In [29]:
```
import tensorflow as tf
from tensorflow.keras import models, layers
from tensorflow.keras.layers import Dense
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import EarlyStopping
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

In [2]:
```
iz = 244
bs = 32
```

In [3]:
```
training_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "train",
    shuffle = True,
    seed = 16,
    image_size = (iz, iz),
    batch_size = bs
)
```

```
Found 7222 files belonging to 4 classes.
```

In [4]:
```
for images, labels in training_ds.take(1):
    print(images.shape, labels.shape)
    print(images.dtype, labels.dtype)
```

```
(32, 244, 244, 3) (32,)
<dtype: 'float32'> <dtype: 'int32'>
```

In [5]:
```
amount = int(len(training_ds) * 20/100)
# amount
validation_ds = training_ds.take(amount)
```

In [6]:
```
testing_ds = tf.keras.preprocessing.image_dataset_from_directory(
    "test",
    shuffle = True,
    seed = 16,
    image_size = (iz, iz),
    batch_size = bs
)
```

```
Found 1805 files belonging to 4 classes.
```

In [7]:
```
class_names = training_ds.class_names
class_names

class_number = len(class_names)
class_number
```

Out [7]: 4

In [59]:
```
from tensorflow.keras.applications import ResNet50

model = models.Sequential()
model.add(ResNet50(include_top=False, pooling='max', weights='imagenet', input_shape = (iz, iz, 3)))
model.add(Dense(class_number, activation='softmax'))
model.layers[0].trainable = False
```

In [60]:
```
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 dense_6 (Dense)             (None, 4)                 8196
```

```
=================================================================
Total params: 23,595,908
Trainable params: 8,196
Non-trainable params: 23,587,712
_________________________________________________________________
```

In [61]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [62]:
```
early = EarlyStopping(monitor = 'val_loss', patience=3)
```

In [63]:
```
history_1 = model.fit(training_ds, epochs=10, validation_data=validation_ds, callbacks=early)
```

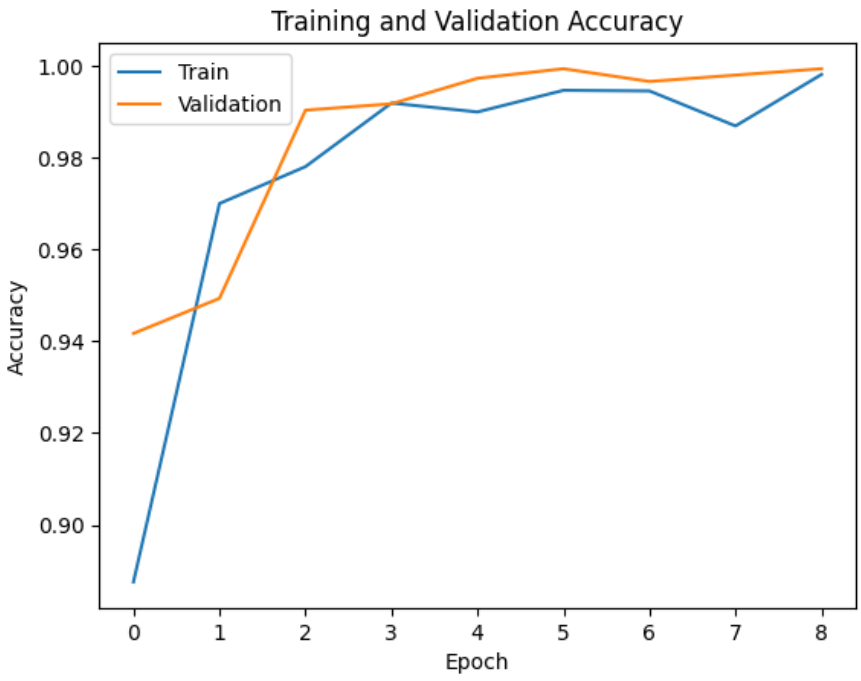
```
Epoch 1/10

226/226 [==============================] - 29s 115ms/step - loss: 0.5409 - accuracy: 0.8876 - val_loss: 0.2182 - val_accuracy: 0.9417
Epoch 2/10
226/226 [==============================] - 25s 108ms/step - loss: 0.1022 - accuracy: 0.9700 - val_loss: 0.2027 - val_accuracy: 0.9493
Epoch 3/10
226/226 [==============================] - 25s 109ms/step - loss: 0.0842 - accuracy: 0.9780 - val_loss: 0.0216 - val_accuracy: 0.9903
Epoch 4/10
226/226 [==============================] - 25s 108ms/step - loss: 0.0227 - accuracy: 0.9918 - val_loss: 0.0197 - val_accuracy: 0.9917
Epoch 5/10
226/226 [==============================] - 25s 108ms/step - loss: 0.0363 - accuracy: 0.9899 - val_loss: 0.0086 - val_accuracy: 0.9972
Epoch 6/10
226/226 [==============================] - 25s 109ms/step - loss: 0.0154 - accuracy: 0.9946 - val_loss: 0.0047 - val_accuracy: 0.9993
Epoch 7/10
226/226 [==============================] - 25s 108ms/step - loss: 0.0156 - accuracy: 0.9945 - val_loss: 0.0093 - val_accuracy: 0.9965
Epoch 8/10
226/226 [==============================] - 25s 109ms/step - loss: 0.0489 - accuracy: 0.9868 - val_loss: 0.0051 - val_accuracy: 0.9979
Epoch 9/10
226/226 [==============================] - 25s 110ms/step - loss: 0.0045 - accuracy: 0.9981 - val_loss: 0.0018 - val_accuracy: 0.9993
```

In [64]:
```
scores_1 = model.evaluate(testing_ds)
scores_1
```

```
57/57 [==============================] - 5s 92ms/step - loss: 0.0375 - accuracy: 0.9911
```

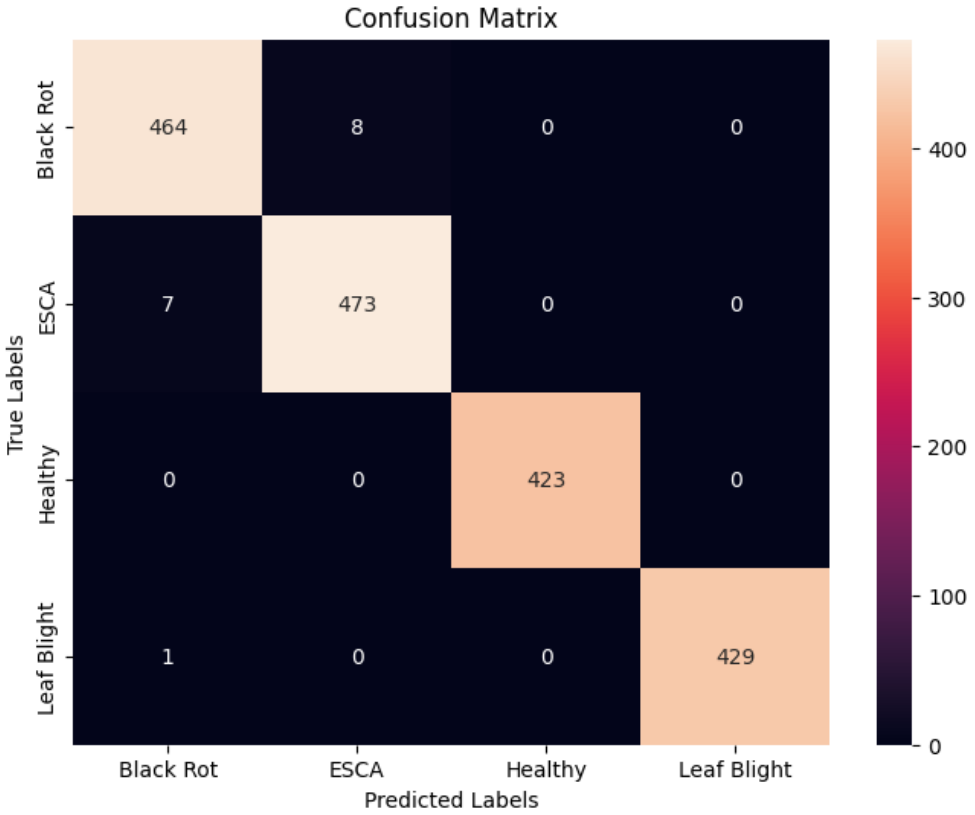
Out [64]:
```
[0.03753839060664177, 0.9911357164382935]
```

In [65]:
```
# Plot training and validation accuracy
plt.plot(history_1.history['accuracy'], label='Train')
plt.plot(history_1.history['val_accuracy'], label='Validation')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')
plt.show()
```



In [66]:
```
# Get the predicted labels for the test set
predicted_labels = []
true_labels = []

for images, labels in testing_ds:
    true_labels.extend(labels.numpy())
```

```
    predictions = model.predict(images, verbose=0)
    predicted_labels.extend(np.argmax(predictions, axis=1))

# Create a confusion matrix
conf_matrix = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [67]:
```
model.save(f"../Models/ResNet50")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 53). These
functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: ../Models/ResNet50\assets
```

```
INFO:tensorflow:Assets written to: ../Models/ResNet50\assets
```

In [33]:
```
from tensorflow.keras.applications import VGG16

model = models.Sequential()
model.add(VGG16(include_top=False, pooling='max', weights='imagenet', input_shape = (iz, iz, 3)))
model.add(Dense(class_number, activation='softmax'))
model.layers[0].trainable = False
```

In [34]:
```
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 512)               14714688

 dense_3 (Dense)             (None, 4)                 2052

=================================================================
Total params: 14,716,740
Trainable params: 2,052
```

```
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [35]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [36]:
```
early = EarlyStopping(monitor = 'val_loss', patience=3)
```

In [37]:
```
history_2 = model.fit(training_ds, epochs=10, validation_data=validation_ds, callbacks=early)
```

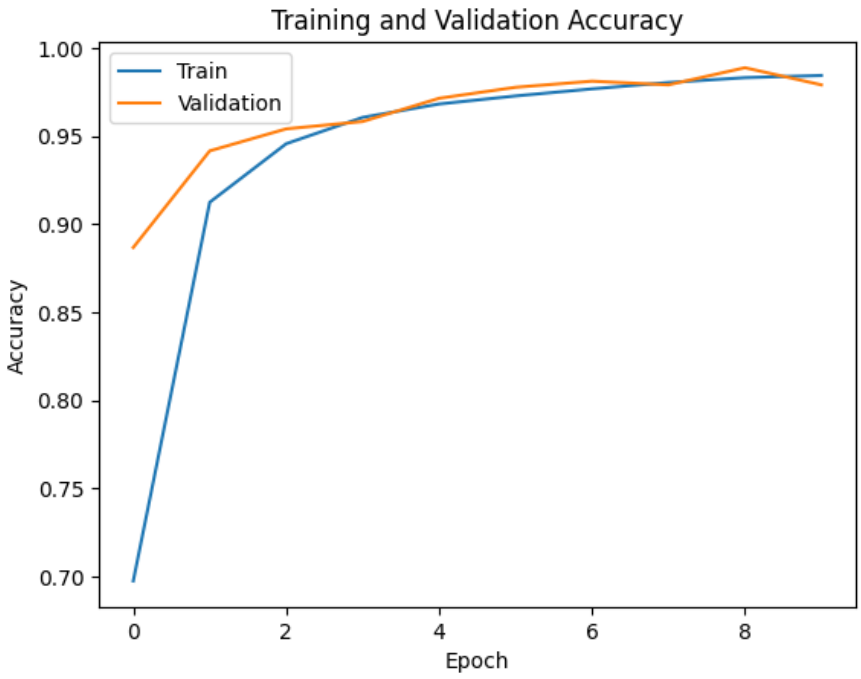
```
Epoch 1/10
226/226 [==============================] - 40s 156ms/step - loss: 7.4789 - accuracy: 0.6975 - val_loss: 1.7211 - val_accuracy: 0.8868
Epoch 2/10
226/226 [==============================] - 32s 141ms/step - loss: 1.1532 - accuracy: 0.9125 - val_loss: 0.7295 - val_accuracy: 0.9417
Epoch 3/10
226/226 [==============================] - 32s 142ms/step - loss: 0.6197 - accuracy: 0.9457 - val_loss: 0.5121 - val_accuracy: 0.9542
Epoch 4/10
226/226 [==============================] - 32s 141ms/step - loss: 0.3847 - accuracy: 0.9607 - val_loss: 0.3684 - val_accuracy: 0.9583
Epoch 5/10
226/226 [==============================] - 32s 141ms/step - loss: 0.3059 - accuracy: 0.9683 - val_loss: 0.2306 - val_accuracy: 0.9715
Epoch 6/10
226/226 [==============================] - 32s 143ms/step - loss: 0.2353 - accuracy: 0.9729 - val_loss: 0.1520 - val_accuracy: 0.9778
Epoch 7/10
226/226 [==============================] - 32s 142ms/step - loss: 0.1605 - accuracy: 0.9769 - val_loss: 0.1378 - val_accuracy: 0.9812
Epoch 8/10
226/226 [==============================] - 32s 142ms/step - loss: 0.1315 - accuracy: 0.9805 - val_loss: 0.1209 - val_accuracy: 0.9792
Epoch 9/10
226/226 [==============================] - 33s 144ms/step - loss: 0.1009 - accuracy: 0.9832 - val_loss: 0.0827 - val_accuracy: 0.9889
Epoch 10/10
226/226 [==============================] - 32s 142ms/step - loss: 0.0909 - accuracy: 0.9845 - val_loss: 0.1494 - val_accuracy: 0.9792
```

In [38]:
```
scores_2 = model.evaluate(testing_ds)
scores_2
```

```
57/57 [==============================] - 9s 160ms/step - loss: 0.3489 - accuracy: 0.9573
```

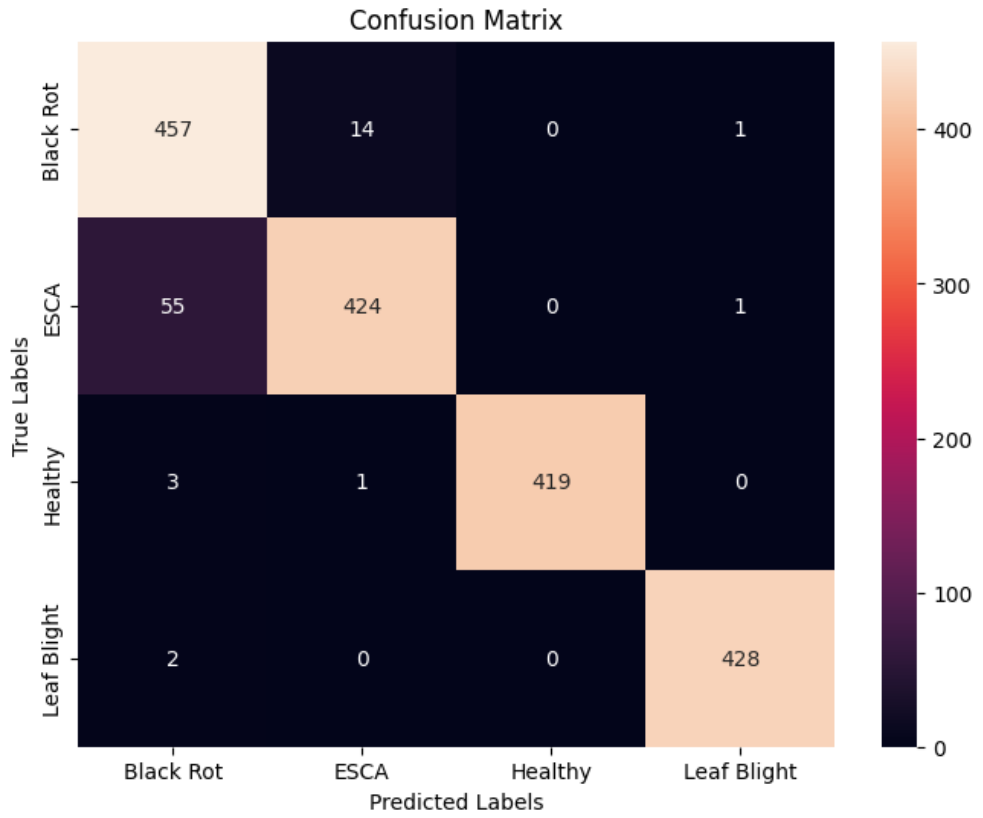
Out [38]:
```
[0.34887319803237915, 0.9573407173156738]
```

In [39]:
```
# Plot training and validation accuracy
plt.plot(history_2.history['accuracy'], label='Train')
plt.plot(history_2.history['val_accuracy'], label='Validation')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')
plt.show()
```



In [40]:
```
# Get the predicted labels for the test set
predicted_labels = []
true_labels = []

for images, labels in testing_ds:
    true_labels.extend(labels.numpy())
    predictions = model.predict(images, verbose=0)
```

```
    predicted_labels.extend(np.argmax(predictions, axis=1))

# Create a confusion matrix
conf_matrix = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [41]:
```
model.save(f"../Models/VGG16")
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 13). These
functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: ../Models/VGG16\assets
```

```
INFO:tensorflow:Assets written to: ../Models/VGG16\assets
```

In [ ]:
```
```

In [42]:
```
from tensorflow.keras.applications import EfficientNetV2B3

model = models.Sequential()
model.add(EfficientNetV2B3(include_top=False, pooling='max', weights='imagenet', input_shape = (iz, iz,
model.add(Dense(class_number, activation='softmax'))
model.layers[0].trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/efficientnet_v2/efficientnetv2-b3_notop.h5
52606240/52606240 [==============================] - 7s 0us/step
```

In [43]:
```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetv2-b3 (Function  (None, 1536)             12930622
 al)

 dense_4 (Dense)             (None, 4)                 6148
```

```
=================================================================
Total params: 12,936,770
Trainable params: 6,148
Non-trainable params: 12,930,622
_________________________________________________________________
```

In [44]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [45]:
```
early = EarlyStopping(monitor = 'val_loss', patience=3)
```

In [46]:
```
history_3 = model.fit(training_ds, epochs=10, validation_data=validation_ds, callbacks=early)
```

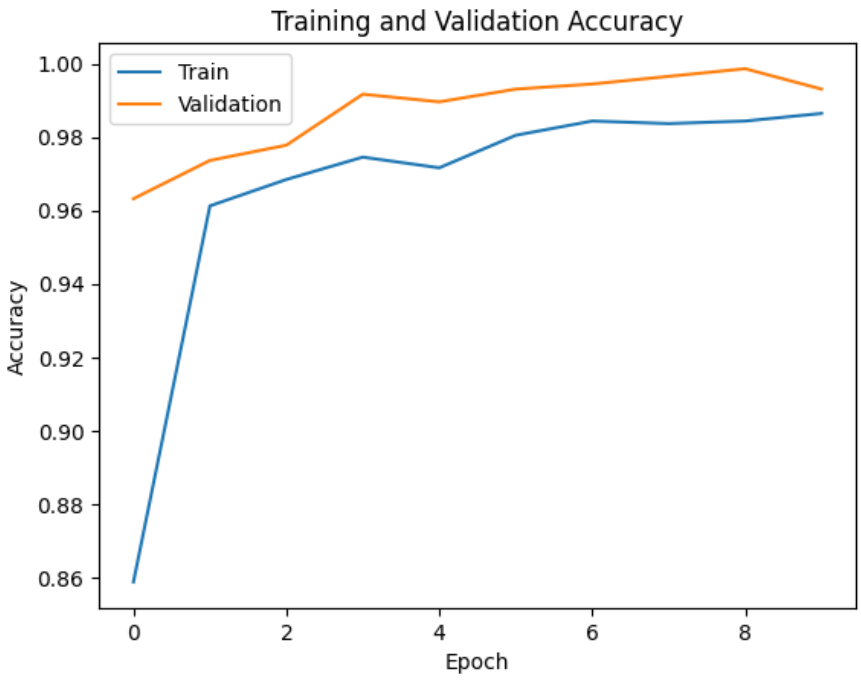
```
Epoch 1/10
226/226 [==============================] - 37s 120ms/step - loss: 0.4098 - accuracy: 0.8588 - val_loss: 0.0973 - val_accuracy: 0.9632
Epoch 2/10
226/226 [==============================] - 24s 108ms/step - loss: 0.1133 - accuracy: 0.9612 - val_loss: 0.0705 - val_accuracy: 0.9736
Epoch 3/10
226/226 [==============================] - 24s 107ms/step - loss: 0.0870 - accuracy: 0.9684 - val_loss: 0.0570 - val_accuracy: 0.9778
Epoch 4/10
226/226 [==============================] - 24s 107ms/step - loss: 0.0713 - accuracy: 0.9745 - val_loss: 0.0241 - val_accuracy: 0.9917
Epoch 5/10
226/226 [==============================] - 24s 106ms/step - loss: 0.0791 - accuracy: 0.9716 - val_loss: 0.0238 - val_accuracy: 0.9896
Epoch 6/10
226/226 [==============================] - 24s 106ms/step - loss: 0.0562 - accuracy: 0.9805 - val_loss: 0.0176 - val_accuracy: 0.9931
Epoch 7/10
226/226 [==============================] - 24s 106ms/step - loss: 0.0432 - accuracy: 0.9844 - val_loss: 0.0177 - val_accuracy: 0.9944
Epoch 8/10
226/226 [==============================] - 24s 105ms/step - loss: 0.0441 - accuracy: 0.9837 - val_loss: 0.0137 - val_accuracy: 0.9965
Epoch 9/10
226/226 [==============================] - 24s 106ms/step - loss: 0.0417 - accuracy: 0.9844 - val_loss: 0.0093 - val_accuracy: 0.9986
Epoch 10/10
226/226 [==============================] - 24s 108ms/step - loss: 0.0405 - accuracy: 0.9864 - val_loss: 0.0184 - val_accuracy: 0.9931
```

In [47]:
```
scores_3 = model.evaluate(testing_ds)
scores_3
```

```
57/57 [==============================] - 5s 90ms/step - loss: 0.0464 - accuracy: 0.9861
```

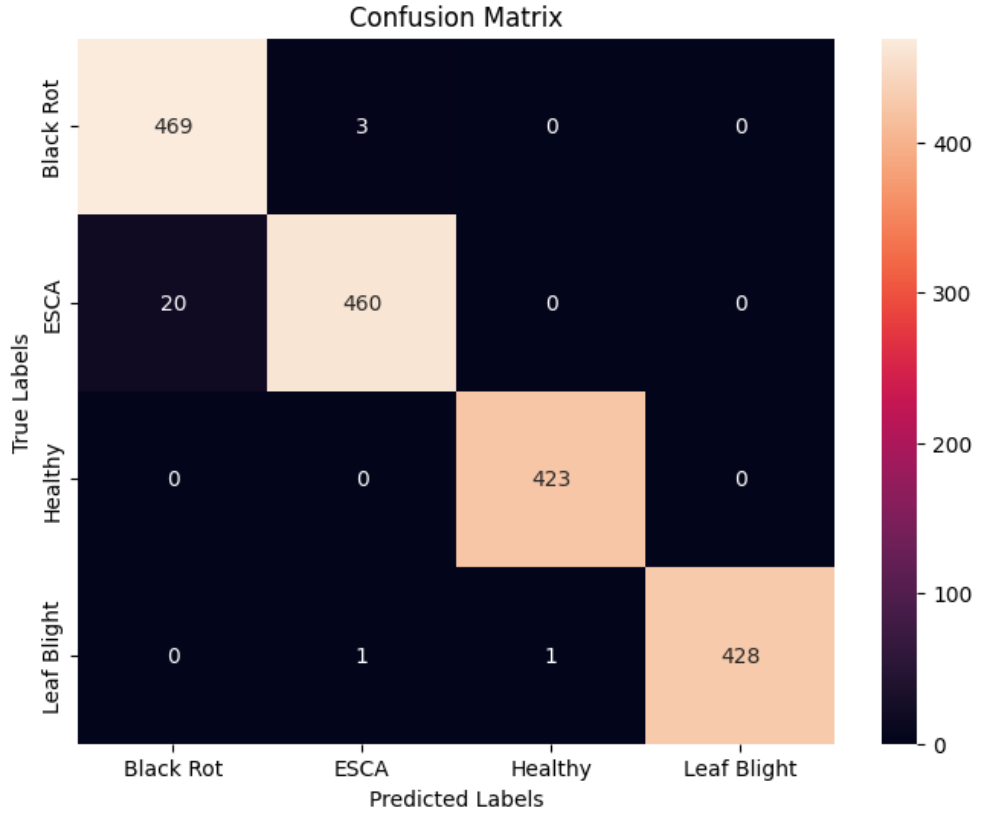
Out [47]:
```
[0.04637763649225235, 0.9861496090888977]
```

In [48]:
```
# Plot training and validation accuracy
plt.plot(history_3.history['accuracy'], label='Train')
plt.plot(history_3.history['val_accuracy'], label='Validation')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')
plt.show()
```



In [49]:
```
# Get the predicted labels for the test set
predicted_labels = []
true_labels = []

for images, labels in testing_ds:
```

```
    true_labels.extend(labels.numpy())
    predictions = model.predict(images, verbose=0)
    predicted_labels.extend(np.argmax(predictions, axis=1))

# Create a confusion matrix
conf_matrix = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix using Seaborn
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```



In [50]:
```
model.save(f"../Models/EfficientNetV2B3")
```

```
WARNING:absl:Function `_wrapped_model` contains input name(s) efficientnetv2-b3_input with unsupported characters which will be renamed to efficientnetv2_b3_input in the SavedModel.
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 136). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: ../Models/EfficientNetV2B3\assets

INFO:tensorflow:Assets written to: ../Models/EfficientNetV2B3\assets
```

In [ ]:
```
```