# LAB CYCLE 8

**Experiment No:8**

**Date : 26/03/2022**

**Aim:**

**Image classification using keras framework**

## Source Code:

In [16]:
```
#8.IMAGE CLASSIFICATION USING KERAS FRAMEWORK
```

In [17]:
```
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
```

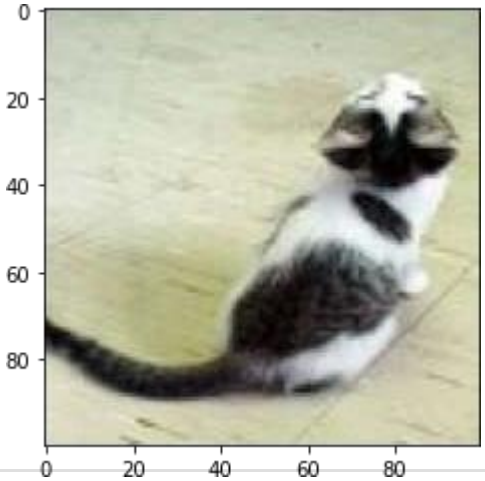
In [18]:
```
X_train = np.loadtxt('input.csv', delimiter = ',')
Y_train = np.loadtxt('labels.csv', delimiter = ',')

X_test = np.loadtxt('input_test.csv', delimiter = ',')
Y_test = np.loadtxt('labels_test.csv', delimiter = ',')
```

In [19]:
```
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 [20]:
```
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 [21]:
```
idx = random.randint(0, len(X_train))
plt.imshow(X_train[idx, :])
plt.show()
```




In [22]:
```
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 [23]:
```
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 [24]:
```
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy
```

In [25]:
```
model.fit(X_train, Y_train, epochs = 5, batch_size = 64)
```

```
Epoch 1/5
32/32 [==============================] - 10s 278ms/step - loss: 0.6973 - accuracy:
0.5270
Epoch 2/5
32/32 [==============================]  - 9s 285ms/step - loss: 0.6588 - accuracy: 0.
6185
Epoch 3/5
32/32 [==============================]  - 9s 286ms/step - loss: 0.6043 - accuracy: 0.
6785
Epoch 4/5
32/32 [==============================]  - 9s 287ms/step - loss: 0.5562 - accuracy: 0.
7250
Epoch 5/5
32/32 [==============================]  - 9s 285ms/step - loss: 0.5064 - accuracy: 0.
7610
```

Out[25]:
```
<keras.callbacks.History at 0x24923e9aa90>
```

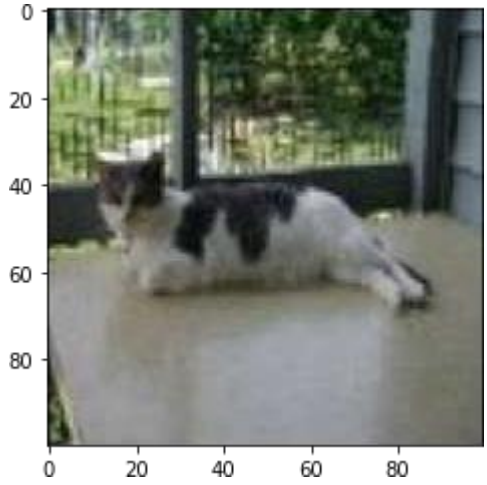
In [26]:
```
model.evaluate(X_test, Y_test)
```

```
13/13 [==============================] - 1s 42ms/step - loss: 0.6152 - accuracy: 0.6
825
```

Out[26]:
```
[0.6151993274688721, 0.6825000047683716]
```

In [27]:
```
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
```

```
    pred = 'dog'
else:
    pred = 'cat'

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




```
Our model says it is a : cat
```