In [1]:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

In [2]:

```
# Specify the path to your zip file
zip_file_path = '/content/sample_data/LAB_09.zip'

# Specify the directory where you want to extract the contents
extracted_folder_path = '/content/sample_data/folder'

# Unzip the file
import zipfile
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_folder_path)
```

In [4]:

```
# Define constants
img_height, img_width = 224, 224
num_classes = 7
batch_size = 32

# Define data paths
train_data_dir = '/content/sample_data/folder/LAB_09/TRAIN'
test_data_dir = '/content/sample_data/folder/LAB_09/TEST'

# Data preprocessing and augmentation
train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

# Data generators
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical'
)
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

In [5]:

```
# Load pre-trained ResNet model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(img_height, im
g_width, 3))

# Freeze the layers of the pre-trained ResNet
for layer in base_model.layers:
    layer.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet
/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 0s 0us/step
```

In [6]:

```
# Build your classification model on top of the pre-trained ResNet
model = models.Sequential()
model.add(base_model)
model.add(layers.GlobalAveragePooling2D())
model.add(layers.Dense(7, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(num_classes, activation='softmax'))
```

In [7]:

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [8]:

```
# Train the model
epochs = 5  # Adjust the number of epochs as needed
history = model.fit(
    train_generator,
    epochs=epochs,
    validation_data=test_generator
)
```

```
Epoch 1/5
898/898 [==============================] - 401s 432ms/step - loss: 1.8797 - accuracy: 0.2
506 - val_loss: 1.8435 - val_accuracy: 0.2471
Epoch 2/5
898/898 [==============================] - 389s 433ms/step - loss: 1.8282 - accuracy: 0.2
513 - val_loss: 1.8233 - val_accuracy: 0.2471
Epoch 3/5
898/898 [==============================] - 390s 434ms/step - loss: 1.8160 - accuracy: 0.2
513 - val_loss: 1.8166 - val_accuracy: 0.2471
Epoch 4/5
898/898 [==============================] - 387s 431ms/step - loss: 1.8119 - accuracy: 0.2
513 - val_loss: 1.8142 - val_accuracy: 0.2471
Epoch 5/5
898/898 [==============================] - 386s 429ms/step - loss: 1.8104 - accuracy: 0.2
513 - val_loss: 1.8134 - val_accuracy: 0.2471
```

In [9]:

```
# Evaluate the model on the test set
accuracy = model.evaluate(test_generator)[1]
print('Test Accuracy: {:.2%}'.format(accuracy))
```

```
225/225 [==============================] - 25s 109ms/step - loss: 1.8134 - accuracy: 0.24
71
Test Accuracy: 24.71%
```

In [15]:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image

image_index = 0

test_image_path = test_generator.filepaths[image_index]
img = image.load_img(test_image_path, target_size=(img_height, img_width))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.0

predictions = model.predict(img_array)
```

```
predicted_label = test_generator.classes[image_index]
predicted_class_name = list(test_generator.class_indices.keys())[predicted_label]

true_label = int(test_generator.classes[image_index])
true_class_name = list(test_generator.class_indices.keys())[true_label]

plt.imshow(img)
plt.axis('off')

print("True Label:", true_class_name)
print("Predicted Label:", predicted_class_name)

# Show the plot
plt.show()
```

```
1/1 [==============================] - 0s 67ms/step
True Label: angry
Predicted Label: angry
```

In [ ]: