In [1]:

```
#importing necessary libraries
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
from tensorflow.keras.datasets import cifar10
%matplotlib inline
```

In [2]:

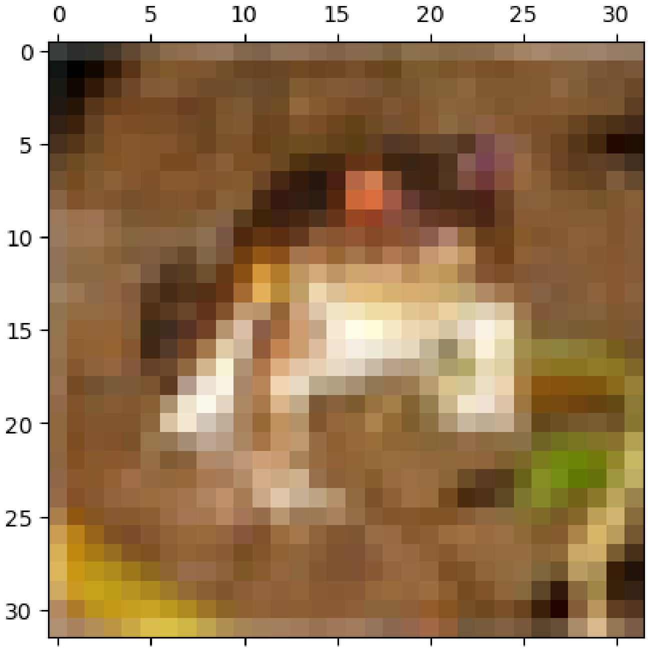
```
#import dataset and split into train and test data
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

In [3]:

```
plt.matshow(x_train[0])
```

Out[3]:

```
<matplotlib.image.AxesImage at 0x146c3b62bc8>
```

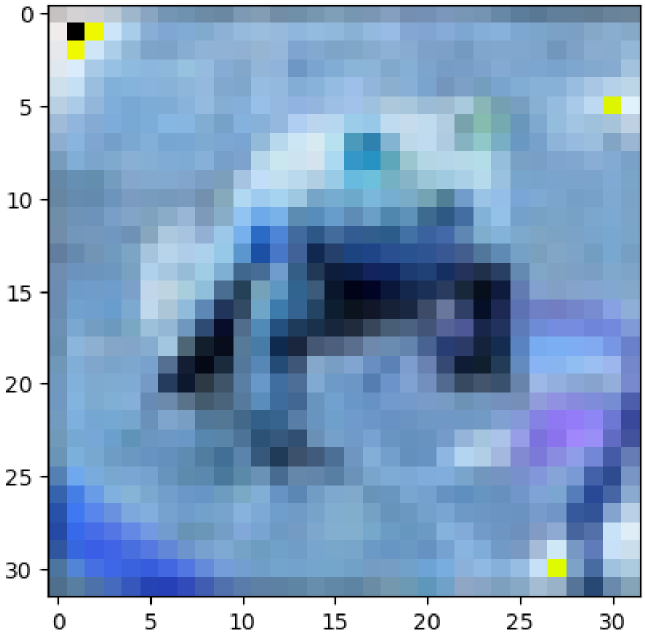


In [4]:

```
plt.imshow(-x_train[0], cmap="gray")
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x146c3856848>
```



In [5]:

```
x_train = x_train / 255
x_test = x_test / 255
```

In [6]:

```
model = keras.Sequential([
keras.layers.Flatten(input_shape=(32, 32,3)),
keras.layers.Dense(128, activation="relu"),
keras.layers.Dense(10, activation="softmax")
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 3072)              0         
_________________________________________________________________
dense (Dense)                (None, 128)               393344    
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290      
=================================================================
Total params: 394,634
Trainable params: 394,634
Non-trainable params: 0
_________________________________________________________________
```

In [7]:

```
model.compile(optimizer="sgd",
loss="sparse_categorical_crossentropy",
metrics=['accuracy'])
```

In [8]:

```
history=model.fit(x_train,
y_train,validation_data=(x_test,y_test),epochs=10)
```

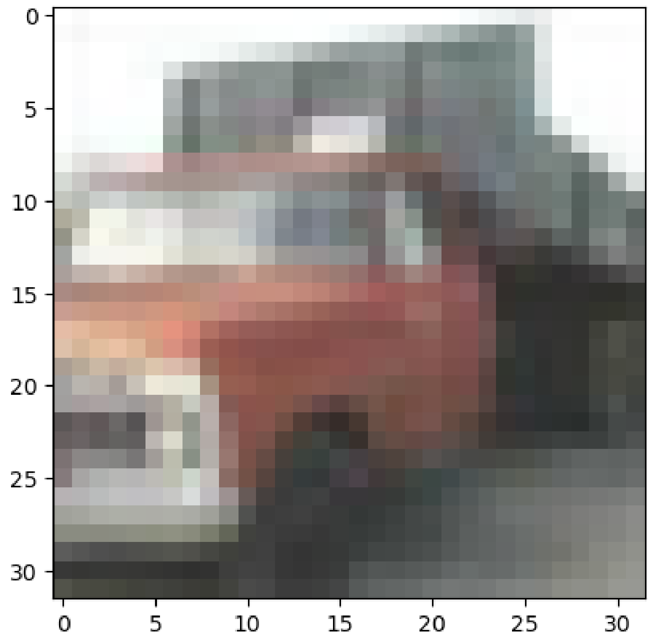
```
Epoch 1/10
1546/1563 [============================>.] - ETA: 0s - loss: 1.8893 - accuracy: 0.3231WARNING:tensorflow:Callbac
ks method `on_test_batch_begin` is slow compared to the batch time (batch time: 0.0000s vs `on_test_batch_begin`
time: 0.0010s). Check your callbacks.
1563/1563 [==============================] - 2s 1ms/step - loss: 1.8883 - accuracy: 0.3237 - val_loss: 1.7685 -
val_accuracy: 0.3721
Epoch 2/10
1563/1563 [==============================] - 2s 1ms/step - loss: 1.7184 - accuracy: 0.3934 - val_loss: 1.6731 -
val_accuracy: 0.4170
Epoch 3/10
1563/1563 [==============================] - 2s 1ms/step - loss: 1.6438 - accuracy: 0.4212 - val_loss: 1.6656 -
val_accuracy: 0.4046
Epoch 4/10
1563/1563 [==============================] - 2s 1ms/step - loss: 1.5979 - accuracy: 0.4377 - val_loss: 1.6545 -
val_accuracy: 0.4071
Epoch 5/10
1563/1563 [==============================] - 2s 1ms/step - loss: 1.5597 - accuracy: 0.4532 - val_loss: 1.5598 -
val_accuracy: 0.4424
Epoch 6/10
1563/1563 [==============================] - 2s 1ms/step - loss: 1.5291 - accuracy: 0.4630 - val_loss: 1.5596 -
val_accuracy: 0.4515
Epoch 7/10
1563/1563 [==============================] - 2s 1ms/step - loss: 1.5007 - accuracy: 0.4739 - val_loss: 1.5019 -
val_accuracy: 0.4679
Epoch 8/10
1563/1563 [==============================] - 3s 2ms/step - loss: 1.4782 - accuracy: 0.4820 - val_loss: 1.5338 -
val_accuracy: 0.4517
Epoch 9/10
1563/1563 [==============================] - 2s 2ms/step - loss: 1.4550 - accuracy: 0.4889 - val_loss: 1.5088 -
val_accuracy: 0.4566
Epoch 10/10
1563/1563 [==============================] - 2s 1ms/step - loss: 1.4377 - accuracy: 0.4961 - val_loss: 1.5045 -
val_accuracy: 0.4647
```

In [9]:

```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

```
313/313 [==============================] - 0s 573us/step - loss: 1.5045 - accuracy: 0.4647
Loss=1.504
Accuracy=0.465
```

In [10]:
```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```



In [11]:
```
x_train
```

Out[11]:
```
array([[[[0.23137255, 0.24313725, 0.24705882],
         [0.16862745, 0.18039216, 0.17647059],
         [0.19607843, 0.18823529, 0.16862745],
         ...,
         [0.61960784, 0.51764706, 0.42352941],
         [0.59607843, 0.49019608, 0.4       ],
         [0.58039216, 0.48627451, 0.40392157]],

        [[0.0627451 , 0.07843137, 0.07843137],
         [0.        , 0.        , 0.        ],
         [0.07058824, 0.03137255, 0.        ],
         ...,
         [0.48235294, 0.34509804, 0.21568627],
         [0.46666667, 0.3254902 , 0.19607843],
         [0.47843137, 0.34117647, 0.22352941]],

        [[0.09803922, 0.09411765, 0.08235294],
         [0.0627451 , 0.02745098, 0.        ],
         [0.19215686, 0.10588235, 0.03137255],
```

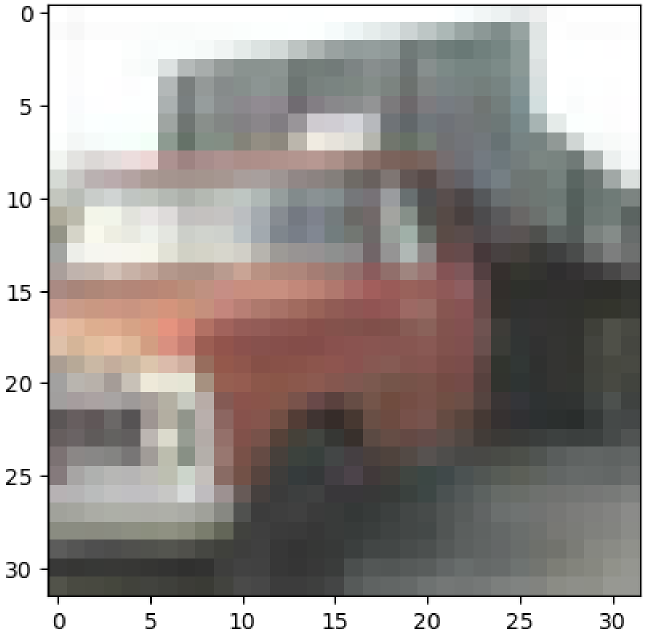
In [12]:
```
predicted_val = model.predict(x_test)
print(predicted_val[n])
maximum = -1
index = -1
for i in (0,9):
    if maximum<predicted_val[n][i]:
        maximum = predicted_val[n][i]
        index = i

print(index)
```

```
[0.02580737 0.03731725 0.03147932 0.0797637  0.04992767 0.05231224
 0.01397747 0.3509212  0.07246367 0.28603008]
9
```

```
In [13]: x_test
```

```
Out[13]: array([[[[0.61960784, 0.43921569, 0.19215686],
          [0.62352941, 0.43529412, 0.18431373],
          [0.64705882, 0.45490196, 0.2       ],
          ...,
          [0.5372549 , 0.37254902, 0.14117647],
          [0.49411765, 0.35686275, 0.14117647],
          [0.45490196, 0.33333333, 0.12941176]],

         [[0.59607843, 0.43921569, 0.2       ],
          [0.59215686, 0.43137255, 0.15686275],
          [0.62352941, 0.44705882, 0.17647059],
          ...,
          [0.53333333, 0.37254902, 0.12156863],
          [0.49019608, 0.35686275, 0.1254902 ],
          [0.46666667, 0.34509804, 0.13333333]],

         [[0.59215686, 0.43137255, 0.18431373],
          [0.59215686, 0.42745098, 0.12941176],
          [0.61960784, 0.43529412, 0.14117647],
```

```
In [14]: 
plt.imshow(x_test[n])
plt.show()

print(predicted_val[n])
```



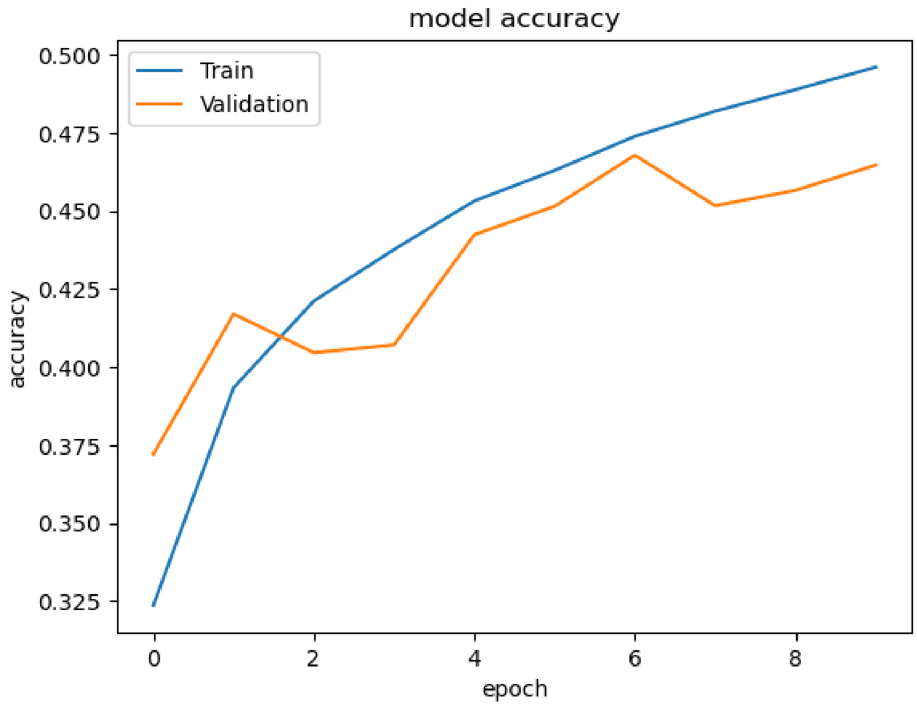
```
[0.02580737 0.03731725 0.03147932 0.0797637  0.04992767 0.05231224
 0.01397747 0.3509212  0.07246367 0.28603008]
```

```
In [15]: if index == 0:
    print("airplane")
if index == 1:
    print("automobile")
if index == 2:
    print("bird")
if index == 3:
    print("cat")
if index == 4:
    print("deer")
if index == 5:
    print("dog")
if index == 6:
    print("frog")
if index == 7:
    print("horse")
if index == 8:
    print("ship")
if index == 9:
    print("truck")
```

```
truck
```

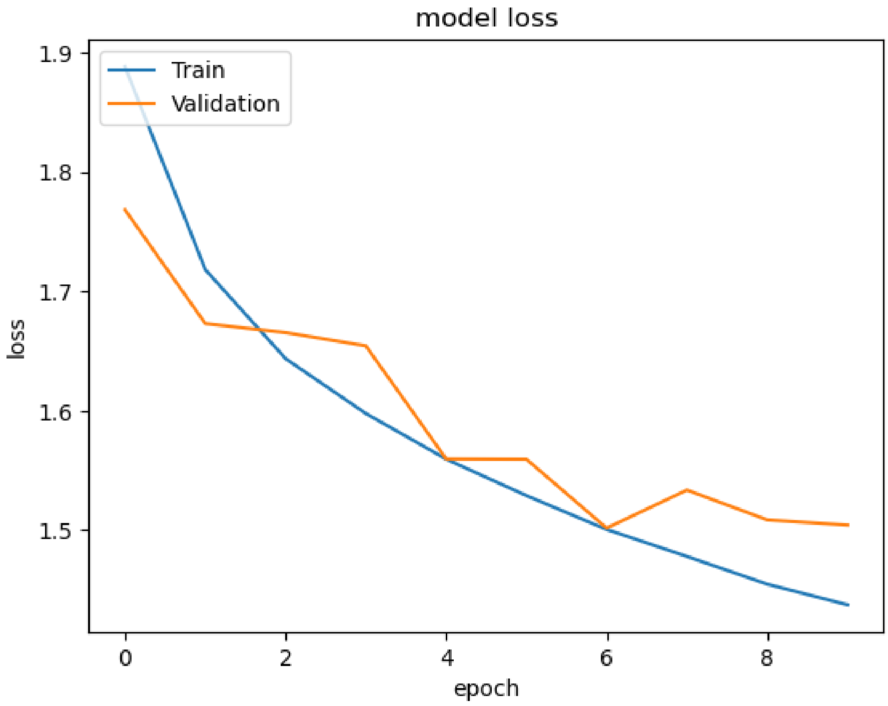
In [16]:

```python
# history.history()
history.history.keys()
# dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



In [17]:

```python
# history.history()
history.history.keys()
# dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```


model loss
loss
epoch
Train
Validation