In [1]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
C:\Users\KIIT\anaconda3\lib\site-packages\pandas\core\arrays\masked.py:60: UserWarnin
g: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currentl
y installed).
  from pandas.core import (
```

DATA PREPROCESSING

Training Image Preprocessing

In [19]:
```
import os
import shutil

# Directory containing images and annotation text file
directory = 'train'

# Read the annotation text file
with open(os.path.join(directory, '_annotations.txt'), 'r') as file:
    lines = file.readlines()

# Process each line in the annotation text file
for line in lines:
    parts = line.strip().split()
    image_name = parts[0]
    annotations = parts[1:]

    # Determine the class ID from the first annotation (assuming all annotations for a
    class_id = annotations[0].split(',')[-1]

    # Create a directory for the class if it doesn't exist
    class_dir = os.path.join(directory, f'class_{class_id}')
    if not os.path.exists(class_dir):
        os.makedirs(class_dir)

    # Move the image to the class directory
    image_path = os.path.join(directory, image_name)
    new_image_path = os.path.join(class_dir, image_name)
    shutil.move(image_path, new_image_path)

print('Images have been moved to separate folders based on annotations.')
```

```
Images have been moved to separate folders based on annotations.
```

In [2]:
```
training_set = tf.keras.utils.image_dataset_from_directory(
    './train/',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
```

```
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False,
)
```

```
Found 1236 files belonging to 3 classes.
```

VALIDATION

In [27]:
```
import os
import shutil

# Directory containing images and annotation text file
directory = 'valid'

# Read the annotation text file
with open(os.path.join(directory, '_annotations.txt'), 'r') as file:
    lines = file.readlines()

# Process each line in the annotation text file
for line in lines:
    parts = line.strip().split()
    image_name = parts[0]
    annotations = parts[1:]

    # Determine the class ID from the first annotation (assuming all annotations for a
    class_id = annotations[0].split(',')[-1]

    # Create a directory for the class if it doesn't exist
    class_dir = os.path.join(directory, f'class_{class_id}')
    if not os.path.exists(class_dir):
        os.makedirs(class_dir)

    # Move the image to the class directory
    image_path = os.path.join(directory, image_name)
    new_image_path = os.path.join(class_dir, image_name)
    shutil.move(image_path, new_image_path)

print('Images have been moved to separate folders based on annotations.')
```

```
Images have been moved to separate folders based on annotations.
```

In [3]:
```
validation_set = tf.keras.utils.image_dataset_from_directory(
    './valid/',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
```

```
    follow_links=False,
    crop_to_aspect_ratio=False,
)
```

```
Found 200 files belonging to 3 classes.
```

In [4]:
```
training_set
```

Out[4]:
```
<_PrefetchDataset element_spec=(TensorSpec(shape=(None, 128, 128, 3), dtype=tf.float3
2, name=None), TensorSpec(shape=(None, 3), dtype=tf.float32, name=None))>
```

In [5]:
```
for x,y in training_set:
    print(x,x.shape)
    print(y,y.shape)
    break
```

```
tf.Tensor(
[[[209. 207. 218.]
  [219. 217. 228.]
  [132. 130. 141.]
  ...
  [154. 154. 152.]
  [147. 146. 152.]
  [ 54.  53.  49.]]

 [[103. 102. 110.]
  [141. 140. 148.]
  [136. 135. 143.]
  ...
  [ 75.  76.  70.]
  [144. 157. 137.]
  [172. 196. 170.]]

 [[ 94.  93.  99.]
  [133. 132. 138.]
  [133. 132. 138.]
  ...
  [157. 184. 153.]
  [172. 211. 158.]
  [195. 239. 204.]]

 ...

 [[122. 121. 116.]
  [ 74.  73.  69.]
  [226. 225. 223.]
  ...
  [180. 197. 155.]
  [223. 255. 201.]
  [224. 247. 201.]]

 [[ 14.  10.   9.]
  [ 19.  15.  16.]
  [169. 164. 170.]
  ...
  [214. 239. 197.]
  [204. 247. 201.]
  [207. 246. 202.]]

 [[  0.  10.   0.]
  [ 60.  73.  63.]
  [185. 198. 191.]
  ...
  [184. 195. 161.]
  [220. 253. 224.]
  [222. 253. 222.]]]


[[[210. 211. 206.]
  [128. 129. 124.]
  [210. 211. 206.]
  ...
  [  1.   0.   2.]
  [  3.  34.   3.]
  [ 51. 115.  54.]]
```

```
 [[216. 217. 212.]
  [214. 215. 210.]
  [210. 211. 206.]
  ...
  [  7.   0.   1.]
  [ 63.  78.  59.]
  [ 71. 112.  70.]]

 [[197. 198. 193.]
  [215. 216. 211.]
  [212. 213. 208.]
  ...
  [ 16.  12.  11.]
  [ 95. 105.  94.]
  [ 99. 124. 102.]]

 ...

 [[ 93.  89.  86.]
  [173. 169. 166.]
  [ 75.  71.  68.]
  ...
  [  0.   0.   0.]
  [  0.   0.   0.]
  [  0.   0.   0.]]

 [[148. 144. 141.]
  [143. 139. 136.]
  [ 64.  60.  57.]
  ...
  [  0.   0.   0.]
  [  0.   0.   0.]
  [  0.   0.   0.]]

 [[166. 162. 159.]
  [140. 136. 133.]
  [163. 159. 156.]
  ...
  [  0.   0.   0.]
  [  0.   0.   0.]
  [  0.   0.   0.]]]


[[[194. 226. 163.]
  [194. 247. 175.]
  [175. 219. 132.]
  ...
  [170. 222. 160.]
  [174. 222. 162.]
  [151. 196. 137.]]

 [[161. 204. 124.]
  [187. 255. 154.]
  [189. 254. 138.]
  ...
  [167. 213. 151.]
  [159. 200. 140.]
  [183. 223. 163.]]

 [[214. 242. 183.]
```

```
  [178. 237. 145.]
  [181. 240. 132.]
  ...
  [180. 231. 165.]
  [177. 226. 161.]
  [163. 209. 145.]]

 ...

 [[ 56.  56.  56.]
  [ 56.  56.  56.]
  [ 56.  56.  56.]
  ...
  [153. 155. 141.]
  [123. 125. 111.]
  [137. 139. 125.]]

 [[ 56.  56.  56.]
  [ 56.  56.  56.]
  [ 56.  56.  56.]
  ...
  [145. 146. 140.]
  [ 87.  88.  82.]
  [ 90.  91.  85.]]

 [[ 56.  56.  56.]
  [ 56.  56.  56.]
  [ 56.  56.  56.]
  ...
  [ 52.  52.  50.]
  [ 54.  54.  52.]
  [ 56.  56.  54.]]]


...


[[[137. 137. 127.]
  [ 99.  99.  91.]
  [ 58.  58.  48.]
  ...
  [240. 255. 172.]
  [251. 246. 214.]
  [251. 255. 233.]]

 [[169. 171. 157.]
  [112. 114. 101.]
  [ 41.  41.  41.]
  ...
  [152. 185.  96.]
  [247. 255. 196.]
  [245. 251. 205.]]

 [[131. 138. 122.]
  [131. 140. 111.]
  [ 60.  67.  51.]
  ...
  [197. 228. 152.]
  [225. 255. 169.]
  [230. 251. 184.]]
```

```
...

[[ 26.  26.  26.]
 [ 26.  26.  26.]
 [ 26.  26.  26.]
 ...
 [ 26.  26.  26.]
 [ 26.  26.  26.]
 [ 26.  26.  26.]]

[[ 26.  26.  26.]
 [ 26.  26.  26.]
 [ 26.  26.  26.]
 ...
 [ 26.  26.  26.]
 [ 26.  26.  26.]
 [ 26.  26.  26.]]

[[ 26.  26.  26.]
 [ 26.  26.  26.]
 [ 26.  26.  26.]
 ...
 [ 26.  26.  26.]
 [ 26.  26.  26.]
 [ 26.  26.  26.]]]


[[[ 33.  44.  10.]
  [ 57.  65.  41.]
  [ 28.  30.  17.]
  ...
  [ 81. 155.  30.]
  [ 75. 144.  28.]
  [ 71. 136.  16.]]

 [[ 59.  67.  52.]
  [ 23.  29.  19.]
  [ 49.  49.  47.]
  ...
  [ 84. 156.  28.]
  [ 72. 141.  24.]
  [ 66. 136.  14.]]

 [[ 63.  70.  62.]
  [ 43.  48.  42.]
  [ 50.  50.  48.]
  ...
  [ 74. 139.  23.]
  [ 83. 155.  30.]
  [102. 179.  37.]]

 ...

 [[101. 156.  91.]
  [ 80. 114.  63.]
  [ 43.  87.  34.]
  ...
  [151. 187. 123.]
  [162. 182. 133.]
```

```
  [150. 170. 119.]]

 [[118. 205.  90.]
  [107. 159.  93.]
  [164. 233. 144.]
  ...
  [189. 221. 156.]
  [147. 189. 117.]
  [132. 178. 103.]]

 [[134. 220. 109.]
  [151. 241. 127.]
  [ 58. 106.  46.]
  ...
  [138. 169. 112.]
  [142. 186. 109.]
  [129. 170.  92.]]]


[[[152. 187. 105.]
  [136. 167.  99.]
  [114. 136.  87.]
  ...
  [209. 222. 194.]
  [157. 167. 143.]
  [151. 185. 111.]]

 [[117. 176.  70.]
  [106. 147.  71.]
  [ 89. 110.  67.]
  ...
  [242. 255. 216.]
  [207. 242. 162.]
  [215. 255. 152.]]

 [[ 86. 154.  55.]
  [120. 159.  92.]
  [ 51.  58.  25.]
  ...
  [239. 240. 209.]
  [183. 230. 126.]
  [209. 255. 156.]]

 ...

 [[143. 144.  86.]
  [201. 204. 151.]
  [255. 255. 233.]
  ...
  [ 43.  43.  43.]
  [ 43.  43.  43.]
  [ 43.  43.  43.]]

 [[161. 166. 102.]
  [255. 255. 240.]
  [255. 255. 250.]
  ...
  [ 43.  43.  43.]
  [ 43.  43.  43.]
  [ 43.  43.  43.]]
```

```
  [[255. 255. 230.]
   [253. 252. 250.]
   [254. 255. 248.]
   ...
   [ 43.  43.  43.]
   [ 43.  43.  43.]
   [ 43.  43.  43.]]]], shape=(32, 128, 128, 3), dtype=float32) (32, 128, 128, 3)
tf.Tensor(
[[0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]], shape=(32, 3), dtype=float32) (32, 3)
```

Building Model

In [6]:
```
from tensorflow.keras.layers import Dense,Conv2D,MaxPool2D,Flatten,Dropout
from tensorflow.keras.models import Sequential
```

In [7]:
```
model=Sequential()
```

In [8]:
```
## Bulding Cobvolution Model
```

In [9]:
```
model.add(Conv2D(filters=32,kernel_size=3,padding='same',activation='relu',input_shape
model.add(Conv2D(filters=32,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

```
C:\Users\KIIT\anaconda3\lib\site-packages\keras\src\layers\convolutional\base_conv.p
y:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
n using Sequential models, prefer using an `Input(shape)` object as the first layer i
n the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [10]:
```
model.add(Conv2D(filters=64,kernel_size=3,padding='same',activation='relu',input_shape
model.add(Conv2D(filters=64,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

In [11]:
```
model.add(Conv2D(filters=128,kernel_size=3,padding='same',activation='relu',input_shap
model.add(Conv2D(filters=128,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

In [12]:
```
model.add(Conv2D(filters=256,kernel_size=3,padding='same',activation='relu',input_shap
model.add(Conv2D(filters=256,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

In [13]:
```
model.add(Conv2D(filters=512,kernel_size=3,padding='same',activation='relu',input_shap
model.add(Conv2D(filters=512,kernel_size=3,activation='relu'))
model.add(MaxPool2D(pool_size=2,strides=2))
```

In [14]:
```
model.add(Dropout(0.25))
```

In [15]:
```
model.add(Flatten())
```

In [16]:
```
from tensorflow.keras.regularizers import l2


model.add(Dense(256, activation='relu', kernel_regularizer=l2(0.01)))
```

In [17]:
```
model.add(Dropout(0.4))
```

In [18]:
```
#output layer
model.add(Dense(units=3,activation='softmax'))
```

In [19]:
```
import os
os.environ['TF_USE_LEGACY_KERAS'] = 'True'
import tensorflow as tf
```

Compilling Model

In [20]:
```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [21]:
```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Pa
conv2d (Conv2D)	(None, 128, 128, 32)	
conv2d_1 (Conv2D)	(None, 126, 126, 32)	
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	
conv2d_2 (Conv2D)	(None, 63, 63, 64)	1
conv2d_3 (Conv2D)	(None, 61, 61, 64)	3
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	
conv2d_4 (Conv2D)	(None, 30, 30, 128)	7
conv2d_5 (Conv2D)	(None, 28, 28, 128)	14
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	
conv2d_6 (Conv2D)	(None, 14, 14, 256)	29
conv2d_7 (Conv2D)	(None, 12, 12, 256)	59
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	
conv2d_8 (Conv2D)	(None, 6, 6, 512)	1,18
conv2d_9 (Conv2D)	(None, 4, 4, 512)	2,35
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 512)	
dropout (Dropout)	(None, 2, 2, 512)	
flatten (Flatten)	(None, 2048)	
dense (Dense)	(None, 256)	52
dropout_1 (Dropout)	(None, 256)	
dense_1 (Dense)	(None, 3)	

Total params: 5,237,539 (19.98 MB)

Trainable params: 5,237,539 (19.98 MB)

MODEL TRAINING

In [22]:
```
training_history=model.fit(x=training_set,validation_data=validation_set,epochs=10)
```

```
Epoch 1/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 42s 998ms/step - accuracy: 0.4032 - loss: 5.8742 - val_acc
uracy: 0.2750 - val_loss: 5.1589
Epoch 2/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 46s 1s/step - accuracy: 0.4235 - loss: 5.0421 - val_accura
cy: 0.5100 - val_loss: 4.6415
Epoch 3/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 46s 1s/step - accuracy: 0.4803 - loss: 4.6121 - val_accura
cy: 0.5450 - val_loss: 4.2576
Epoch 4/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.4454 - loss: 4.2250 - val_accura
cy: 0.4100 - val_loss: 3.9762
Epoch 5/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 44s 1s/step - accuracy: 0.4833 - loss: 3.8939 - val_accura
cy: 0.2900 - val_loss: 3.7695
Epoch 6/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.4777 - loss: 3.5970 - val_accura
cy: 0.4350 - val_loss: 3.3542
Epoch 7/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.4877 - loss: 3.3305 - val_accura
cy: 0.4200 - val_loss: 3.1969
Epoch 8/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.5172 - loss: 3.0764 - val_accura
cy: 0.4400 - val_loss: 2.9180
Epoch 9/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.5058 - loss: 2.8837 - val_accura
cy: 0.3000 - val_loss: 3.0200
Epoch 10/10
39/39 ━━━━━━━━━━━━━━━━━━━━ 45s 1s/step - accuracy: 0.5104 - loss: 2.7411 - val_accura
cy: 0.3550 - val_loss: 2.6718
```

In [ ]:

MODEL EVALUATION

In [23]:
```
#Training set Accuracy
train_loss, train_acc = model.evaluate(training_set)
print('Training accuracy:', train_acc)
```

```
39/39 ━━━━━━━━━━━━━━━━━━━━ 9s 223ms/step - accuracy: 0.5535 - loss: 2.5545
Training accuracy: 0.5542071461677551
```

In [24]:
```
#Validation set Accuracy
val_loss, val_acc = model.evaluate(validation_set)
print('Validation accuracy:', val_acc)
```

```
7/7 ━━━━━━━━━━━━━━━━━━━━ 1s 188ms/step - accuracy: 0.3212 - loss: 2.6946
Validation accuracy: 0.35499998927116394
```

SAVING MODEL

In [25]:
```
model.save('trained_plant_disease_model.keras')
```

In [26]:
```
training_history.history
```

Out[26]:

```
{'accuracy': [0.3923948109149933,
  0.4279935359954834,
  0.4538834890708923,
  0.4563106894493103,
  0.48543688654899597,
  0.47653722763061523,
  0.491909384727478,
  0.5064724683761597,
  0.5105177760124207,
  0.5177993774414062],
 'loss': [5.550051689147949,
  4.94712495803833,
  4.528512477874756,
  4.154115200042725,
  3.8225958347320557,
  3.537806749343872,
  3.274664878845215,
  3.0392541885375977,
  2.8498735427856445,
  2.686023235321045],
 'val_accuracy': [0.2750000059604645,
  0.5099999904632568,
  0.5450000166893005,
  0.4099999964237213,
  0.28999999165534973,
  0.4350000023841858,
  0.41999998688697815,
  0.4399999976158142,
  0.30000001192092896,
  0.35499998927116394],
 'val_loss': [5.158854961395264,
  4.641493797302246,
  4.257646560668945,
  3.976235866546631,
  3.7694733142852783,
  3.554166269302368,
  3.1968741416931152,
  2.918027400970459,
  3.0200390815734863,
  2.671832323074341]}
```


In [27]:
```
#Recording History in json
import json
with open('training_hist.json','w') as f:
  json.dump(training_history.history,f)
```

In [28]:
```
print(training_history.history.keys())
```

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

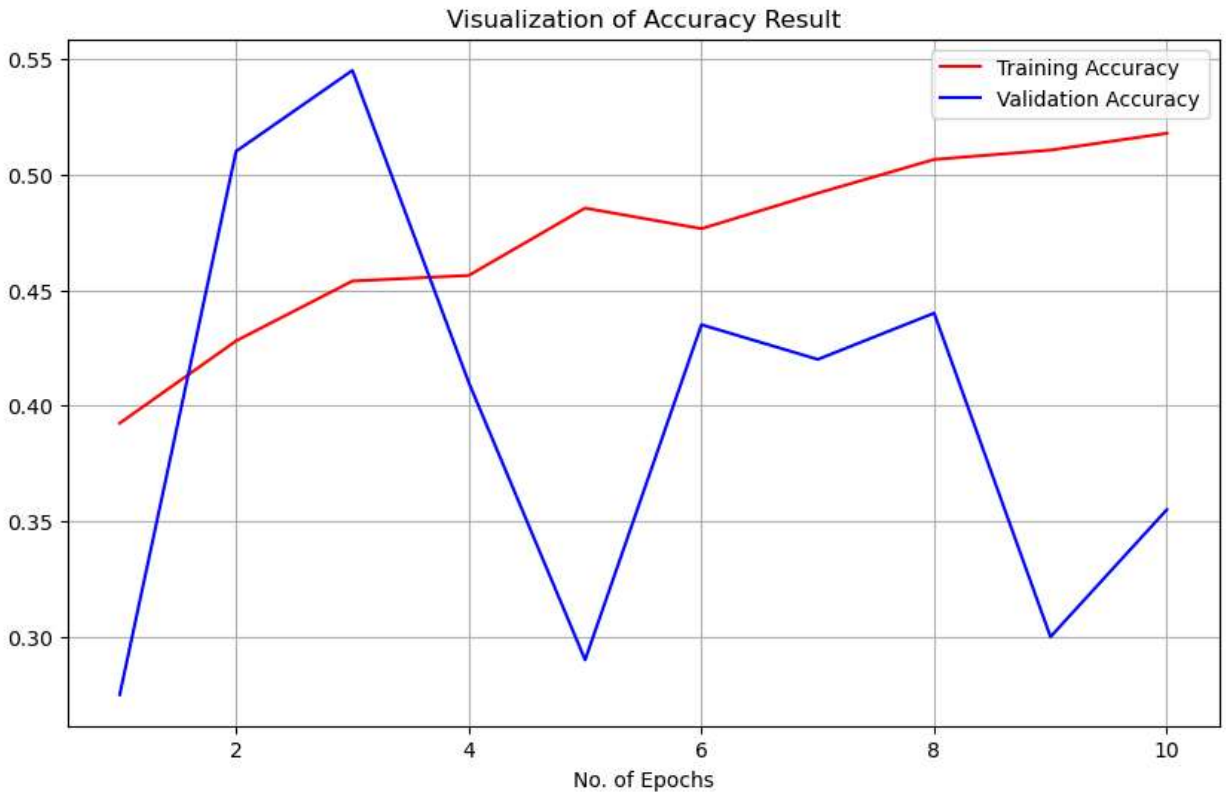
In [29]:
```
training_history.history['accuracy']
```

Out[29]:
```
[0.3923948109149933,
 0.4279935359954834,
 0.45388349890708923,
 0.4563106894493103,
 0.48543688654899597,
 0.47653722763061523,
 0.491909384727478,
 0.5064724683761597,
 0.5105177760124207,
 0.5177993774414062]
```

Accuracy Visualization

In [32]:
```
epochs = range(1, min(len(training_history.history['accuracy']) + 1,
                      len(training_history.history['val_accuracy']) + 1))

plt.figure(figsize=(10, 6))
plt.plot(epochs, training_history.history['accuracy'][:len(epochs)],
         color='red', label='Training Accuracy')
plt.plot(epochs, training_history.history['val_accuracy'][:len(epochs)],
         color='blue', label='Validation Accuracy')
plt.xlabel('No. of Epochs')
plt.title('Visualization of Accuracy Result')
plt.legend()
plt.grid(True)
plt.show()
```



Some other metrics for model evaluation

In [33]:
```
class_name = validation_set.class_names
```

In [34]:
```
test_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    class_names=None,
    color_mode="rgb",
    batch_size=1,
    image_size=(128, 128),
    shuffle=False,
    seed=None,
    validation_split=None,
    subset=None,
    interpolation="bilinear",
    follow_links=False,
    crop_to_aspect_ratio=False
)
```

```
Found 200 files belonging to 3 classes.
```

In [35]:
```
y_pred = model.predict(test_set)
predicted_categories = tf.argmax(y_pred, axis=1)
```

```
200/200 ━━━━━━━━━━━━━━━━━━━━ 3s 16ms/step
```

In [36]:
```
true_categories = tf.concat([y for x, y in test_set], axis=0)
Y_true = tf.argmax(true_categories, axis=1)
```

In [37]:
```
Y_true
```

Out[37]:
```
<tf.Tensor: shape=(200,), dtype=int64, numpy=
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2], dtype=int64)>
```

In [38]:
```
predicted_categories
```

Out[38]:
```
<tf.Tensor: shape=(200,), dtype=int64, numpy=
array([2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 2, 0, 0, 1, 2, 2, 2, 2, 0, 2, 2,
       0, 2, 2, 2, 1, 0, 2, 0, 2, 1, 2, 2, 0, 0, 2, 2, 0, 0, 0, 2, 0, 2,
       0, 0, 2, 0, 2, 2, 2, 2, 1, 2, 0, 0, 2, 0, 2, 2, 2, 0, 2, 0, 1, 2,
       2, 1, 0, 2, 2, 2, 2, 0, 2, 2, 1, 2, 1, 2, 0, 2, 2, 1, 2, 2, 2, 2,
       2, 2, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 0, 0, 0, 2, 0, 1, 0, 2, 0,
       2, 2, 2, 0, 2, 1, 2, 0, 2, 2, 2, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1,
       2, 2, 2, 2, 2, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 2, 0, 0, 2, 2,
       2, 2, 1, 0, 1, 1, 2, 1, 2, 2, 2, 2, 0, 2, 0, 2, 2, 2, 0, 2, 1, 1,
       0, 1, 2, 2, 1, 0, 1, 2, 2, 0, 1, 2, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2,
       1, 1], dtype=int64)>
```

In [39]:
```
from sklearn.metrics import confusion_matrix,classification_report
cm = confusion_matrix(Y_true,predicted_categories)
```

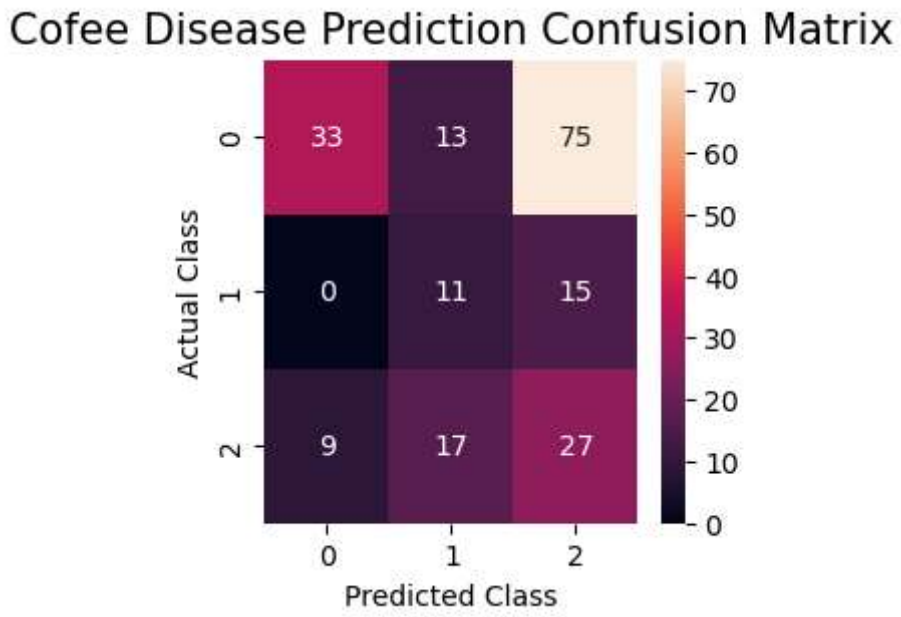
In [40]:
```
# Precision Recall Fscore
print(classification_report(Y_true,predicted_categories,target_names=class_name))
```

	precision	recall	f1-score	support
class_0	0.79	0.27	0.40	121
class_1	0.27	0.42	0.33	26
class_2	0.23	0.51	0.32	53
accuracy			0.35	200
macro avg	0.43	0.40	0.35	200
weighted avg	0.57	0.35	0.37	200

Confusion Matrix Visualization

In [41]:
```
plt.figure(figsize=(3, 3))
sns.heatmap(cm,annot=True,annot_kws={"size": 10})

plt.xlabel('Predicted Class',fontsize = 10)
plt.ylabel('Actual Class',fontsize = 10)
plt.title('Cofee Disease Prediction Confusion Matrix',fontsize = 15)
plt.show()
```



In [42]:
```
class_name
```

Out[42]:
```
['class_0', 'class_1', 'class_2']
```

In [ ]:

In [ ]:

In [ ]: