In [2]:

```
import numpy as np
import random
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten
```

In [3]:

```
X_train = np.loadtxt('C:\\Users\\k200237\\Desktop\\input.csv', delimiter = ',')
Y_train = np.loadtxt('C:\\Users\\k200237\\Desktop\\labels.csv', delimiter = ',')

X_test = np.loadtxt('C:\\Users\\k200237\\Desktop\\input_test.csv', delimiter = ',')
Y_test = np.loadtxt('C:\\Users\\k200237\\Desktop\\labels_test.csv', delimiter = ',')
```

In [4]:

```
X_train = X_train.reshape(len(X_train), 100, 100, 3)
Y_train = Y_train.reshape(len(Y_train), 1)

X_test = X_test.reshape(len(X_test), 100, 100, 3)
Y_test = Y_test.reshape(len(Y_test), 1)

X_train = X_train/255.0
X_test = X_test/255.0
```

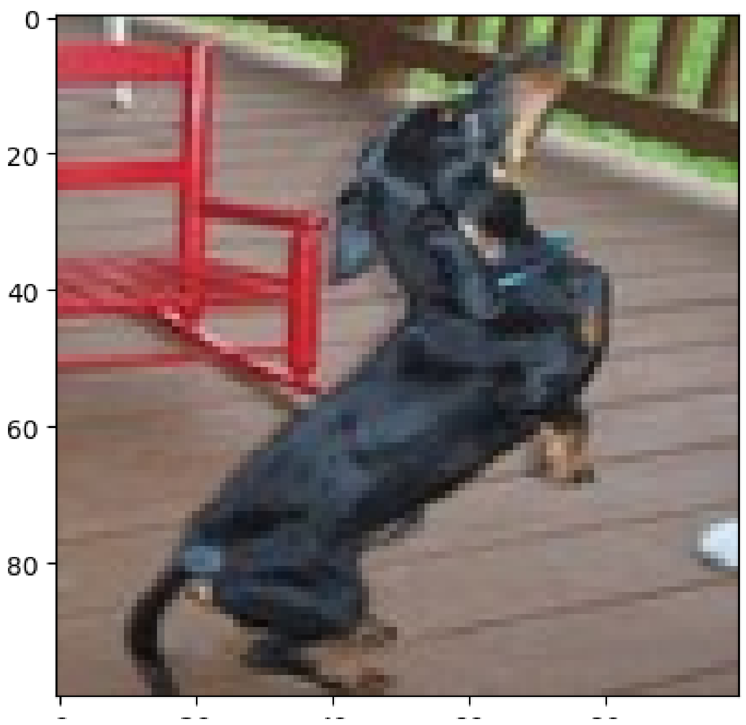
In [5]:

```
print("Shape of X_train: ", X_train.shape)
print("Shape of Y_train: ", Y_train.shape)
print("Shape of X_test: ", X_test.shape)
print("Shape of Y_test: ", Y_test.shape)
```

```
Shape of X_train:  (2000, 100, 100, 3)
Shape of Y_train:  (2000, 1)
Shape of X_test:  (400, 100, 100, 3)
Shape of Y_test:  (400, 1)
```

In [6]:

```
idx = random.randint(0, len(X_train))
plt.imshow(X_train[idx, :])
plt.show()
```



In [7]:

```
model = Sequential([
    Conv2D(32, (3,3), activation = 'relu', input_shape = (100, 100, 3)),
    MaxPooling2D((2,2)),

    Conv2D(32, (3,3), activation = 'relu'),
    MaxPooling2D((2,2)),

    Flatten(),
    Dense(64, activation = 'relu'),
    Dense(1, activation = 'sigmoid')
])
```

In [8]:

```
model = Sequential()

model.add(Conv2D(32, (3,3), activation = 'relu', input_shape = (100, 100, 3)))
model.add(MaxPooling2D((2,2)))

model.add(Conv2D(32, (3,3), activation = 'relu'))
model.add(MaxPooling2D((2,2)))

model.add(Flatten())
model.add(Dense(64, activation = 'relu'))
model.add(Dense(1, activation = 'sigmoid'))
```

In [9]:

```
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

In [10]:

```
model.fit(X_train, Y_train, epochs = 10, batch_size = 64)
```

```
Epoch 1/10
32/32 [==============================] - 9s 245ms/step - loss: 0.7428 - accuracy: 0.5100
Epoch 2/10
32/32 [==============================] - 9s 269ms/step - loss: 0.6755 - accuracy: 0.5855
Epoch 3/10
32/32 [==============================] - 9s 270ms/step - loss: 0.6451 - accuracy: 0.6285
Epoch 4/10
32/32 [==============================] - 9s 271ms/step - loss: 0.5979 - accuracy: 0.6900
Epoch 5/10
32/32 [==============================] - 9s 275ms/step - loss: 0.5735 - accuracy: 0.7050
Epoch 6/10
32/32 [==============================] - 9s 271ms/step - loss: 0.5297 - accuracy: 0.7320
Epoch 7/10
32/32 [==============================] - 9s 273ms/step - loss: 0.5001 - accuracy: 0.7730
Epoch 8/10
32/32 [==============================] - 9s 272ms/step - loss: 0.4717 - accuracy: 0.7720
Epoch 9/10
32/32 [==============================] - 9s 276ms/step - loss: 0.4273 - accuracy: 0.8070
Epoch 10/10
32/32 [==============================] - 9s 286ms/step - loss: 0.3896 - accuracy: 0.8240
```

Out[10]:

```
<keras.src.callbacks.History at 0x2a25a1ae710>
```

In [11]:

```
model.evaluate(X_test, Y_test)
```

```
13/13 [==============================] - 0s 26ms/step - loss: 0.6122 - accuracy: 0.6900
```

Out[11]:

```
[0.6122192740440369, 0.6899999976158142]
```

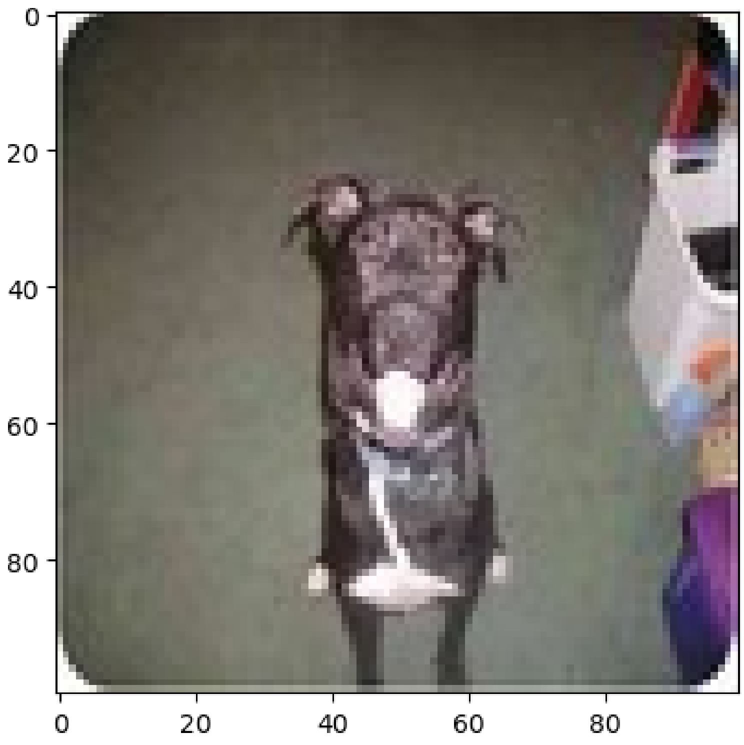
In [12]:

```
idx2 = random.randint(0, len(Y_test))
plt.imshow(X_test[idx2, :])
plt.show()

y_pred = model.predict(X_test[idx2, :].reshape(1, 100, 100, 3))
y_pred = y_pred > 0.5

if(y_pred == 0):
    pred = 'dog'
else:
    pred = 'cat'

print("Our model says it is a :", pred)
```



```
1/1 [==============================] - 0s 60ms/step
Our model says it is a : dog
```