In [15]:
```
from keras.applications.vgg16 import VGG16
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input, decode_predictions
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
model = VGG16(weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```

In [3]:
```
model.summary()
```

Model: "vgg16"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102,764,544
fc2 (Dense)	(None, 4096)	16,781,312
predictions (Dense)	(None, 1000)	4,097,000

Total params: 138,357,544 (527.79 MB)

Trainable params: 138,357,544 (527.79 MB)

Non-trainable params: 0 (0.00 B)

In [6]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [13]:
```
img_path = '/content/drive/My Drive/Dataset/image_test.jpg'
img = load_img(img_path, target_size=(224, 224))
plt.imshow(img)
plt.axis('off')  # Optional: Hide axes
plt.show()
```

In [20]:
```
x = img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
x.shape
```

Out[20]: (1, 224, 224, 3)

In [21]:
```
# normalization
x = x.preprocess_input
preds = model.predict(x)
preds.shape
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
```

Out[21]: (1, 1000)

In [24]:
```
preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step
Predicted: [('n02504458', 'African_elephant', np.float32(0.90489537)), ('n01871265', 'tusker', np.float32(0.0672
014)), ('n02504013', 'Indian_elephant', np.float32(0.014559235))]
```