In [13]:
```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.preprocessing import image
```

In [2]:
```
train_dir = 'C:\\Users\\sd616\\Downloads\\sp\\chest_xray\\train'
val_dir = 'C:\\Users\\sd616\\Downloads\\sp\\chest_xray\\val'
test_dir = 'C:\\Users\\sd616\\Downloads\\sp\\chest_xray\\test'
```

In [3]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,   # Normalize pixel values to [0, 1]
    shear_range=0.2,  # Random shear
    zoom_range=0.2,   # Random zoom
    horizontal_flip=True  # Randomly flip images
)

val_datagen = ImageDataGenerator(rescale=1./255)
```

In [4]:
```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),  # Resize all images to 150x150
    batch_size=32,           # Process 32 images at a time
    class_mode='binary'      # Binary classification: pneumonia or normal
)
```

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

In [5]:
```
val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)
```

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

In [6]:
```
model = models.Sequential()

# First convolutional layer
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)))
model.add(layers.MaxPooling2D((2, 2)))

# Second convolutional layer
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# Third convolutional layer
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

# Flatten the results to feed into a dense layer
model.add(layers.Flatten())

# Fully connected layer
```

```
model.add(layers.Dense(128, activation='relu'))

# Output layer (binary classification)
model.add(layers.Dense(1, activation='sigmoid'))
```

```
C:\Users\sd616\anaconda\lib\site-packages\keras\src\layers\convolutional\base_conv.p
y:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Whe
n using Sequential models, prefer using an `Input(shape)` object as the first layer i
n the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [7]:
```
model.compile(
    optimizer=Adam(),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

In [8]:
```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=4,  # Set to a higher value for better results
    validation_data=val_generator,
    validation_steps=val_generator.samples // val_generator.batch_size
)
```

```
Epoch 1/4
C:\Users\sd616\anaconda\lib\site-packages\keras\src\trainers\data_adapters\py_dataset
_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**
kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
163/163 ━━━━━━━━━━━━━━━━━━━━ 330s 2s/step - accuracy: 0.7432 - loss: 0.5922 - val_acc
uracy: 0.6875 - val_loss: 0.6011
Epoch 2/4
C:\Users\sd616\anaconda\lib\contextlib.py:137: UserWarning: Your input ran out of dat
a; interrupting training. Make sure that your dataset or generator can generate at le
ast `steps_per_epoch * epochs` batches. You may need to use the `.repeat()` function
when building your dataset.
  self.gen.throw(typ, value, traceback)
163/163 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 -
val_accuracy: 0.6875 - val_loss: 0.6011
Epoch 3/4
163/163 ━━━━━━━━━━━━━━━━━━━━ 312s 2s/step - accuracy: 0.8946 - loss: 0.2332 - val_acc
uracy: 0.8750 - val_loss: 0.3527
Epoch 4/4
163/163 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.0000e+00 - loss: 0.0000e+00 -
val_accuracy: 0.8750 - val_loss: 0.3527
```

In [9]:
```
test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

