 2, 2, 64)            0
 g2D)

 flatten (Flatten)           (32, 256)                 0

 dense (Dense)               (32, 64)                  16448

 dense_1 (Dense)             (32, 3)                   195

=================================================================
Total params: 183747 (717.76 KB)
Trainable params: 183747 (717.76 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [24]:

```
model.compile(
    optimizer='adam',
    loss=t.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

```
WARNING:tensorflow:From C:\Users\Guruji\anaconda3\Lib\site-packages\keras\src\optimizers\__init_
_.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer inst
ead.
```

In [25]:
```
history=model.fit(
    train_ds,
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    verbose=1,
    validation_data=val_ds
)
```

```
Epoch 1/50
WARNING:tensorflow:From C:\Users\Guruji\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:4
92: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTens
orValue instead.

WARNING:tensorflow:From C:\Users\Guruji\anaconda3\Lib\site-packages\keras\src\engine\base_layer_u
tils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.
v1.executing_eagerly_outside_functions instead.

54/54 [==============================] - 117s 2s/step - loss: 1.5843 - accuracy: 0.5139 - val_los
s: 0.8782 - val_accuracy: 0.5417
Epoch 2/50
54/54 [==============================] - 103s 2s/step - loss: 0.7968 - accuracy: 0.6094 - val_los
s: 0.7340 - val_accuracy: 0.6823
Epoch 3/50
54/54 [==============================] - 113s 2s/step - loss: 0.4701 - accuracy: 0.8009 - val_los
s: 0.5916 - val_accuracy: 0.7083
Epoch 4/50
54/54 [==============================] - 98s 2s/step - loss: 0.3226 - accuracy: 0.8709 - val_los
s: 0.3792 - val_accuracy: 0.8229
Epoch 5/50
54/54 [==============================] - 102s 2s/step - loss: 0.3233 - accuracy: 0.8611 - val_los
s: 0.4294 - val_accuracy: 0.8385
Epoch 6/50
54/54 [==============================] - 249s 5s/step - loss: 0.2525 - accuracy: 0.9010 - val_los
s: 0.3943 - val_accuracy: 0.8646
Epoch 7/50
54/54 [==============================] - 145s 3s/step - loss: 0.2444 - accuracy: 0.8953 - val_los
s: 0.4073 - val_accuracy: 0.8854
Epoch 8/50
54/54 [==============================] - 172s 3s/step - loss: 0.2149 - accuracy: 0.9201 - val_los
s: 0.2438 - val_accuracy: 0.8958
Epoch 9/50
54/54 [==============================] - 199s 4s/step - loss: 0.1840 - accuracy: 0.9236 - val_los
s: 0.3685 - val_accuracy: 0.8333
Epoch 10/50
54/54 [==============================] - 97s 2s/step - loss: 0.1822 - accuracy: 0.9294 - val_los
s: 0.2496 - val_accuracy: 0.8906
Epoch 11/50
54/54 [==============================] - 97s 2s/step - loss: 0.1457 - accuracy: 0.9427 - val_los
s: 0.2283 - val_accuracy: 0.9115
Epoch 12/50
54/54 [==============================] - 95s 2s/step - loss: 0.1931 - accuracy: 0.9207 - val_los
s: 0.1198 - val_accuracy: 0.9479
Epoch 13/50
54/54 [==============================] - 94s 2s/step - loss: 0.1423 - accuracy: 0.9450 - val_los
s: 0.1491 - val_accuracy: 0.9479
Epoch 14/50
54/54 [==============================] - 95s 2s/step - loss: 0.1233 - accuracy: 0.9543 - val_los
s: 0.1325 - val_accuracy: 0.9323
Epoch 15/50
54/54 [==============================] - 94s 2s/step - loss: 0.1207 - accuracy: 0.9473 - val_los
s: 0.1976 - val_accuracy: 0.9167
Epoch 16/50
54/54 [==============================] - 95s 2s/step - loss: 0.1283 - accuracy: 0.9508 - val_los
s: 0.1143 - val_accuracy: 0.9479
Epoch 17/50
54/54 [==============================] - 95s 2s/step - loss: 0.1018 - accuracy: 0.9572 - val_los
s: 0.1967 - val_accuracy: 0.9427
```

```
Epoch 18/50
54/54 [==============================] - 92s 2s/step - loss: 0.0919 - accuracy: 0.9641 - val_los
s: 0.1242 - val_accuracy: 0.9531
Epoch 19/50
54/54 [==============================] - 93s 2s/step - loss: 0.1308 - accuracy: 0.9473 - val_los
s: 0.1763 - val_accuracy: 0.9115
Epoch 20/50
54/54 [==============================] - 96s 2s/step - loss: 0.0855 - accuracy: 0.9647 - val_los
s: 0.1035 - val_accuracy: 0.9635
Epoch 21/50
54/54 [==============================] - 92s 2s/step - loss: 0.1386 - accuracy: 0.9479 - val_los
s: 0.0965 - val_accuracy: 0.9635
Epoch 22/50
54/54 [==============================] - 94s 2s/step - loss: 0.1156 - accuracy: 0.9537 - val_los
s: 0.0575 - val_accuracy: 0.9688
Epoch 23/50
54/54 [==============================] - 94s 2s/step - loss: 0.0877 - accuracy: 0.9659 - val_los
s: 0.0662 - val_accuracy: 0.9635
Epoch 24/50
54/54 [==============================] - 93s 2s/step - loss: 0.1061 - accuracy: 0.9653 - val_los
s: 0.1711 - val_accuracy: 0.9427
Epoch 25/50
54/54 [==============================] - 93s 2s/step - loss: 0.0648 - accuracy: 0.9769 - val_los
s: 0.0691 - val_accuracy: 0.9688
Epoch 26/50
54/54 [==============================] - 95s 2s/step - loss: 0.0558 - accuracy: 0.9786 - val_los
s: 0.0463 - val_accuracy: 0.9792
Epoch 27/50
54/54 [==============================] - 88s 2s/step - loss: 0.0612 - accuracy: 0.9786 - val_los
s: 0.1163 - val_accuracy: 0.9688
Epoch 28/50
54/54 [==============================] - 86s 2s/step - loss: 0.0560 - accuracy: 0.9803 - val_los
s: 0.0492 - val_accuracy: 0.9844
Epoch 29/50
54/54 [==============================] - 87s 2s/step - loss: 0.0577 - accuracy: 0.9815 - val_los
s: 0.0931 - val_accuracy: 0.9635
Epoch 30/50
54/54 [==============================] - 89s 2s/step - loss: 0.0494 - accuracy: 0.9844 - val_los
s: 0.0495 - val_accuracy: 0.9792
Epoch 31/50
54/54 [==============================] - 89s 2s/step - loss: 0.0641 - accuracy: 0.9786 - val_los
s: 0.1915 - val_accuracy: 0.9167
Epoch 32/50
54/54 [==============================] - 88s 2s/step - loss: 0.0498 - accuracy: 0.9832 - val_los
s: 0.0652 - val_accuracy: 0.9844
Epoch 33/50
54/54 [==============================] - 87s 2s/step - loss: 0.1077 - accuracy: 0.9578 - val_los
s: 0.0879 - val_accuracy: 0.9740
Epoch 34/50
54/54 [==============================] - 3739s 71s/step - loss: 0.0707 - accuracy: 0.9728 - val_l
oss: 0.0156 - val_accuracy: 0.9896
Epoch 35/50
54/54 [==============================] - 108s 2s/step - loss: 0.0717 - accuracy: 0.9728 - val_los
s: 0.0609 - val_accuracy: 0.9740
Epoch 36/50
54/54 [==============================] - 154s 3s/step - loss: 0.0664 - accuracy: 0.9769 - val_los
s: 0.0409 - val_accuracy: 0.9844
Epoch 37/50
54/54 [==============================] - 128s 2s/step - loss: 0.0650 - accuracy: 0.9797 - val_los
```

```
s: 0.1340 - val_accuracy: 0.9375
Epoch 38/50
54/54 [==============================] - 126s 2s/step - loss: 0.0588 - accuracy: 0.9774 - val_los
s: 0.0290 - val_accuracy: 0.9896
Epoch 39/50
54/54 [==============================] - 133s 2s/step - loss: 0.0415 - accuracy: 0.9850 - val_los
s: 0.0300 - val_accuracy: 0.9792
Epoch 40/50
54/54 [==============================] - 136s 3s/step - loss: 0.0330 - accuracy: 0.9855 - val_los
s: 0.0342 - val_accuracy: 0.9792
Epoch 41/50
54/54 [==============================] - 136s 3s/step - loss: 0.0403 - accuracy: 0.9855 - val_los
s: 0.2493 - val_accuracy: 0.9219
Epoch 42/50
54/54 [==============================] - 129s 2s/step - loss: 0.0656 - accuracy: 0.9803 - val_los
s: 0.0566 - val_accuracy: 0.9688
Epoch 43/50
54/54 [==============================] - 129s 2s/step - loss: 0.0752 - accuracy: 0.9699 - val_los
s: 0.0315 - val_accuracy: 0.9844
Epoch 44/50
54/54 [==============================] - 128s 2s/step - loss: 0.0570 - accuracy: 0.9769 - val_los
s: 0.0468 - val_accuracy: 0.9844
Epoch 45/50
54/54 [==============================] - 129s 2s/step - loss: 0.0455 - accuracy: 0.9867 - val_los
s: 0.0065 - val_accuracy: 1.0000
Epoch 46/50
54/54 [==============================] - 127s 2s/step - loss: 0.0811 - accuracy: 0.9740 - val_los
s: 0.0430 - val_accuracy: 0.9844
Epoch 47/50
54/54 [==============================] - 127s 2s/step - loss: 0.0449 - accuracy: 0.9832 - val_los
s: 0.0370 - val_accuracy: 0.9844
Epoch 48/50
54/54 [==============================] - 130s 2s/step - loss: 0.0435 - accuracy: 0.9850 - val_los
s: 0.0200 - val_accuracy: 0.9896
Epoch 49/50
54/54 [==============================] - 128s 2s/step - loss: 0.0496 - accuracy: 0.9815 - val_los
s: 0.0610 - val_accuracy: 0.9844
Epoch 50/50
54/54 [==============================] - 128s 2s/step - loss: 0.0344 - accuracy: 0.9873 - val_los
s: 0.0320 - val_accuracy: 0.9948
```

In [26]:
```
scores=model.evaluate(test_ds)
```

```
8/8 [==============================] - 19s 520ms/step - loss: 0.0364 - accuracy: 0.9883
```

In [27]:
```
scores
```

Out[27]:
```
[0.03642941266298294, 0.98828125]
```

In [28]:
```
history
```

Out[28]:
```
<keras.src.callbacks.History at 0x1562b207c90>
```

In [29]:
```
history.params
```

Out[29]:
```
{'verbose': 1, 'epochs': 50, 'steps': 54}
```

In [30]:
```
history.history.keys()
```

Out[30]: dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

In [31]:
```
history.history['accuracy']
```

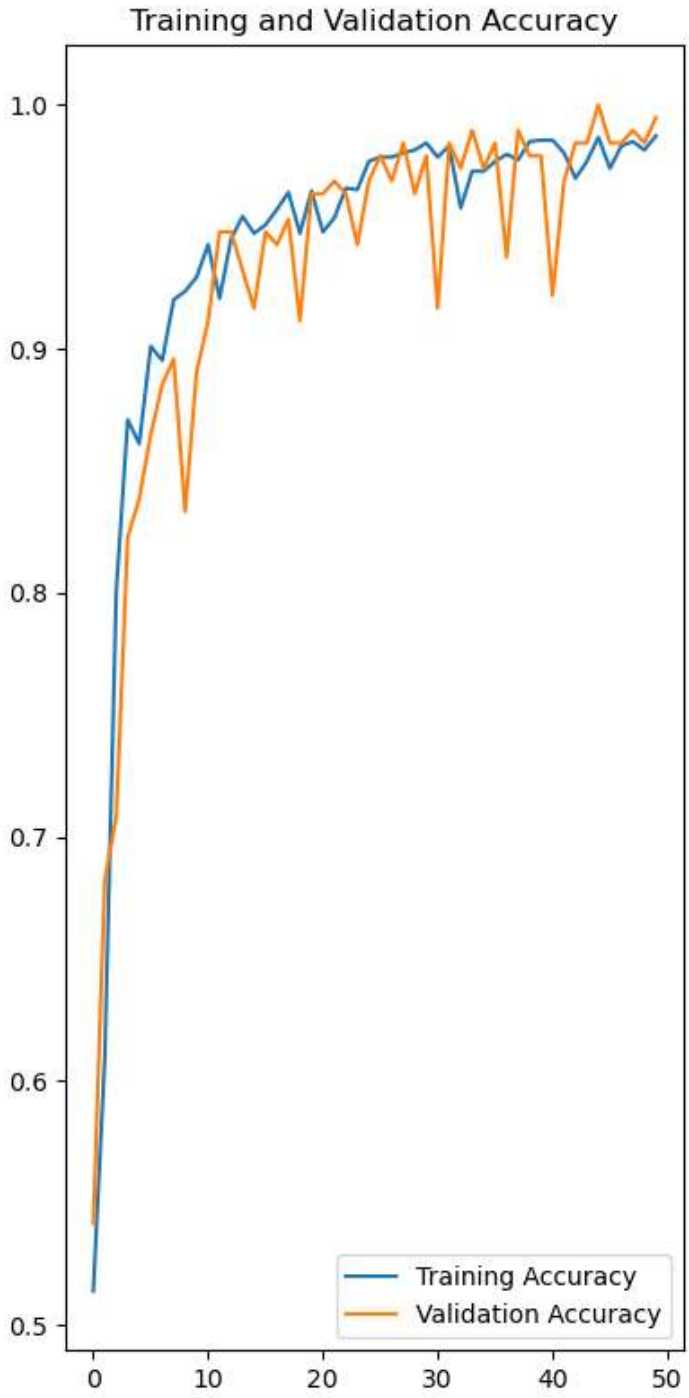
Out[31]:
```
[0.5138888955116272,
 0.609375,
 0.8009259104728699,
 0.8709490895271301,
 0.8611111044883728,
 0.9010416865348816,
 0.8952546119689941,
 0.9201388955116272,
 0.9236111044883728,
 0.9293981194496155,
 0.9427083134651184,
 0.9207175970077515,
 0.9450231194496155,
 0.9542824029922485,
 0.9473379850387573,
 0.9508101940155029,
 0.9571759104728699,
 0.9641203880310059,
 0.9473379850387573,
 0.9646990895271301,
 0.9479166865348816,
 0.9537037014961243,
 0.9658564925193787,
 0.9652777910232544,
 0.9768518805503845,
 0.9785879850387573,
 0.9785879850387573,
 0.9803240895271301,
 0.9814814925193787,
 0.984375,
 0.9785879850387573,
 0.9832175970077515,
 0.9577546119689941,
 0.9728009104728699,
 0.9728009104728699,
 0.9768518805503845,
 0.9797453880310059,
 0.9774305820465088,
 0.9849537014961243,
 0.9855324029922485,
 0.9855324029922485,
 0.9803240895271301,
 0.9699074029922485,
 0.9768518805503845,
 0.9866898059844971,
 0.9739583134651184,
 0.9832175970077515,
 0.9849537014961243,
 0.9814814925193787,
 0.9872685074806213]
```

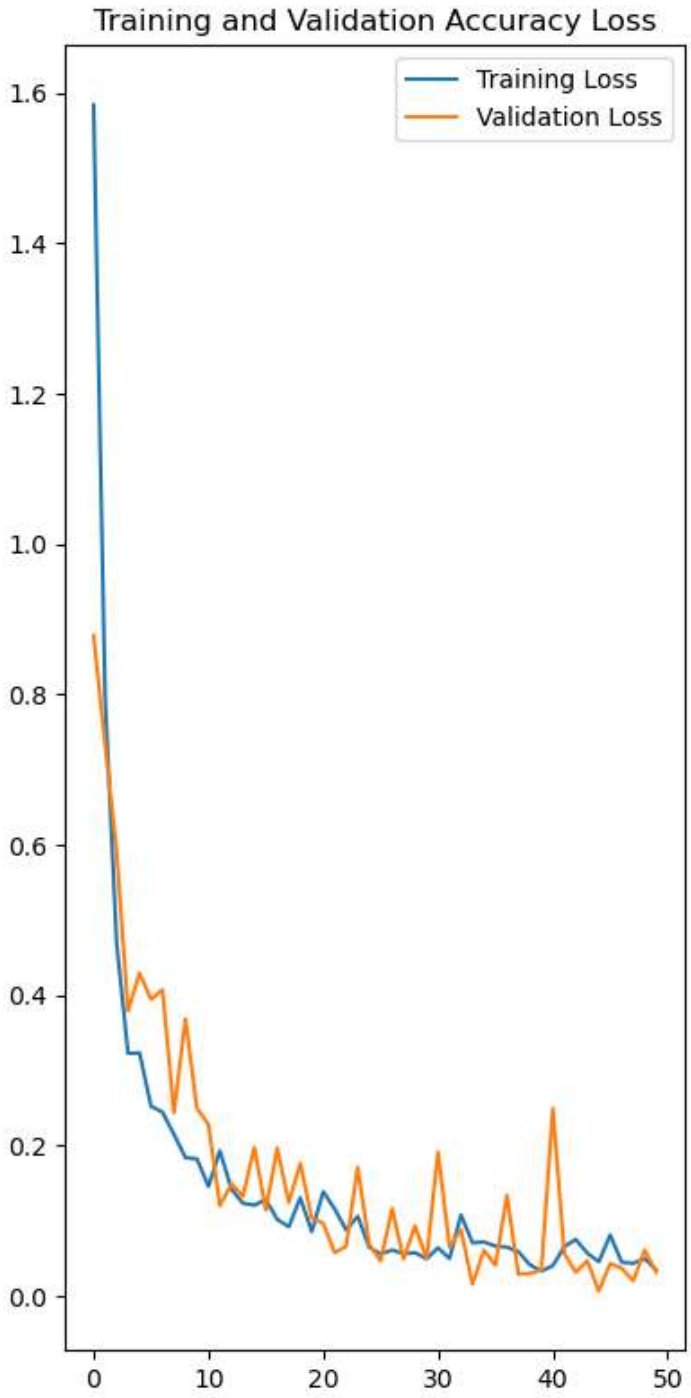
In [32]:
```
acc=history.history['accuracy']
val_acc=history.history['val_accuracy']
```

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

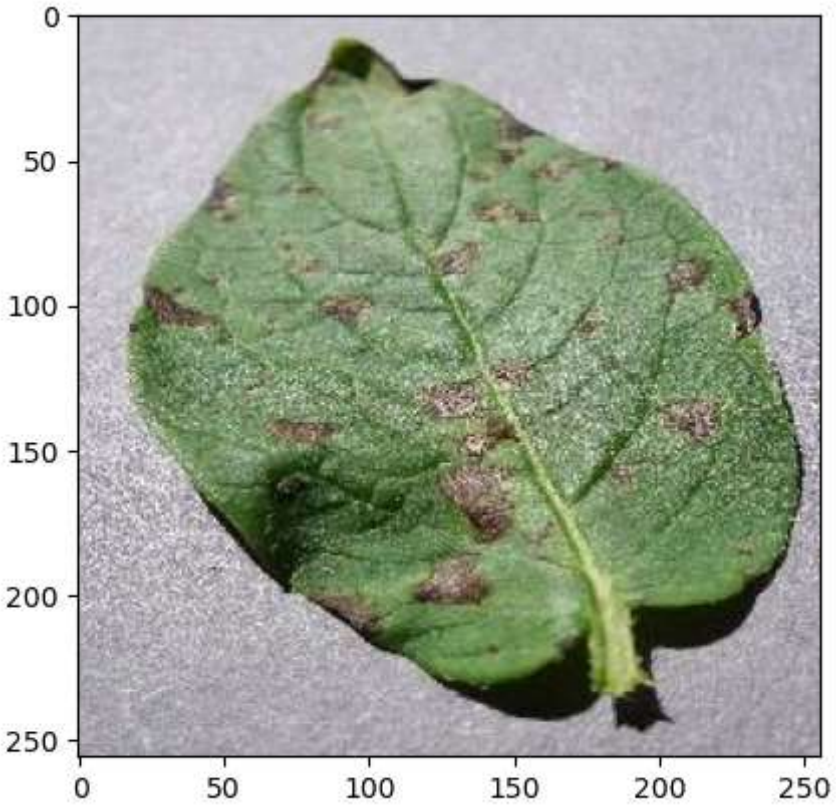
In [33]:

