Import Dependencies

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
import torch
from torchvision import datasets, transforms # datsets  , transforms
from torch.utils.data.sampler import SubsetRandomSampler
import torch.nn as nn
import torch.nn.functional as F
from datetime import datetime
```

In [3]:

```
%load_ext nb_black
```

Import Dataset

Dataset Link (Plant Vliiage Dataset):
https://data.mendeley.com/datasets/tywbtsjrjv/1 (https://data.mendeley.com/datasets/tywbtsjrjv/1)

In [4]:

```
transform = transforms.Compose(
    [transforms.Resize(255), transforms.CenterCrop(224), transforms.ToTensor()]
)
```

In [5]:

```
dataset = datasets.ImageFolder("Dataset", transform=transform)
```

In [6]:

```
dataset
```

Out[6]:

```
Dataset ImageFolder
    Number of datapoints: 61486
    Root Location: Dataset
    Transforms (if any): Compose(
                             Resize(size=255, interpolation=PIL.Image.BILINE
AR)
                             CenterCrop(size=(224, 224))
                             ToTensor()
                         )
    Target Transforms (if any): None
```

In [7]:

```
indices = list(range(len(dataset)))
```

In [8]:

```
split = int(np.floor(0.8 * len(dataset)))  # train_size
```

In [9]:

```
np.random.shuffle(indices)
```

Split into Train and Test

In [10]:

```
train_indices, test_indices = indices[:split], indices[split:]
```

In [11]:

```
train_sampler = SubsetRandomSampler(train_indices)
test_sampler = SubsetRandomSampler(test_indices)
```

In [12]:

```
batch_size = 64
train_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=train_sampler
)
test_loader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, sampler=test_sampler
)
```

In [13]:

```
targets_size = len(dataset.class_to_idx)
```

Model

Convolution Aithmetic Equation : (W - F + 2P) / S + 1
W = Input Size
F = Filter Size
P = Padding Size
S = Stride

Shape on Each Layer

Image		(3 , 224 , 224)
Conv 1	32	(32 , 222 , 222)
Max Pool 1	2	(32 , 111 , 111)
Conv2	64	(64 , 109 , 109)
MaxPool2	2	(64 , 54 , 54)
Conv3	128	(128 , 52 , 52)
MaxPool3	2	(128 , 26 , 26)
Conv4	256	(256 , 24 , 24)
MaxPool4	2	(256 , 12 , 12)
F C	1024	(256 *12* 12 , 1024)
FC	39	(1024 , 39)

In [14]:

```
class CNN(nn.Module):
    def __init__(self, K):
        super(CNN, self).__init__()
        self.conv_layer1 = nn.Conv2d(
            in_channels=3, out_channels=32, kernel_size=3, stride=1
        )

        self.max_pool1 = nn.MaxPool2d(2, 2)

        self.conv_layer2 = nn.Conv2d(
            in_channels=32, out_channels=64, kernel_size=3, stride=1
        )

        self.max_pool2 = nn.MaxPool2d(2, 2)

        self.conv_layer3 = nn.Conv2d(
            in_channels=64, out_channels=128, kernel_size=3, stride=1
        )

        self.max_pool3 = nn.MaxPool2d(2, 2)

        self.conv_layer4 = nn.Conv2d(
            in_channels=128, out_channels=256, kernel_size=3, stride=1
        )

        self.max_pool4 = nn.MaxPool2d(2, 2)

        self.fc1 = nn.Linear(12 * 12 * 256, 1024)
        self.fc2 = nn.Linear(1024, K)

    def forward(self, X):
        x = F.relu(self.conv_layer1(X))
        x = self.max_pool1(x)
        x = F.relu(self.conv_layer2(x))
        x = self.max_pool2(x)
        x = F.relu(self.conv_layer3(x))
        x = self.max_pool3(x)
        x = F.relu(self.conv_layer4(x))
        x = self.max_pool4(x)
        # Flatten
        x = x.view(-1, 256 * 12 * 12)

        # Fully connected
        x = F.dropout(x, p=0.5)
        x = self.fc1(x)
        x = F.dropout(x, p=0.5)
        out = self.fc2(x)

        return out
```

In [16]:

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cpu
```

In [18]:

```
model = CNN(targets_size)
```

In [19]:

```
model.to(device)
```

Out[19]:

```
CNN(
  (conv_layer1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
  (max_pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (conv_layer2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (max_pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (conv_layer3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (max_pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (conv_layer4): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1))
  (max_pool4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (fc1): Linear(in_features=36864, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=39, bias=True)
)
```

In [20]:

```
from torchsummary import summary

summary(model, (3, 224, 224))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 222, 222]             896
         MaxPool2d-2         [-1, 32, 111, 111]               0
            Conv2d-3         [-1, 64, 109, 109]          18,496
         MaxPool2d-4           [-1, 64, 54, 54]               0
            Conv2d-5          [-1, 128, 52, 52]          73,856
         MaxPool2d-6          [-1, 128, 26, 26]               0
            Conv2d-7          [-1, 256, 24, 24]         295,168
         MaxPool2d-8          [-1, 256, 12, 12]               0
            Linear-9                [-1, 1024]      37,749,760
           Linear-10                  [-1, 39]          39,975
================================================================
Total params: 38,178,151
Trainable params: 38,178,151
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 26.98
Params size (MB): 145.64
Estimated Total Size (MB): 173.19
----------------------------------------------------------------
```

In [21]:

```
criterion = nn.CrossEntropyLoss()  # this include softmax + cross entropy loss
optimizer = torch.optim.Adam(model.parameters())
```

Batch Gradient Descent

In [22]:

```
def batch_gd(model, criterion, train_loader, test_laoder, epochs):
    train_losses = np.zeros(epochs)
    test_losses = np.zeros(epochs)

    for e in range(epochs):
        t0 = datetime.now()
        train_loss = []
        for inputs, targets in train_loader:
            inputs, targets = inputs.to(device), targets.to(device)

            optimizer.zero_grad()

            output = model(inputs)

            loss = criterion(output, targets)

            train_loss.append(loss.item())  # torch to numpy world

            loss.backward()
            optimizer.step()

        train_loss = np.mean(train_loss)

        test_loss = []

        for inputs, targets in test_loader:

            inputs, targets = inputs.to(device), targets.to(device)

            output = model(inputs)

            loss = criterion(output, targets)

            test_loss.append(loss.item())  # torch to numpy world

        test_loss = np.mean(test_loss)

        train_losses[e] = train_loss
        test_losses[e] = test_loss

        dt = datetime.now() - t0

        print(
            f"Epoch : {e+1}/{epochs} Train_loss:{train_loss:.3f} Test_loss:{test_loss:.3f}
        )

    return train_losses, test_losses
```

In [23]:

```
# train_losses, test_losses = batch_gd(model, criterion, train_loader, test_loader, 5)
```

Save the Model

In [24]:

```
torch.save(model.state_dict() , 'new_Model.pt')
```

Load Model

In [25]:

```
model = CNN(targets_size)
model.load_state_dict(torch.load("new_Model.pt"))
model.eval()
```

Out[25]:

```
CNN(
  (conv_layer1): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1))
  (max_pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (conv_layer2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
  (max_pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (conv_layer3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
  (max_pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (conv_layer4): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1))
  (max_pool4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, cei
l_mode=False)
  (fc1): Linear(in_features=36864, out_features=1024, bias=True)
  (fc2): Linear(in_features=1024, out_features=39, bias=True)
)
```

In [26]:

```
# %matplotlib notebook
```

Plot the loss

In [27]:

```
# plt.plot(train_losses , label = 'train_loss')
# plt.plot(test_losses , label = 'test_loss')
# plt.legend()
# plt.show()
```

Accuracy

In [28]:

```
n_correct = 0
n_total = 0

for inputs , targets in train_loader:
    inputs , targets = inputs.to(device) , targets.to(device)

    outputs = model(inputs)

    _ , predictions = torch.max(outputs,1)

    n_correct += (predictions == targets).sum().item()
    n_total += targets.shape[0]


train_acc = n_correct / n_total

n_correct = 0
n_total = 0

for inputs , targets in test_loader:
    inputs , targets = inputs.to(device) , targets.to(device)

    outputs = model(inputs)

    _ , predictions = torch.max(outputs,1)

    n_correct += (predictions == targets).sum().item()
    n_total += targets.shape[0]


test_acc = n_correct / n_total

print(f"Train Accuracy ; {train_acc}  Test Accuracy : {test_acc}")
```

Single Image Prediction

In [29]:

```
transform_index_to_disease = dataset.class_to_idx
```

In [30]:

```
transform_index_to_disease = dict(
    [(value, key) for key, value in transform_index_to_disease.items()]
)  # reverse the index
```

In [31]:

```
transform_index_to_disease
```

Out[31]:

```
{0: 'Apple___Apple_scab',
 1: 'Apple___Black_rot',
 2: 'Apple___Cedar_apple_rust',
 3: 'Apple___healthy',
 4: 'Background_without_leaves',
 5: 'Blueberry___healthy',
 6: 'Cherry___Powdery_mildew',
 7: 'Cherry___healthy',
 8: 'Corn___Cercospora_leaf_spot Gray_leaf_spot',
 9: 'Corn___Common_rust',
 10: 'Corn___Northern_Leaf_Blight',
 11: 'Corn___healthy',
 12: 'Grape___Black_rot',
 13: 'Grape___Esca_(Black_Measles)',
 14: 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
 15: 'Grape___healthy',
 16: 'Orange___Haunglongbing_(Citrus_greening)',
 17: 'Peach___Bacterial_spot',
 18: 'Peach___healthy',
 19: 'Pepper,_bell___Bacterial_spot',
 20: 'Pepper,_bell___healthy',
 21: 'Potato___Early_blight',
 22: 'Potato___Late_blight',
 23: 'Potato___healthy',
 24: 'Raspberry___healthy',
 25: 'Soybean___healthy',
 26: 'Squash___Powdery_mildew',
 27: 'Strawberry___Leaf_scorch',
 28: 'Strawberry___healthy',
 29: 'Tomato___Bacterial_spot',
 30: 'Tomato___Early_blight',
 31: 'Tomato___Late_blight',
 32: 'Tomato___Leaf_Mold',
 33: 'Tomato___Septoria_leaf_spot',
 34: 'Tomato___Spider_mites Two-spotted_spider_mite',
 35: 'Tomato___Target_Spot',
 36: 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
 37: 'Tomato___Tomato_mosaic_virus',
 38: 'Tomato___healthy'}
```

In [32]:

```
from PIL import Image
import torchvision.transforms.functional as TF
```

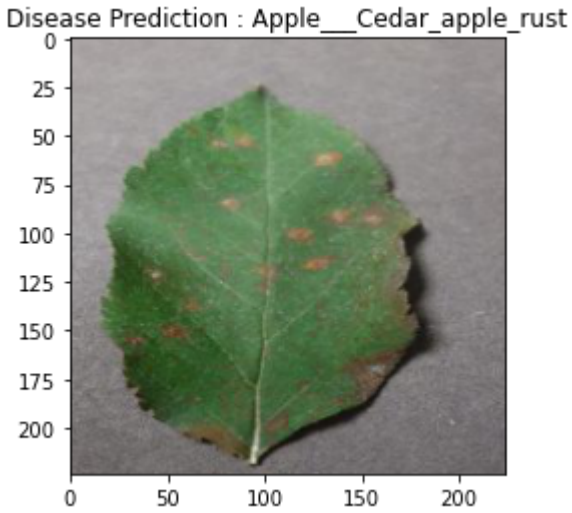
In [104]:

```
def single_prediction(image_path):
    image = Image.open(image_path)
    image = image.resize((224, 224))
    input_data = TF.to_tensor(image)
    input_data = input_data.view((-1, 3, 224, 224))
    output = model(input_data)
    output = output.detach().numpy()
    index = np.argmax(output)
    print("Original : ", image_path[12:-4])
    pred = transform_index_to_disease[index]
    plt.imshow(image)
    plt.title("Disease Prediction : " + pred)
    plt.show()
```

In [105]:

```
single_prediction("test_images/Apple_ceder_apple_rust.JPG")
```

```
Original :  Apple_ceder_apple_rust
```

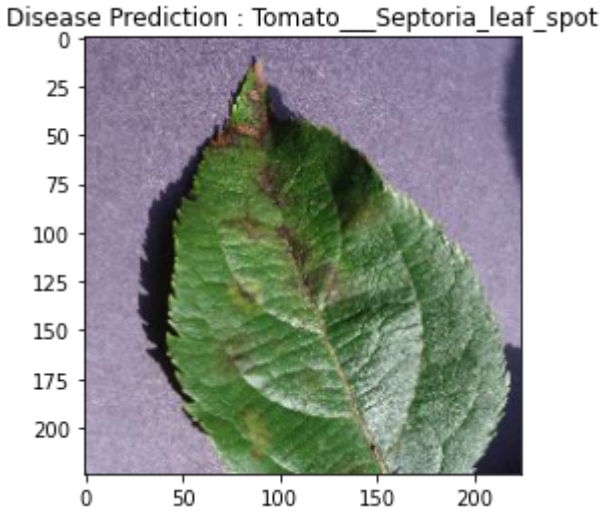


Wrong Prediction

In [106]:

```
single_prediction("test_images/Apple_scab.JPG")
```

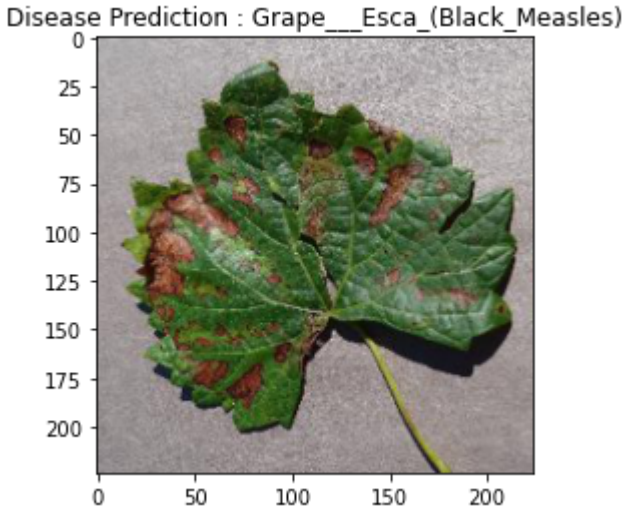
```
Original :  Apple_scab
```



In [107]:

```
single_prediction("test_images/Grape_esca.JPG")
```

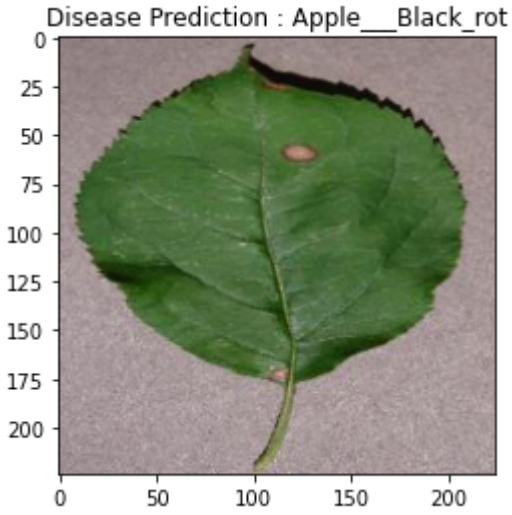
```
Original :  Grape_esca
```



In [108]:

```
single_prediction("test_images/apple_black_rot.JPG")
```

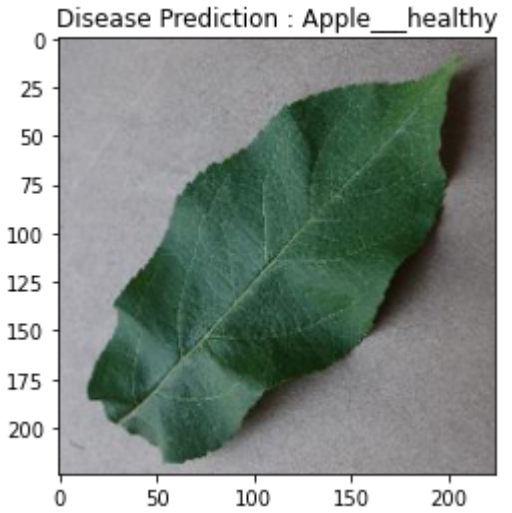
```
Original :  apple_black_rot
```



In [109]:

```
single_prediction("test_images/apple_healthy.JPG")
```

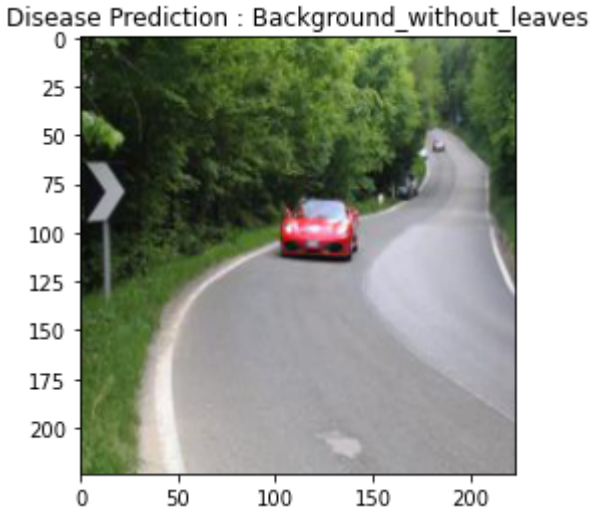
```
Original :  apple_healthy
```



In [110]:

```
single_prediction("test_images/background_without_leaves.jpg")
```

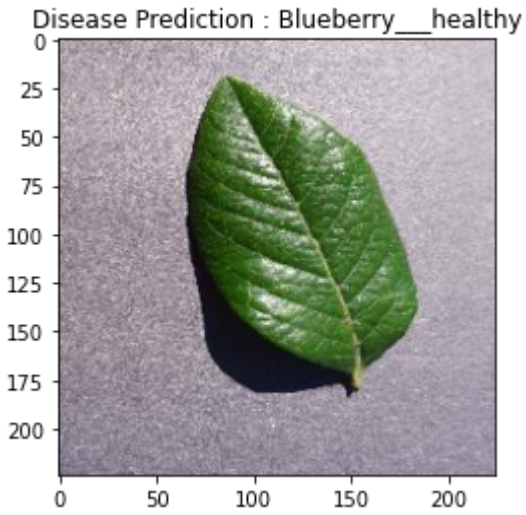
```
Original :  background_without_leaves
```



In [111]:

```
single_prediction("test_images/blueberry_healthy.JPG")
```

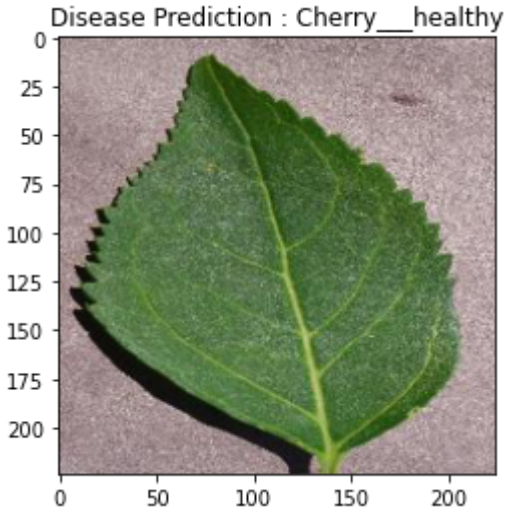
```
Original :  blueberry_healthy
```



In [112]:

```
single_prediction("test_images/cherry_healthy.JPG")
```

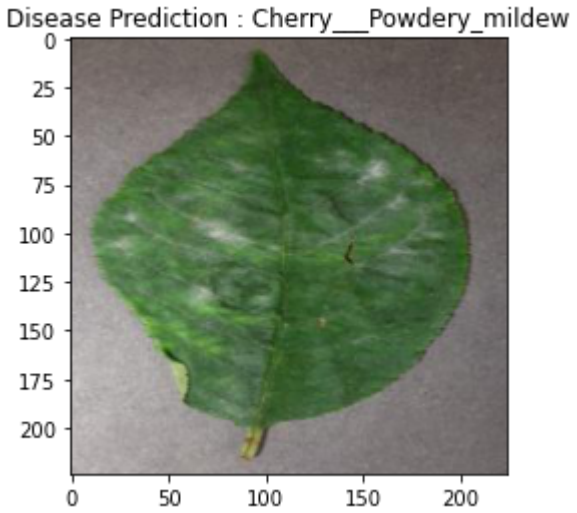
```
Original :  cherry_healthy
```



In [113]:

```
single_prediction("test_images/cherry_powdery_mildew.JPG")
```

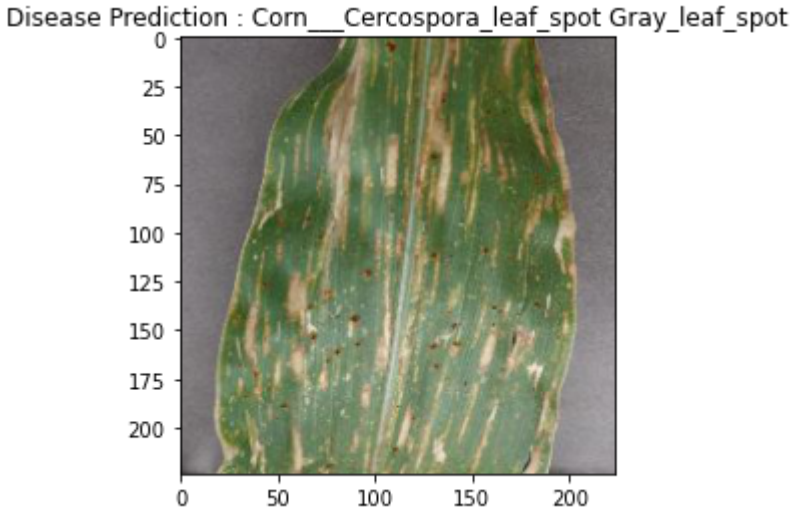
```
Original :  cherry_powdery_mildew
```



In [114]:

```
single_prediction("test_images/corn_cercospora_leaf.JPG")
```

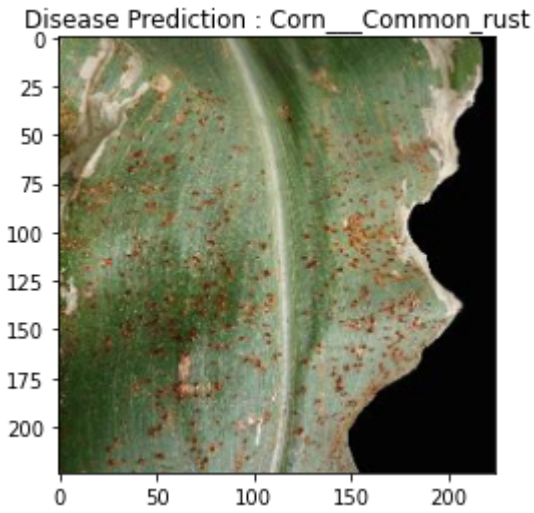
```
Original :  corn_cercospora_leaf
```



In [115]:

```
single_prediction("test_images/corn_common_rust.JPG")
```

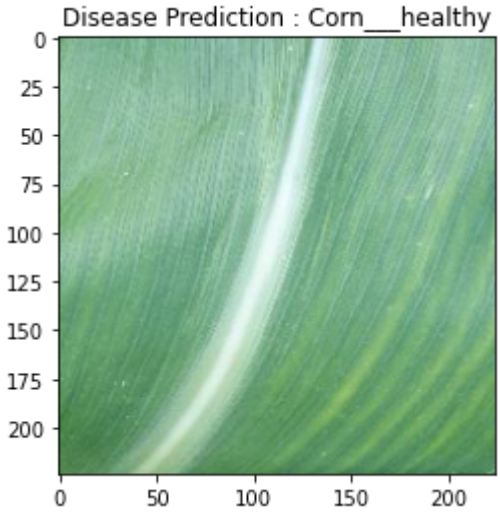
```
Original :  corn_common_rust
```



In [116]:

```
single_prediction("test_images/corn_healthy.jpg")
```

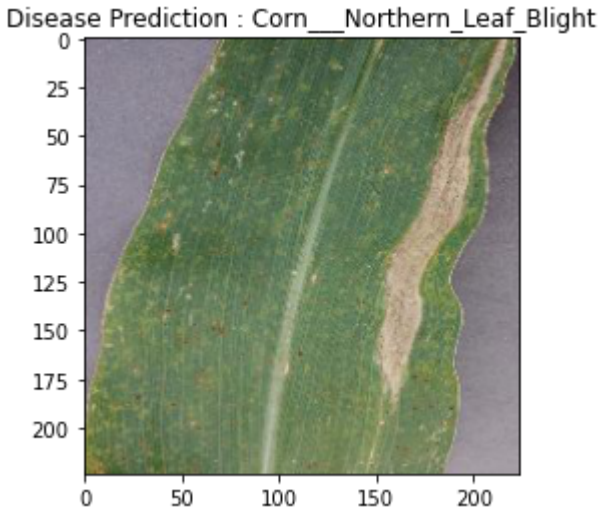
```
Original :  corn_healthy
```



In [117]:

```
single_prediction("test_images/corn_northen_leaf_blight.JPG")
```

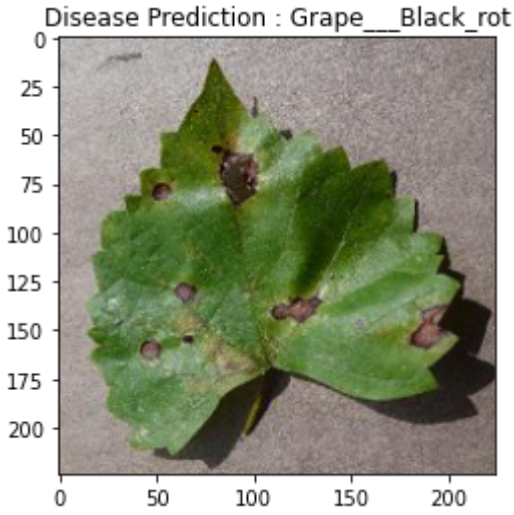
```
Original :  corn_northen_leaf_blight
```



In [118]:

```
single_prediction("test_images/grape_black_rot.JPG")
```

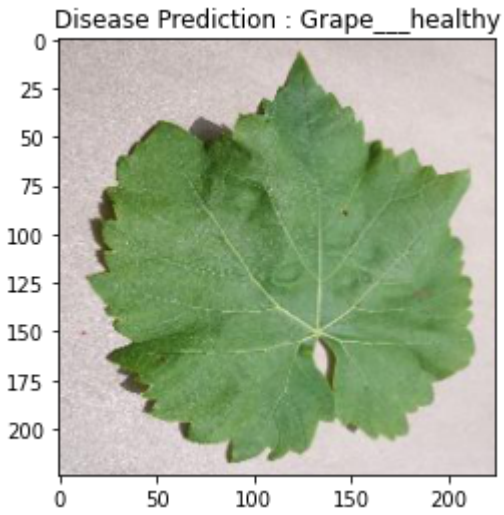
```
Original :  grape_black_rot
```



In [119]:

```
single_prediction("test_images/grape_healthy.JPG")
```

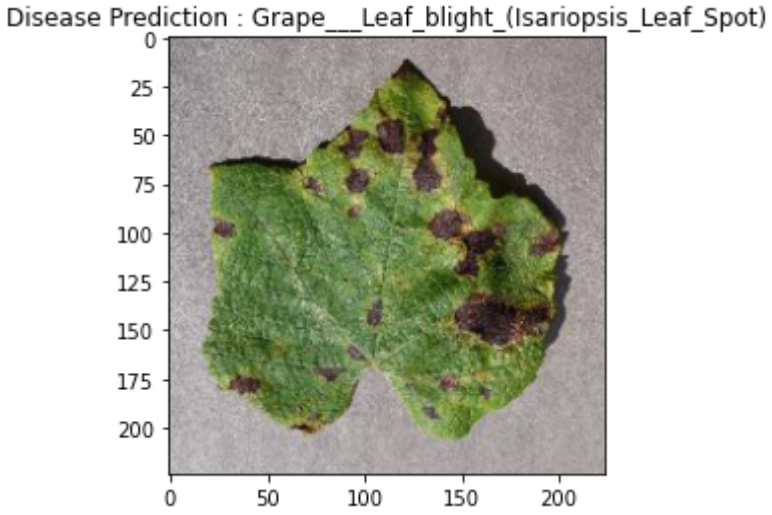
```
Original :  grape_healthy
```



In [120]:

```
single_prediction("test_images/grape_leaf_blight.JPG")
```

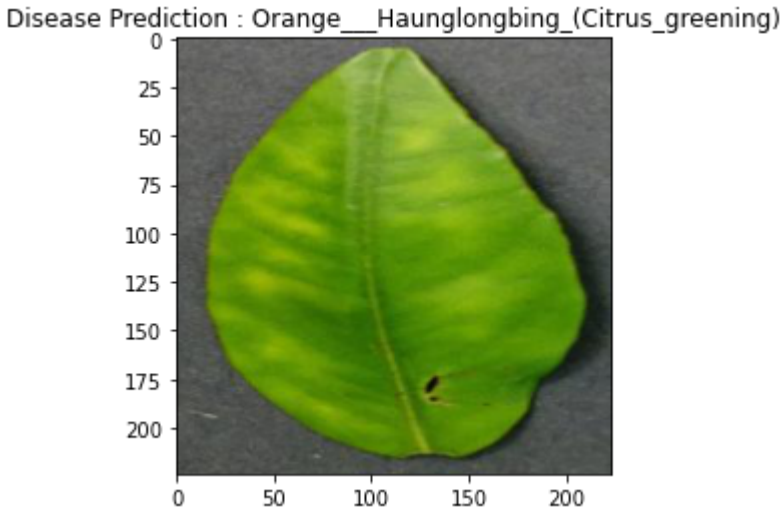
```
Original :  grape_leaf_blight
```



In [121]:

```
single_prediction("test_images/orange_haunglongbing.JPG")
```

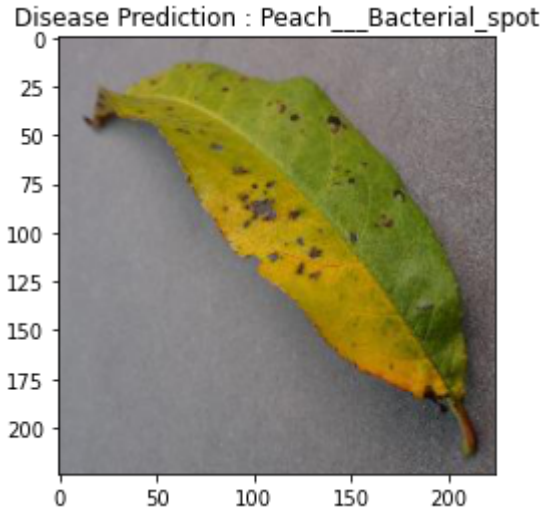
```
Original :  orange_haunglongbing
```



In [122]:

```
single_prediction("test_images/peach_bacterial_spot.JPG")
```

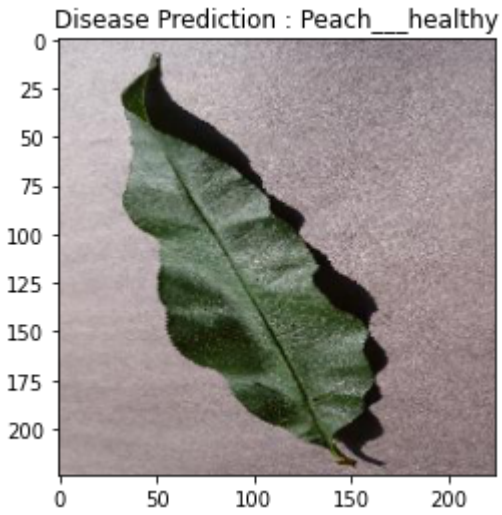
```
Original :  peach_bacterial_spot
```



In [123]:

```
single_prediction("test_images/peach_healthy.JPG")
```

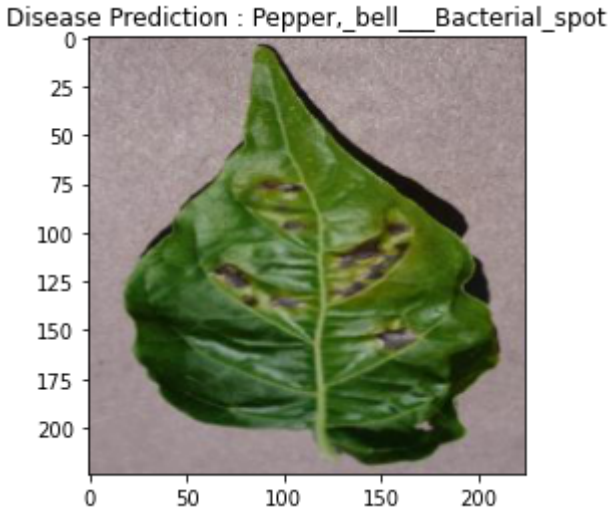
```
Original :  peach_healthy
```



In [124]:

```
single_prediction("test_images/pepper_bacterial_spot.JPG")
```

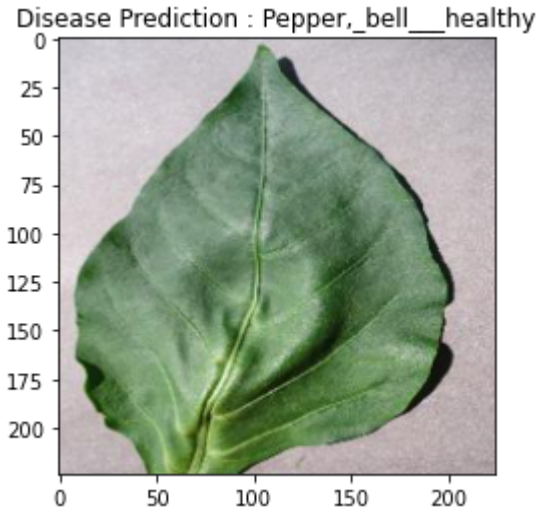
```
Original :  pepper_bacterial_spot
```



In [125]:

```
single_prediction("test_images/pepper_bell_healthy.JPG")
```

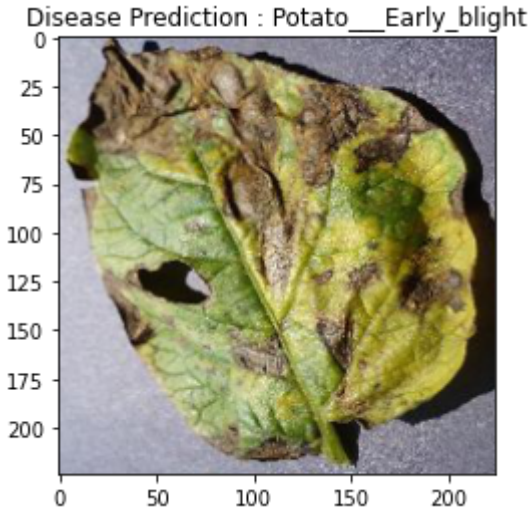
```
Original :  pepper_bell_healthy
```



In [126]:

```
single_prediction("test_images/potato_early_blight.JPG")
```

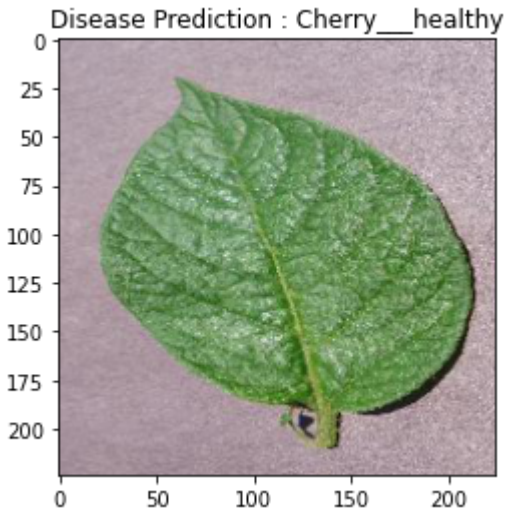
```
Original :  potato_early_blight
```



In [127]:

```
single_prediction("test_images/potato_healthy.JPG")
```

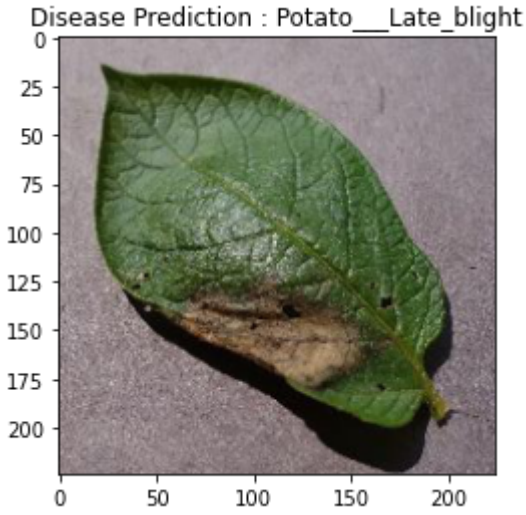
```
Original :  potato_healthy
```



In [128]:

```
single_prediction("test_images/potato_late_blight.JPG")
```

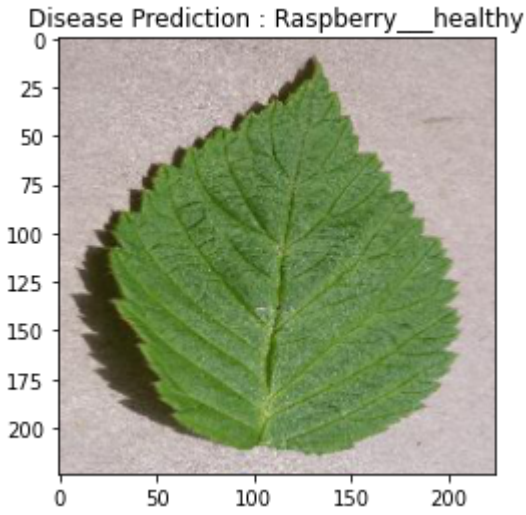
```
Original :  potato_late_blight
```



In [129]:

```
single_prediction("test_images/raspberry_healthy.JPG")
```

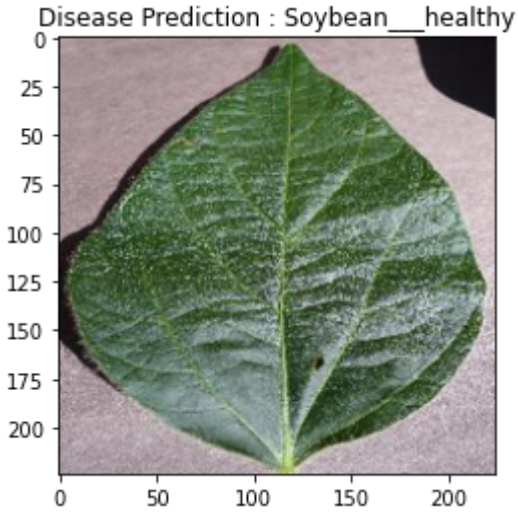
```
Original :  raspberry_healthy
```



In [130]:

```
single_prediction("test_images/soyaben healthy.JPG")
```

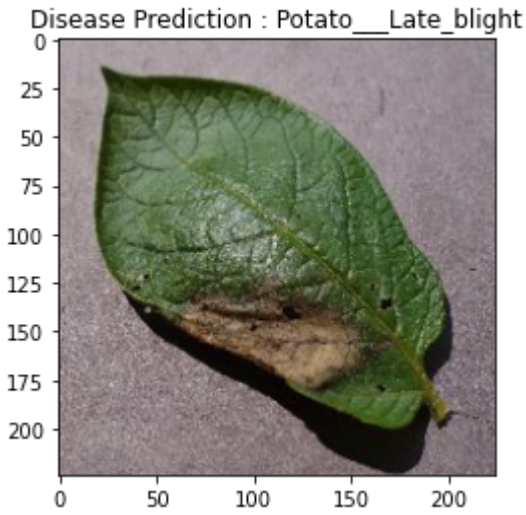
```
Original :  soyaben healthy
```



In [131]:

```
single_prediction("test_images/potato_late_blight.JPG")
```

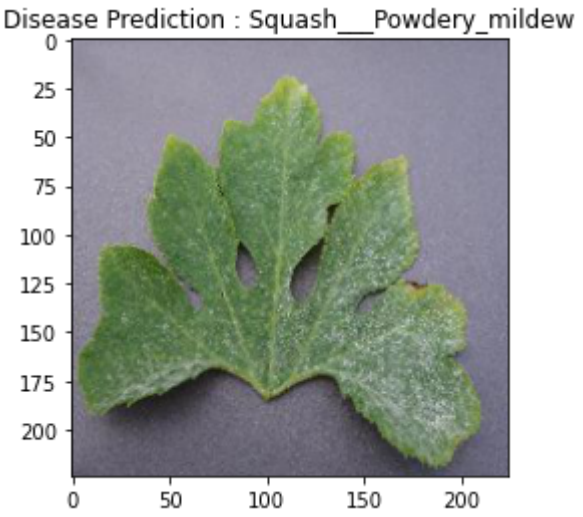
```
Original :  potato_late_blight
```



In [132]:

```
single_prediction("test_images/squash_powdery_mildew.JPG")
```

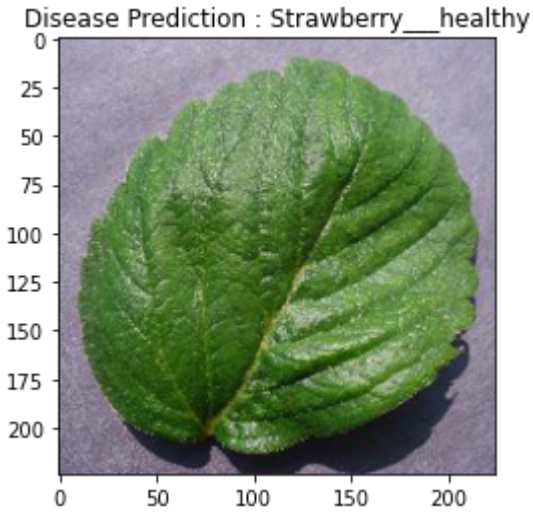
```
Original :  squash_powdery_mildew
```



In [133]:

```
single_prediction("test_images/starwberry_healthy.JPG")
```

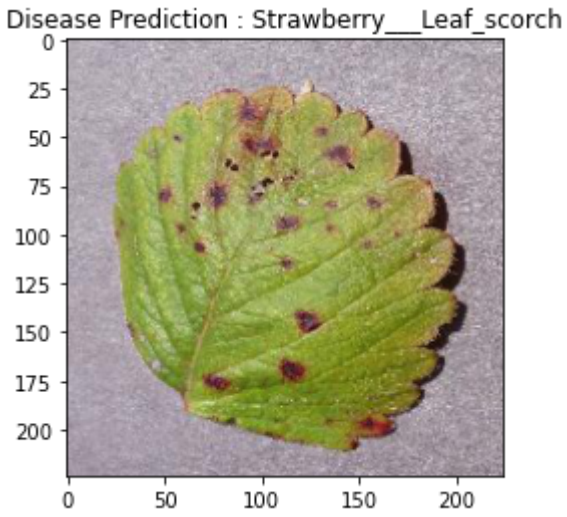
```
Original :  starwberry_healthy
```



In [134]:

```
single_prediction("test_images/starwberry_leaf_scorch.JPG")
```

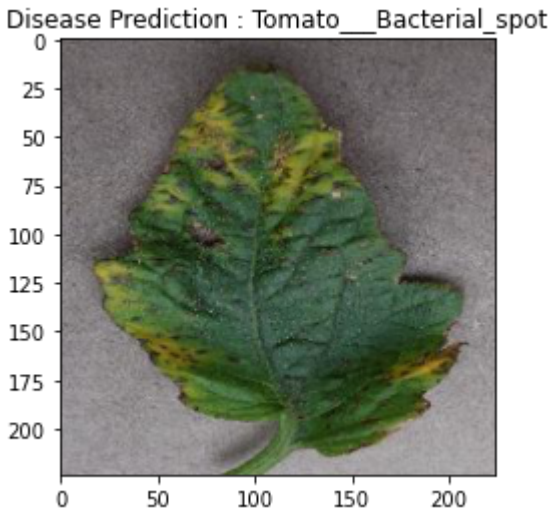
```
Original :  starwberry_leaf_scorch
```



In [135]:

```
single_prediction("test_images/tomato_bacterial_spot.JPG")
```

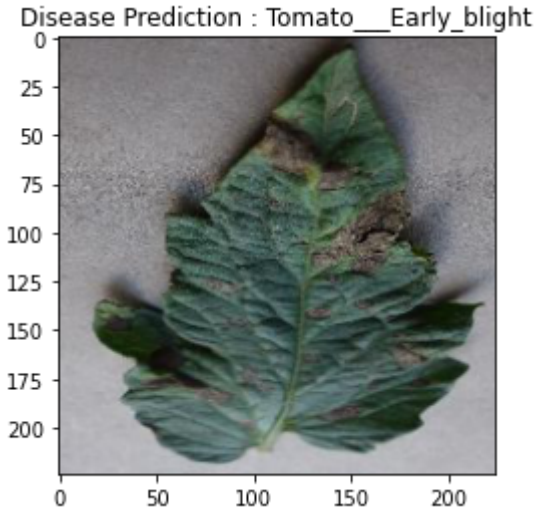
```
Original :  tomato_bacterial_spot
```



In [136]:

```
single_prediction("test_images/tomato_early_blight.JPG")
```

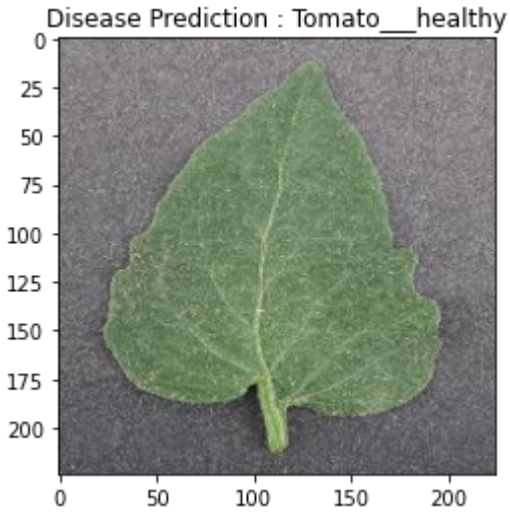
```
Original :  tomato_early_blight
```



In [137]:

```
single_prediction("test_images/tomato_healthy.JPG")
```

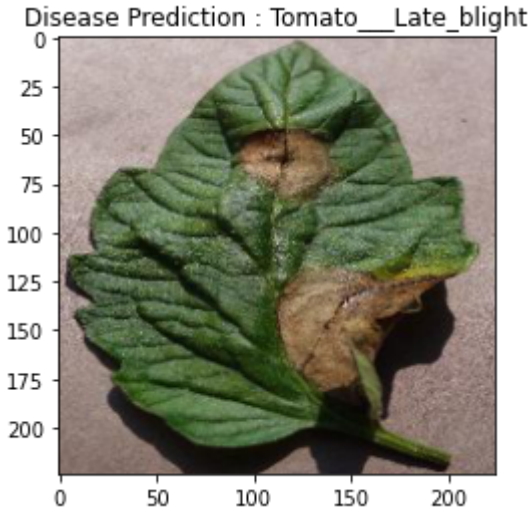
```
Original :  tomato_healthy
```



In [138]:

```
single_prediction("test_images/tomato_late_blight.JPG")
```

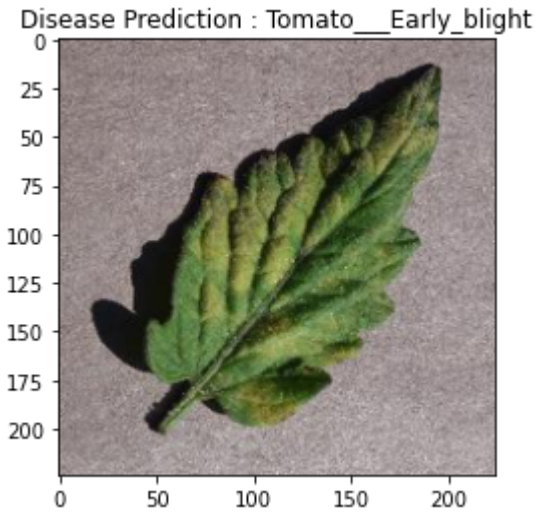
```
Original :  tomato_late_blight
```



In [139]:

```
single_prediction("test_images/tomato_leaf_mold.JPG")
```

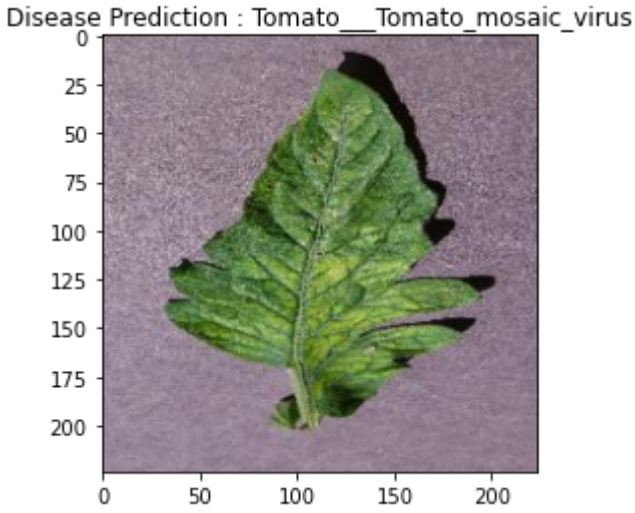
```
Original :  tomato_leaf_mold
```



In [140]:

```
single_prediction("test_images/tomato_mosaic_virus.JPG")
```

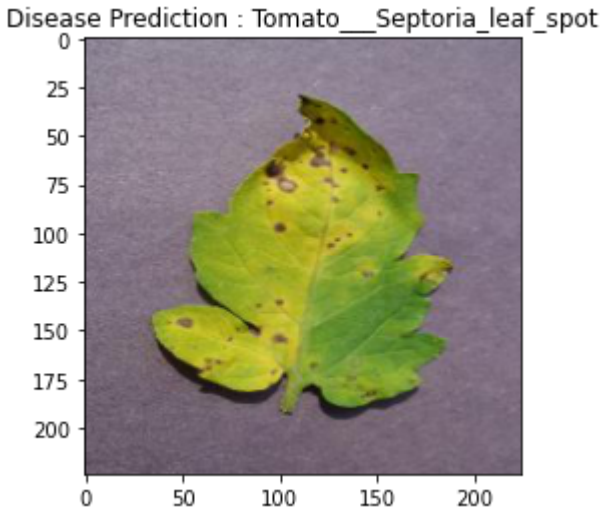
```
Original :  tomato_mosaic_virus
```



In [141]:

```
single_prediction("test_images/tomato_septoria_leaf_spot.JPG")
```

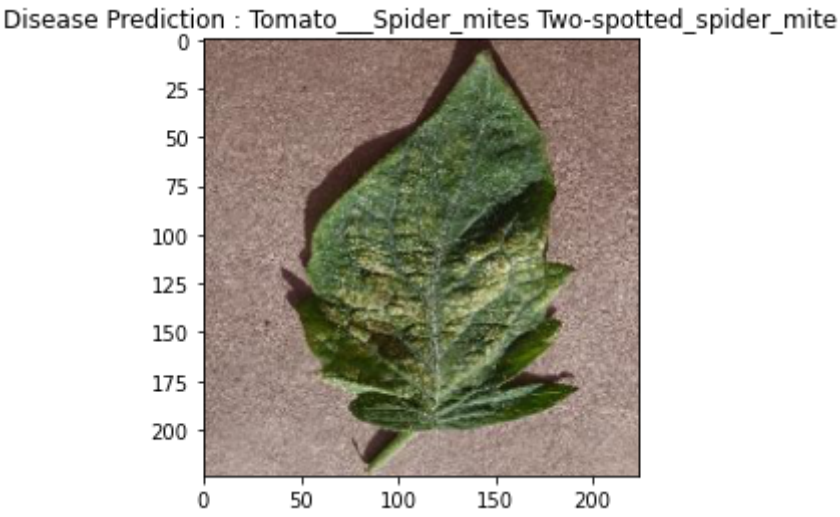
```
Original :  tomato_septoria_leaf_spot
```



In [142]:

```
single_prediction("test_images/tomato_spider_mites_two_spotted_spider_mites.JPG")
```

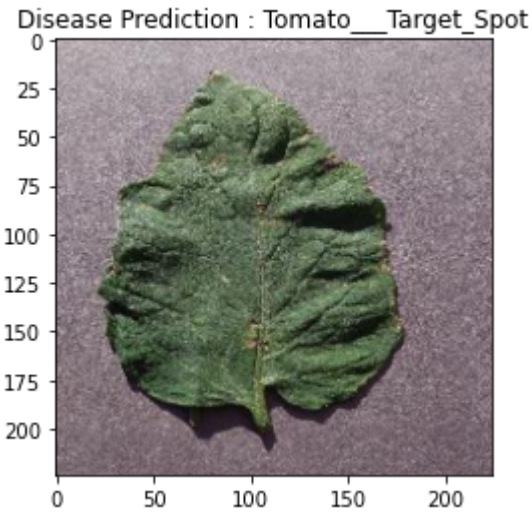
```
Original :  tomato_spider_mites_two_spotted_spider_mites
```



In [143]:

```
single_prediction("test_images/tomato_target_spot.JPG")
```

```
Original :  tomato_target_spot
```



In [144]:

```
single_prediction("test_images/tomato_yellow_leaf_curl_virus.JPG")
```

```
Original :  tomato_yellow_leaf_curl_virus
```

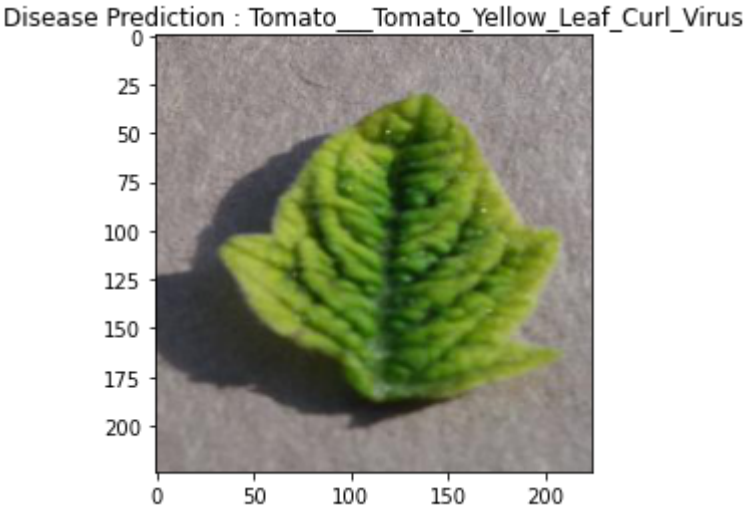
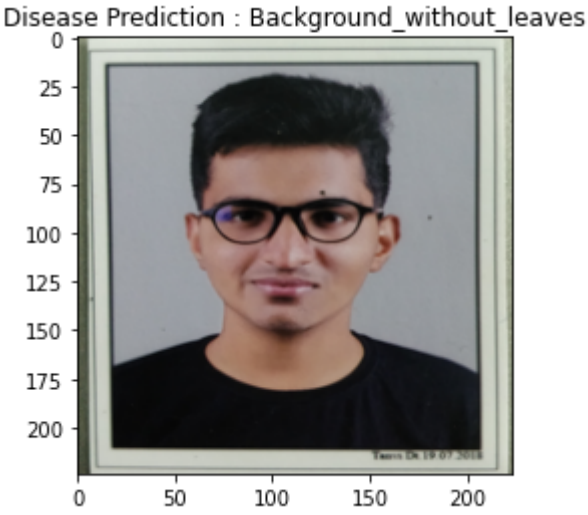


Image Outside of Dataset

In [145]:

```
single_prediction("PHOTO.jpg")
```

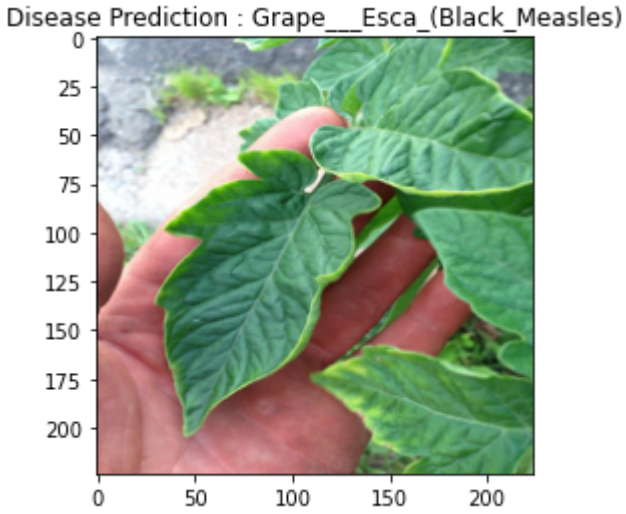
```
Original :
```



In [146]:

```
single_prediction("test_images/tomato_yellow_leaf_curl_virus2.jpg")
```

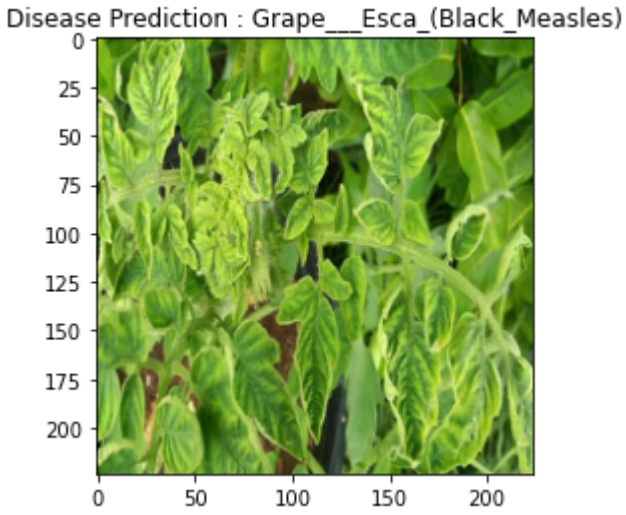
```
Original :  tomato_yellow_leaf_curl_virus2
```



In [147]:

```
single_prediction("test_images/tomato-leaf-curl-virus3.jpg")
```

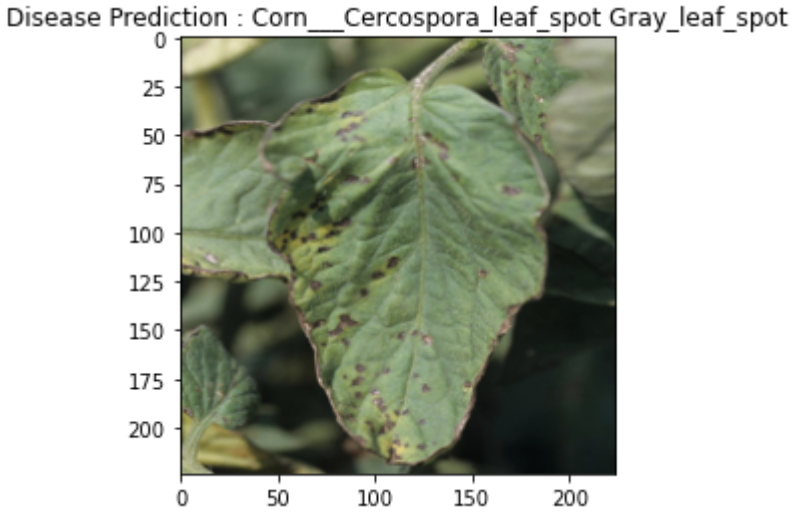
```
Original :  tomato-leaf-curl-virus3
```



In [148]:

```
single_prediction("test_images/tomato-bacterial-spot2.jpg")
```

```
Original :  tomato-bacterial-spot2
```



In [149]:

```
single_prediction("test_images/tomato-mold.jpg")
```

```
Original :  tomato-mold
```