test_loss, test_acc = model.evaluate(test_generator)
print(f"Test accuracy: {test_acc}")
```

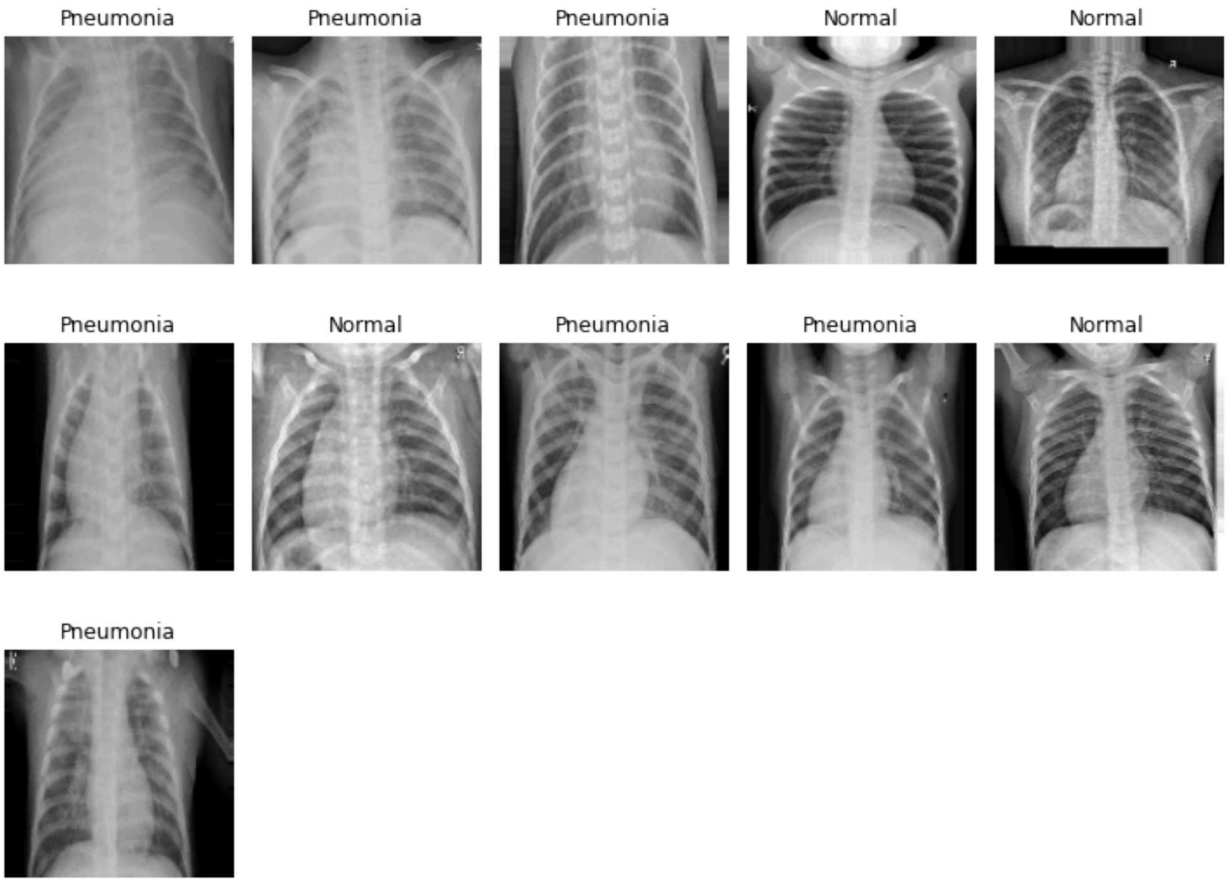
```
Found 624 images belonging to 2 classes.
20/20 ━━━━━━━━━━━━━━━━━━━━ 33s 2s/step - accuracy: 0.8832 - loss: 0.3190
Test accuracy: 0.8830128312110901
```

In [27]:
```python
x_batch, y_batch = next(train_generator)

# Plotting the images in the batch
plt.figure(figsize=(10, 10))

for i in range(11):  # Display 9 images
    plt.subplot(4,5 , i + 1)
    plt.imshow(x_batch[i])
    plt.title('Pneumonia' if y_batch[i] == 1 else 'Normal')
    plt.axis('off')

plt.tight_layout()
plt.show()
```



In [11]:
```python
def load_and_preprocess_image(img_path):
    img = image.load_img(img_path, target_size=(150, 150))  # Resize image to match mo
    img_array = image.img_to_array(img)  # Convert image to array
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension (1, 150, 150,
    img_array /= 255.0  # Rescale pixel values to [0, 1]
    return img_array
```

In [26]:
```python
img_path ='C:\\Users\\sd616\\Downloads\\sp\\chest_xray\\train\\PNEUMONIA\\person9_bact

img_array = load_and_preprocess_image(img_path)

# Make a prediction
prediction = model.predict(img_array)
```

```
# Output prediction result
if prediction[0] > 0.5:
    print("Pneumonia affected")
else:
    print("Normal")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 285ms/step
Pneumonia affected
```

In [ ]: