In [1]:
```
import numpy as np # forlinear algebra
import matplotlib.pyplot as plt #for plotting things
import os
from PIL import Image
#print(os.listdir("../input"))

# Keras Libraries
import keras
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.preprocessing.image import ImageDataGenerator, load_img
from sklearn.metrics import classification_report, confusion_matrix
```

In [2]:
```
mainDIR = os.listdir('chest_xray/chest_xray')
print(mainDIR)
```

```
['.DS_Store', 'test', 'train', 'val']
```

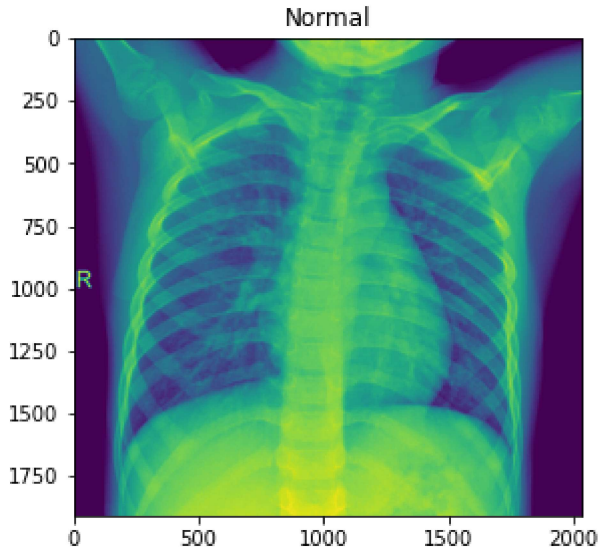
In [3]:
```
train_folder= 'chest_xray/chest_xray/train/'
val_folder = 'chest_xray/chest_xray/val/'
test_folder = 'chest_xray/chest_xray/test/'
```

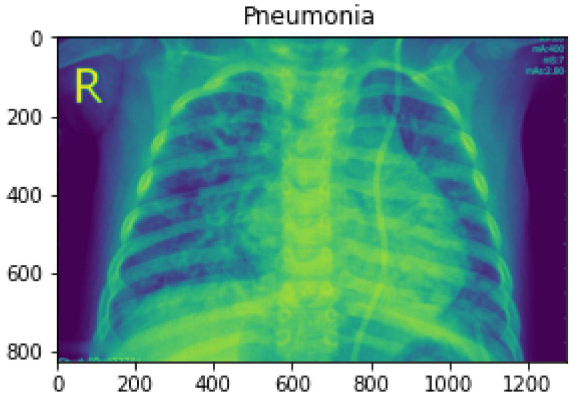
In [4]:
```
# train
os.listdir(train_folder)
train_n = train_folder+'NORMAL/'
train_p = train_folder+'PNEUMONIA/'
```

In [5]:
```
print(len(os.listdir(train_n)))
rand_norm= np.random.randint(0,len(os.listdir(train_n)))
norm_pic = os.listdir(train_n)[rand_norm]
print('normal picture title: ',norm_pic)

norm_pic_address = train_n+norm_pic

#Pneumonia
rand_p = np.random.randint(0,len(os.listdir(train_p)))

sic_pic =  os.listdir(train_p)[rand_norm]
sic_address = train_p+sic_pic
print('pneumonia picture title:', sic_pic)

# Load the images
norm_load = Image.open(norm_pic_address)
sic_load = Image.open(sic_address)

#Let's plt these images
f = plt.figure(figsize= (10,6))
a1 = f.add_subplot(1,2,1)
img_plot = plt.imshow(norm_load)
a1.set_title('Normal')

a2 = f.add_subplot(1, 2, 2)
img_plot = plt.imshow(sic_load)
a2.set_title('Pneumonia')
```

```
1342
normal picture title:  NORMAL2-IM-0397-0001.jpeg
pneumonia picture title: person1310_bacteria_3304.jpeg
```

Out[5]: Text(0.5, 1.0, 'Pneumonia')





In [6]:
```
cnn = Sequential()

#Convolution
cnn.add(Conv2D(32, (3, 3), activation="relu", input_shape=(64, 64, 3)))

#Pooling
cnn.add(MaxPooling2D(pool_size = (2, 2)))

# 2nd Convolution
cnn.add(Conv2D(32, (3, 3), activation="relu"))

# 2nd Pooling Layer
cnn.add(MaxPooling2D(pool_size = (2, 2)))

# Flatten the Layer
cnn.add(Flatten())

# Fully Connected Layers
cnn.add(Dense(activation = 'relu', units = 128))
cnn.add(Dense(activation = 'sigmoid', units = 1))

# Compile the Neural network
cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accurac
```

In [7]:
```
num_of_test_samples = 600
batch_size = 32
```

In [8]:
```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255)  #Image normalization.

training_set = train_datagen.flow_from_directory('chest_xray/chest_xray/train',
                                                 target_size = (64, 64),
                                                 batch_size = 32,
                                                 class_mode = 'binary')

validation_generator = test_datagen.flow_from_directory('chest_xray/chest_xray/va
    target_size=(64, 64),
    batch_size=32,
    class_mode='binary')

test_set = test_datagen.flow_from_directory('chest_xray/chest_xray/test',
                                            target_size = (64, 64),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

```
Found 5216 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

In [9]:
```
cnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 62, 62, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 29, 29, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 128)               802944
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 813,217
Trainable params: 813,217
Non-trainable params: 0
_________________________________________________________________
```