```
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 Accuracy Loss')
plt.tight_layout()
plt.show()
```



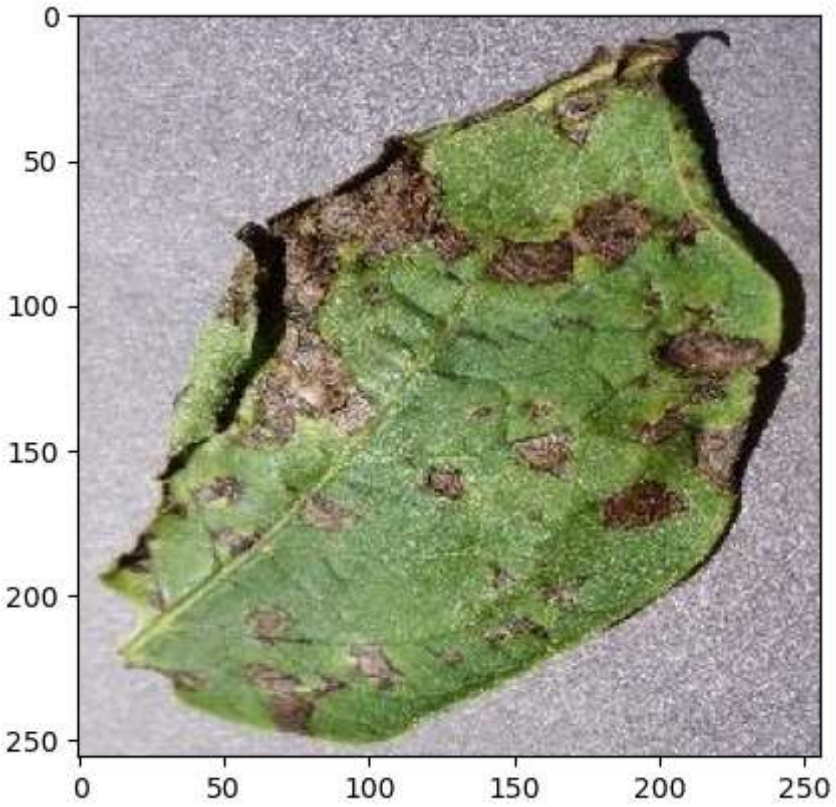


In [34]:
```
for images_batch,labels_batch in test_ds.take(1):

    plt.imshow(images_batch[0].numpy().astype('uint8'))
```



In [35]:
```
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(batch_prediction[0])
```

```
first image to predict
actual label: Potato___Early_blight
1/1 [==============================] - 1s 775ms/step
[1.0000000e+00 7.1454326e-10 7.6566772e-26]
```

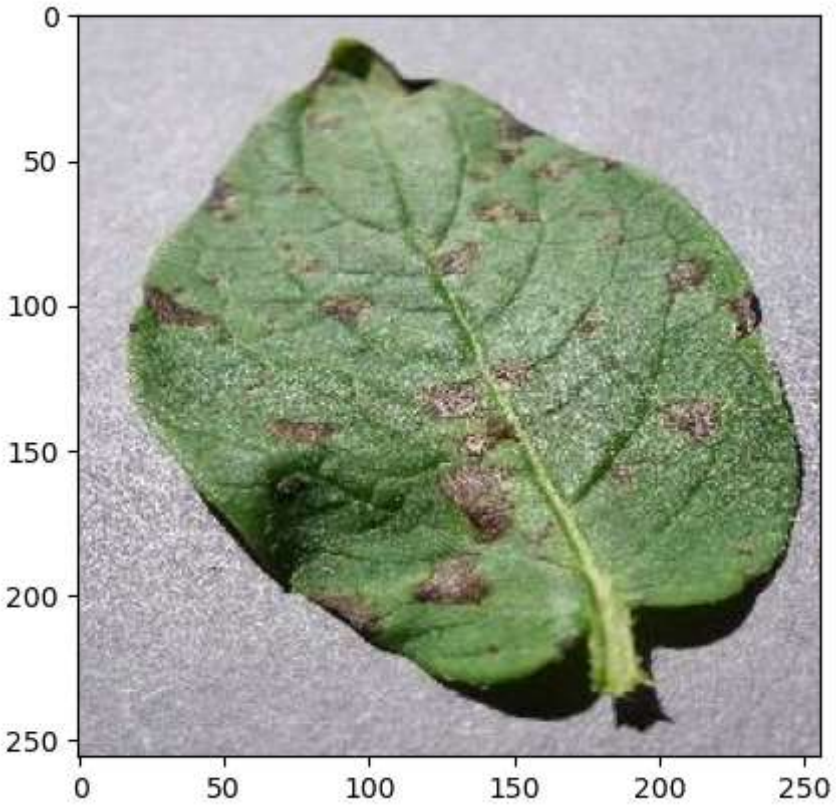


In [ ]:

In [36]:
```
#np.argmax([8.966722e-04, 9.990941e-01,9.219155e-06])
```

In [37]:
```
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: Potato___Early_blight
1/1 [==============================] - 0s 346ms/step
predicted label: Potato___Early_blight
```



In [38]:
```
def predict(model, img):
     img_array = t.keras.preprocessing.image.img_to_array(images[i].numpy())
     img_array = t.expand_dims(img_array, 0) # Create a batch

     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 [39]:
```
plt.figure(figsize=(15,15))
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 Confidence: {confid
        plt.axis("off")
```

```
1/1 [==============================] - 0s 270ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 37ms/step
```

Actual:Potato___Late_blight,
predicted: Potato___Late_blight.
Confidence: 100.0%

Actual:Potato___Early_blight,
predicted: Potato___Early_blight.
Confidence: 100.0%

Actual:Potato___healthy,
predicted: Potato___healthy.
Confidence: 100.0%

Actual:Potato___Late_blight,
predicted: Potato___Late_blight.
Confidence: 100.0%

Actual:Potato___Late_blight,
predicted: Potato___Late_blight.
Confidence: 99.45%

Actual:Potato___healthy,
predicted: Potato___healthy.
Confidence: 99.94%

Actual:Potato___Late_blight,
predicted: Potato___Late_blight.
Confidence: 99.98%

Actual:Potato___Late_blight,
predicted: Potato___healthy.
Confidence: 96.3%

Actual:Potato___Late_blight,
predicted: Potato___Late_blight.
Confidence: 99.59%

In [ ]: