In [33]:
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D, Conv3D, BatchNormalization
from keras import backend as K
import os
from PIL import Image
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
from random import randrange
```

In [2]:
```
enc = OneHotEncoder()
enc.fit([[0], [1]])
def names(number):
    if(number == 0):
        return 'Dog'
    else:
        return 'Cat'
```

```
C:\Users\Vee\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:41
5: FutureWarning: The handling of integer data will change in version 0.22. Cur
rently, the categories are determined based on the range [0, max(values)], whil
e in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "cat
egories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the catego
ries to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [3]:
```
data = []
paths = []
ans = []
for r, d, f in os.walk(r"D:\Downloads\train\train\cat"):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    x = img.resize((128,128))
    data.append(np.array(x))
    ans.append(enc.transform([[1]]).toarray())
```

In [4]:
```
paths = []
for r, d, f in os.walk(r"D:\Downloads\train\train\dog"):
    for file in f:
        if '.jpg' in file:
            paths.append(os.path.join(r, file))

for path in paths:
    img = Image.open(path)
    x = img.resize((128,128))
    data.append(np.array(x))
    ans.append(enc.transform([[0]]).toarray())
```

In [5]:
```
data = np.array(data)
data.shape
```

Out[5]:
```
(25000, 128, 128, 3)
```

In [6]:
```
ans = np.array(ans)
ans = ans.reshape(25000,2)
```

In [7]:
```
#splitting data into train and test sets. 3/4 train, 1/4 test.
x_train,x_test,y_train,y_test = train_test_split(data, ans, test_size=0.2, shuff
```

In [8]:
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), input_shape=(128, 128, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(2, activation='softmax'))
model.compile(loss='mean_squared_error', optimizer='adam')
print(model.summary())
```

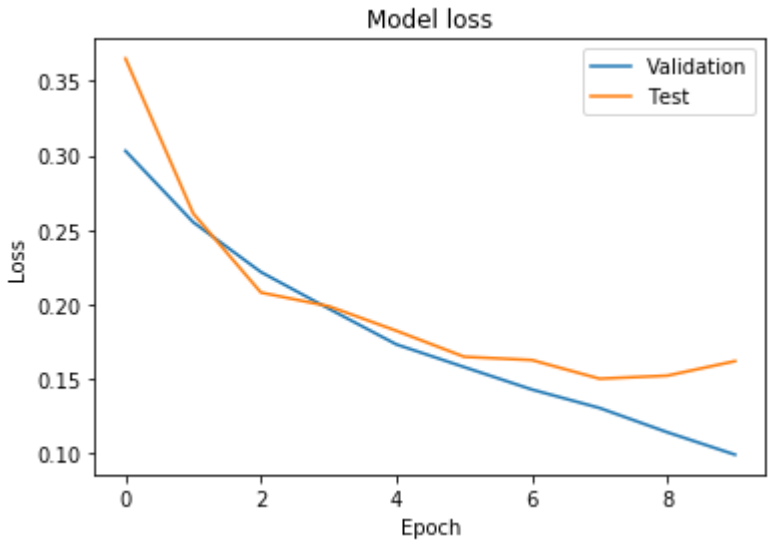
```
WARNING:tensorflow:From C:\Users\Vee\Anaconda3\lib\site-packages\keras\backend
\tensorflow_backend.py:4070: The name tf.nn.max_pool is deprecated. Please use
tf.nn.max_pool2d instead.

Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_1 (Batch (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 61, 61, 32)        9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 61, 61, 32)        128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 30, 30, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 28800)             0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               7373056
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dropout_5 (Dropout)          (None, 256)               0
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 514       
=================================================================
Total params: 7,450,786
Trainable params: 7,450,146
Non-trainable params: 640
_________________________________________________________________
None
```

In [9]:
```
history = model.fit(x_train, y_train, epochs=10, batch_size=500, verbose=1,valida
```

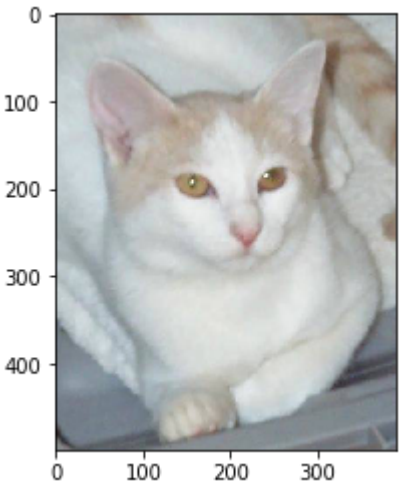
```
WARNING:tensorflow:From C:\Users\Vee\Anaconda3\lib\site-packages\keras\backend
\tensorflow_backend.py:422: The name tf.global_variables is deprecated. Please
use tf.compat.v1.global_variables instead.

Train on 20000 samples, validate on 5000 samples
Epoch 1/10
20000/20000 [==============================] - 13s 667us/step - loss: 0.3030 -
val_loss: 0.3650
Epoch 2/10
20000/20000 [==============================] - 10s 511us/step - loss: 0.2551 -
val_loss: 0.2610
Epoch 3/10
20000/20000 [==============================] - 10s 512us/step - loss: 0.2216 -
val_loss: 0.2079
Epoch 4/10
20000/20000 [==============================] - 10s 523us/step - loss: 0.1971 -
val_loss: 0.1987
Epoch 5/10
20000/20000 [==============================] - 11s 535us/step - loss: 0.1730 -
val_loss: 0.1822
Epoch 6/10
20000/20000 [==============================] - 11s 526us/step - loss: 0.1578 -
val_loss: 0.1647
Epoch 7/10
20000/20000 [==============================] - 10s 522us/step - loss: 0.1428 -
val_loss: 0.1625
Epoch 8/10
20000/20000 [==============================] - 10s 520us/step - loss: 0.1303 -
val_loss: 0.1500
Epoch 9/10
20000/20000 [==============================] - 10s 519us/step - loss: 0.1141 -
val_loss: 0.1521
Epoch 10/10
20000/20000 [==============================] - 10s 522us/step - loss: 0.0990 -
val_loss: 0.1618
```

In [10]:
```
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Validation', 'Test'], loc='upper right')
plt.show()
```



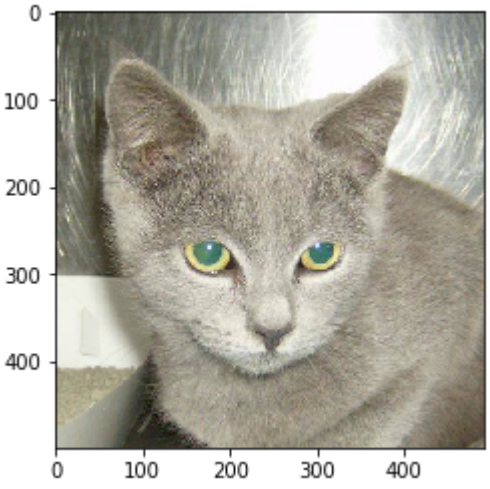
In [55]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
99.99980926513672% Confidence This Is A Cat
```



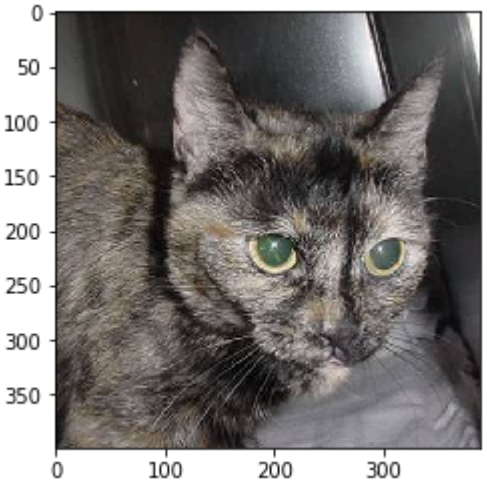
In [56]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
99.82241988182068% Confidence This Is A Cat
```



In [57]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
99.31902289390564% Confidence This Is A Cat
```

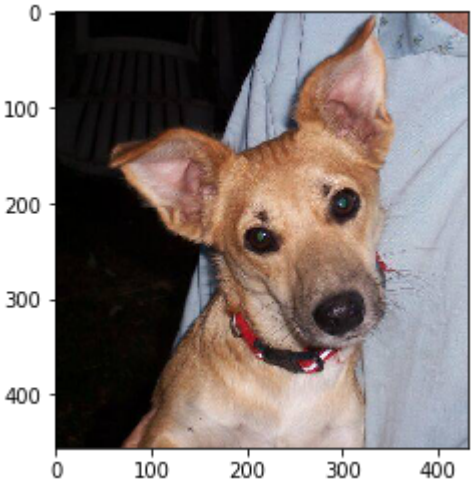


In [58]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
97.91345596313477% Confidence This Is A Cat
```

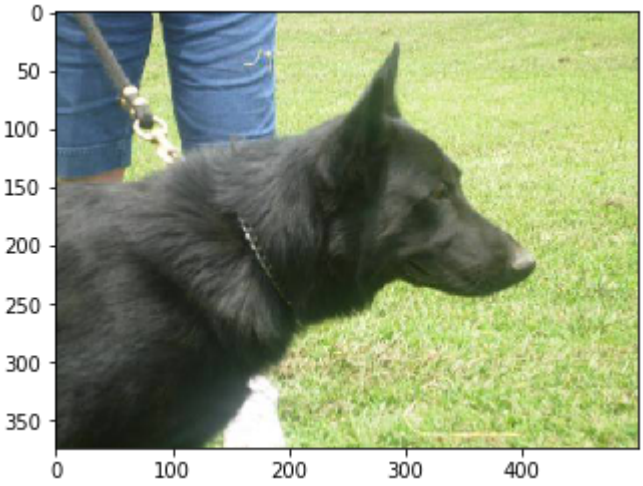
In [60]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
79.34935092926025% Confidence This Is A Dog
```



In [61]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
95.79391479492188% Confidence This Is A Dog
```



In [62]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

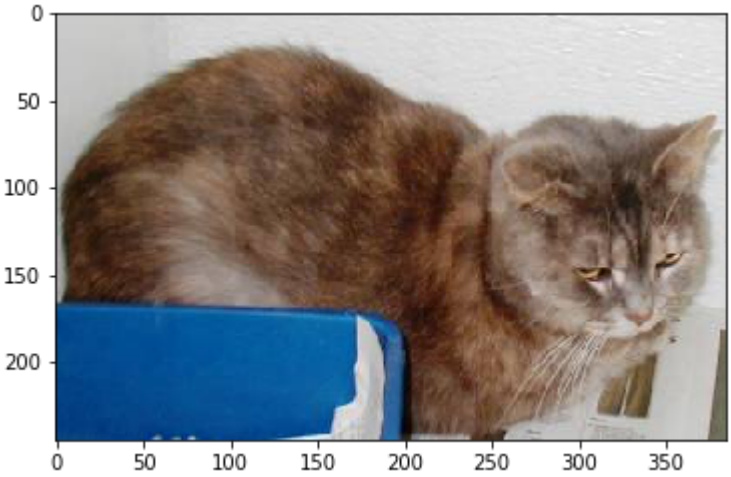
```
99.96683597564697% Confidence This Is A Cat
```

In [65]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
73.27191829681396% Confidence This Is A Dog
```

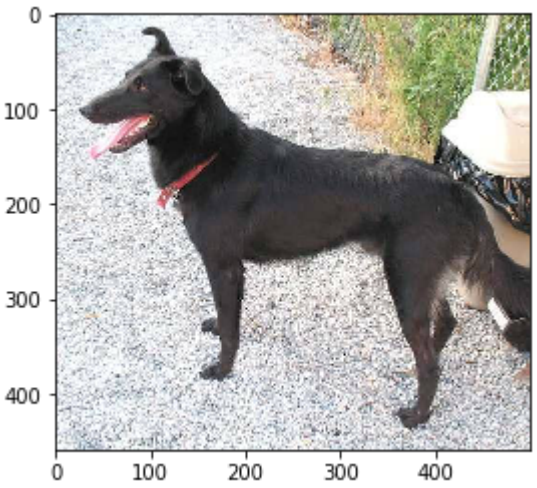
In [66]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
66.52367115020752% Confidence This Is A Cat
```



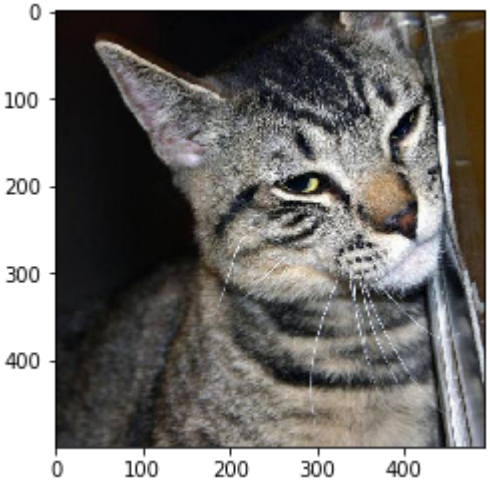
In [68]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
99.98359680175781% Confidence This Is A Dog
```



In [69]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
99.55670237541199% Confidence This Is A Cat
```



In [71]:
```
randy = randrange(12500)
path = os.path.join(r'D:\Downloads\test1\test1', str(randy)+'.jpg')
img = Image.open(path)
x = np.array(img.resize((128,128)))
x = x.reshape(1,128,128,3)
answ = model.predict_on_batch(x)
classification = np.where(answ == np.amax(answ))[1][0]
imshow(img)
print(str(answ[0][classification]*100) + '% Confidence This Is A ' + names(class
```

```
99.67215657234192% Confidence This Is A Dog
```

In [ ]:
```
```