In [17]:
```
cnn_model = cnn.fit_generator(training_set,
                         steps_per_epoch = 163,
                         epochs = 10,
                         validation_data = validation_generator,
                         validation_steps = 624)
```

```
Epoch 1/10
163/163 [==============================] - ETA: 0s - loss: 0.2478 - accuracy:
0.8984WARNING:tensorflow:Your input ran out of data; interrupting training. Mak
e sure that your dataset or generator can generate at least `steps_per_epoch *
epochs` batches (in this case, 624 batches). You may need to use the repeat() f
unction when building your dataset.
163/163 [==============================] - 75s 457ms/step - loss: 0.2478 - accu
racy: 0.8984 - val_loss: 0.7005 - val_accuracy: 0.6875
Epoch 2/10
163/163 [==============================] - 73s 449ms/step - loss: 0.2157 - accu
racy: 0.9137
Epoch 3/10
163/163 [==============================] - 72s 444ms/step - loss: 0.1923 - accu
racy: 0.9247
Epoch 4/10
163/163 [==============================] - 72s 443ms/step - loss: 0.1775 - accu
racy: 0.9354
Epoch 5/10
163/163 [==============================] - 72s 439ms/step - loss: 0.1789 - accu
racy: 0.9273
Epoch 6/10
163/163 [==============================] - 71s 435ms/step - loss: 0.1782 - accu
racy: 0.9314
Epoch 7/10
163/163 [==============================] - 72s 441ms/step - loss: 0.1517 - accu
racy: 0.9448
Epoch 8/10
163/163 [==============================] - 72s 441ms/step - loss: 0.1481 - accu
racy: 0.9436
Epoch 9/10
163/163 [==============================] - 71s 437ms/step - loss: 0.1403 - accu
racy: 0.9477
Epoch 10/10
163/163 [==============================] - 73s 449ms/step - loss: 0.1303 - accu
racy: 0.9515
```

In [18]:
```
test_accu = cnn.evaluate_generator(test_set,steps=624)
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure
that your dataset or generator can generate at least `steps_per_epoch * epochs`
batches (in this case, 624 batches). You may need to use the repeat() function
when building your dataset.
```

In [19]:
```
print('The testing accuracy is :',test_accu[1]*100, '%')
```

```
The testing accuracy is : 87.5 %
```

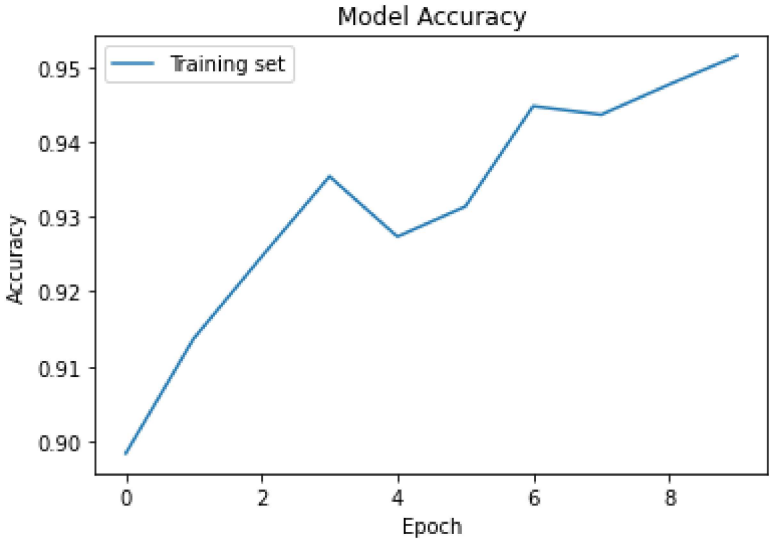
In [20]:
```
Y_pred = cnn.predict_generator(test_set, 100)
y_pred = np.argmax(Y_pred, axis=1)
# confusion_matrix(validation_generator.classes, y_pred)
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure
that your dataset or generator can generate at least `steps_per_epoch * epochs`
batches (in this case, 100 batches). You may need to use the repeat() function
when building your dataset.
```

In [21]:
```
max(y_pred)
```

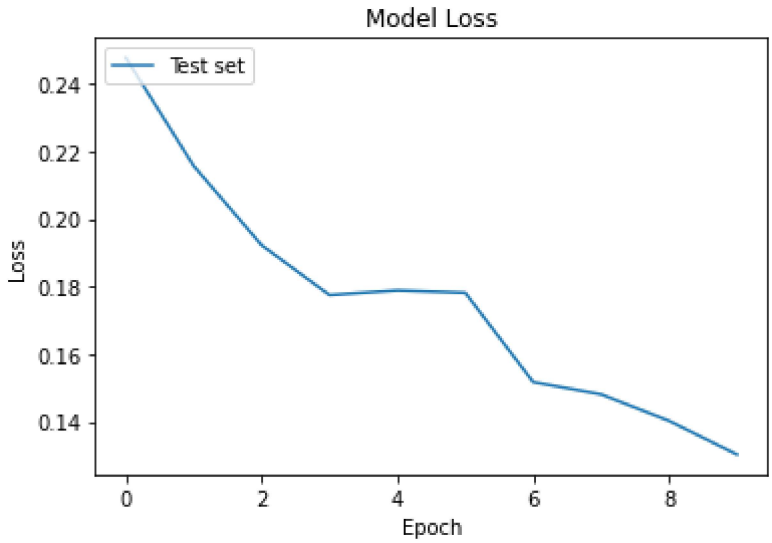
Out[21]:
```
0
```

In [23]:
```
plt.plot(cnn_model.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Training set'], loc='upper left')
plt.show()
```



In [25]:

```
plt.plot(cnn_model.history['loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend([ 'Test set'], loc='upper left')
plt.show()
```



In [ ]: