In [65]:
```
# Import the python libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.models import load_model
import time
```

In [2]:
```
img_dim = 64  # assigning the pixel density
```

In [3]:
```
img_data= []

for dirname, _, filenames in os.walk('C:/College/Final Sem/Emerging Trends In Data Technology/Project/Bone_Fracture/t
    for filename in filenames:
        img_data.append(os.path.join(dirname, filename))
```

In [4]:
```
len(img_data)
```

Out[4]: 9246

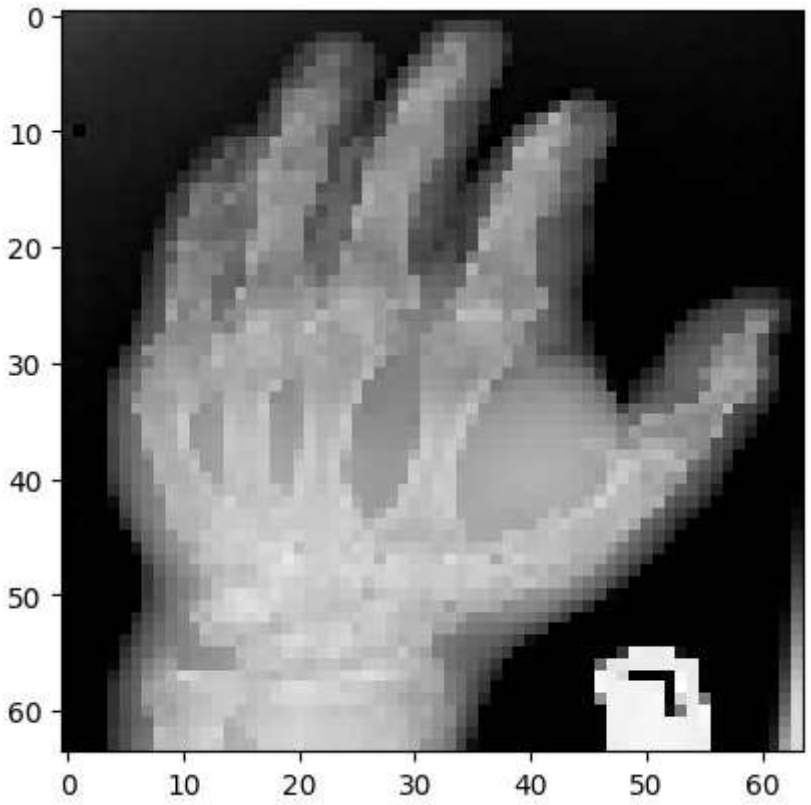
In [5]:
```
def load_dataset(data_dir):
    img_data = []
    img_labels = []

    # Walk through the directory to get all image paths
    for subdir, _, files in os.walk(data_dir):
        for file in files:
            if file.endswith(('.jpg', '.png')):  # Ensure only image files are loaded
                img_path = os.path.join(subdir, file)
                img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
                if img is not None:
                    img = cv2.resize(img, (img_dim, img_dim))  # Resize image
                    img_data.append(img)

                    # Extract label based on the folder name (fractured/not fractured)
```

```
                    parts = os.path.normpath(img_path).split(os.sep)
                    label = parts[-2]  # The second-last component is the folder name
                    img_labels.append(label)

    # Convert the labels to binary format: 0 for 'fractured', 1 for 'not fractured'
    img_labels = np.array([0 if label == "fractured" else 1 for label in img_labels])

    return np.array(img_data), img_labels

# Load the training dataset
train_images, train_labels = load_dataset('C:/College/Final Sem/Emerging Trends In Data Technology/Project/Bone_Fract

# Print the shape of train_images and a sample of the labels
print(f"Training data shape: {train_images.shape}")

# Map binary labels back to string labels
string_labels = ["fractured" if label == 0 else "not fractured" for label in train_labels]

# Print the first 10 string labels to verify
print(f"Sample labels: {string_labels[:10]}")
```

```
Training data shape: (9215, 64, 64)
Sample labels: ['fractured', 'fractured', 'fractured', 'fractured', 'fractured', 'fractured', 'fractured', 'fracture
d', 'fractured', 'fractured']
```

In [6]:

```
# Show the first image in the training set
plt.imshow(train_images[0], cmap='gray')
plt.show()

# Print the array representation of the first image
train_images[0]
```



Out[6]:
```
array([[ 55,  58,  56, ...,  11,  11,   9],
       [ 53,  52,  50, ...,  11,   9,   9],
       [ 50,  50,  46, ...,   9,   9,   8],
       ...,
       [  1,   1,   1, ...,  17, 118, 208],
       [  2,   1,   1, ...,  29, 134, 208],
       [  2,   0,   1, ...,  36, 152, 208]], dtype=uint8)
```

In [7]:
```
# Expand dimensions of the training images to include the channel dimension
train_images = np.expand_dims(train_images, axis=-1)

# Print the shape of the training images
train_images.shape
```

Out[7]:
```
(9215, 64, 64, 1)
```

In [8]:
```
print(f"Sample binary labels: {train_labels[:10]}")
```

```
Sample binary labels: [0 0 0 0 0 0 0 0 0 0]
```

In [9]:
```
# Print the shape of the target array
print(train_labels.shape)
```

```
(9215,)
```

In [10]:
```
# Define the validation set directory
validation_dir = 'C:/College/Final Sem/Emerging Trends In Data Technology/Project/Bone_Fracture/val'

# Create an empty list for storing validation files
validation_images = []
validation_labels = []
```

In [11]:
```
# Load and process each validation image
for dirname, _, filenames in os.walk(validation_dir):
    for filename in filenames:
        if filename.endswith(('.jpg', '.png')):  # Ensure only image files are loaded
            file = os.path.join(dirname, filename)
            img = cv2.imread(file, cv2.IMREAD_GRAYSCALE)
            if img is not None:
                resized_img = cv2.resize(img, (img_dim, img_dim))  # Resize image
                validation_images.append(resized_img)  # Append the resized image to the list

                # Extract label based on the folder name (fractured/not fractured)
                parts = os.path.normpath(file).split(os.sep)
                label = parts[-2]  # The second-last component is the folder name
                validation_labels.append(0 if label == "fractured" else 1)  # Binary conversion
# Convert the validation images list to a NumPy array
validation_images = np.array(validation_images)

# Convert the validation labels to a NumPy array
validation_labels = np.array(validation_labels)

# Expand dimensions of the validation images to include the channel dimension
validation_images = np.expand_dims(validation_images, axis=-1)
```

In [12]:
```
# Verify the shape of the validation images and labels array
print(f"Validation images shape: {validation_images.shape}")
```

```
print(f"Validation labels shape: {validation_labels.shape}")
```

```
Validation images shape: (798, 64, 64, 1)
Validation labels shape: (798,)
```

In [13]:
```
# Print the first 10 labels of the validation set to verify
print(f"Sample validation labels: {validation_labels[:10]}")
```

```
Sample validation labels: [0 0 0 0 0 0 0 0 0 0]
```

In [32]:
```
# Define the CNN model
model = Sequential()

# Add the first convolutional layer
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(img_dim, img_dim, 1)))

# Add max pooling to reduce dimensionality
model.add(MaxPooling2D(pool_size=(2, 2)))

# Add a second convolutional layer
model.add(Conv2D(64, (3, 3), activation='relu'))

# Add max pooling again
model.add(MaxPooling2D(pool_size=(2, 2)))

# Flatten the 3D output to 1D for the fully connected layers
model.add(Flatten())

# Add a dense (fully connected) layer
model.add(Dense(128, activation='relu'))

# Add the output layer for binary classification
model.add(Dense(1, activation='sigmoid'))

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

# Print model summary
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 62, 62, 32)        320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 29, 29, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               1605760
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 1,624,705
Trainable params: 1,624,705
Non-trainable params: 0
_________________________________________________________________
```

In [33]:
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Fit the generator in training data
datagen.fit(train_images)

# Using the generator for training
history = model.fit(datagen.flow(train_images, train_labels, batch_size=32),
                    validation_data=(validation_images, validation_labels), epochs=10)
```

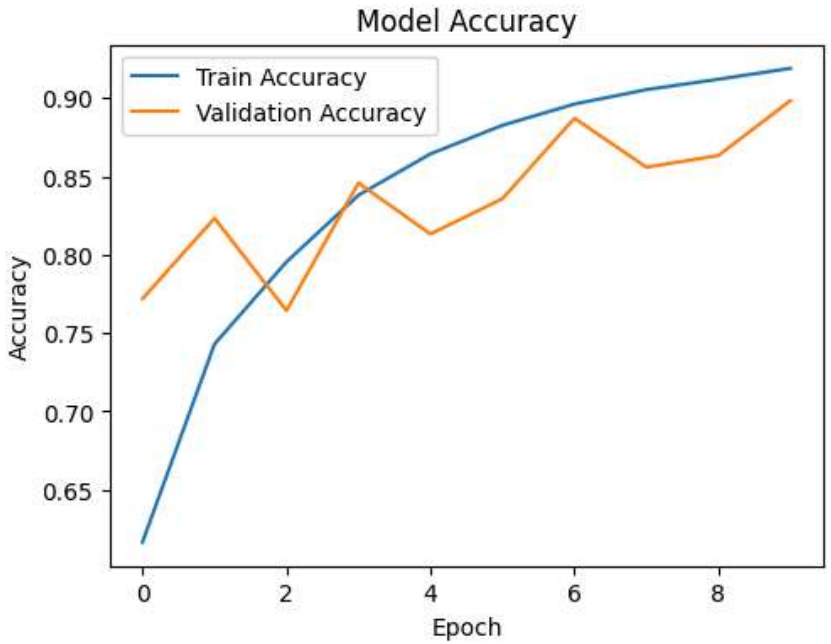
```
Epoch 1/10
288/288 [==============================] - 27s 94ms/step - loss: 1.4989 - accuracy: 0.6165 - val_loss: 0.5118 - val_a
ccuracy: 0.7719
Epoch 2/10
288/288 [==============================] - 26s 91ms/step - loss: 0.5241 - accuracy: 0.7429 - val_loss: 0.4129 - val_a
ccuracy: 0.8233
Epoch 3/10
288/288 [==============================] - 26s 91ms/step - loss: 0.4396 - accuracy: 0.7954 - val_loss: 0.5090 - val_a
ccuracy: 0.7644
Epoch 4/10
288/288 [==============================] - 28s 96ms/step - loss: 0.3715 - accuracy: 0.8381 - val_loss: 0.4179 - val_a
ccuracy: 0.8459
Epoch 5/10
288/288 [==============================] - 30s 105ms/step - loss: 0.3240 - accuracy: 0.8645 - val_loss: 0.4767 - val_
accuracy: 0.8133
Epoch 6/10
288/288 [==============================] - 27s 93ms/step - loss: 0.2868 - accuracy: 0.8827 - val_loss: 0.4213 - val_a
ccuracy: 0.8358
Epoch 7/10
288/288 [==============================] - 28s 96ms/step - loss: 0.2644 - accuracy: 0.8963 - val_loss: 0.3339 - val_a
ccuracy: 0.8872
Epoch 8/10
288/288 [==============================] - 27s 92ms/step - loss: 0.2380 - accuracy: 0.9055 - val_loss: 0.3814 - val_a
ccuracy: 0.8559
Epoch 9/10
288/288 [==============================] - 28s 99ms/step - loss: 0.2281 - accuracy: 0.9121 - val_loss: 0.3864 - val_a
ccuracy: 0.8634
Epoch 10/10
288/288 [==============================] - 28s 96ms/step - loss: 0.2127 - accuracy: 0.9190 - val_loss: 0.3147 - val_a
ccuracy: 0.8985
```

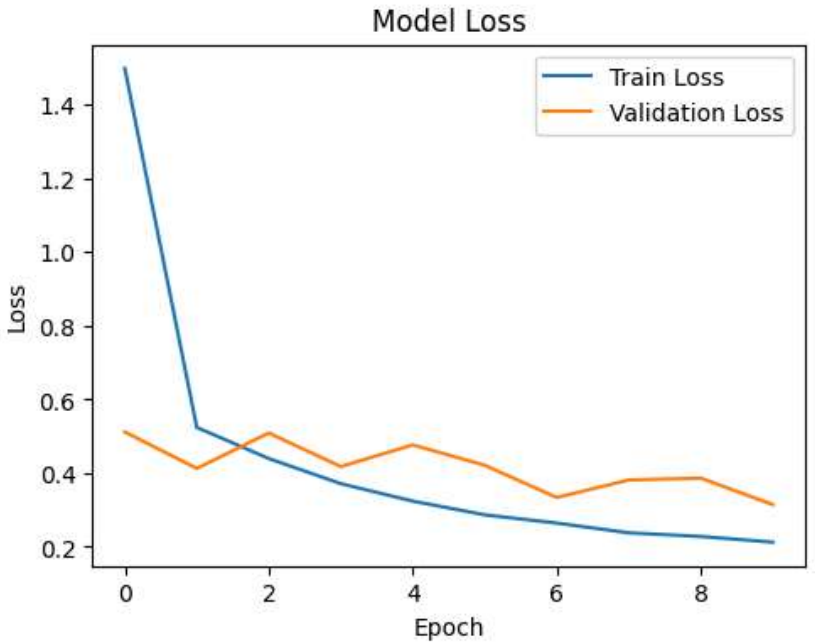
In [50]:
```
# Save the trained model
model.save('bone_fracture_classification_model.h5')
```

In [51]:
```
# Evaluate the model on the validation set
val_loss, val_acc = model.evaluate(validation_images, validation_labels)
print(f"Validation Loss: {val_loss}")
print(f"Validation Accuracy: {val_acc}")
```

```
25/25 [==============================] - 1s 23ms/step - loss: 0.3147 - accuracy: 0.8985
Validation Loss: 0.31466159224510193
Validation Accuracy: 0.8984962701797485
```

In [52]:
```
# Plot accuracy and loss over epochs
plt.figure(figsize=(12, 4))

# Plot training & validation accuracy values
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```





In [53]:
```
#Training complete
# Now testing process initiates
# Define the test set directory
test_dir = 'C:/College/Final Sem/Emerging Trends In Data Technology/Project/Bone_Fracture/test'

# Create an empty list for storing test files
test_images = []
test_labels = []

# Load and process each test image
for dirname, _, filenames in os.walk(test_dir):
    for filename in filenames:
        if filename.endswith(('.jpg', '.png')):  # Ensure only image files are loaded
            file = os.path.join(dirname, filename)
            img = cv2.imread(file, cv2.IMREAD_GRAYSCALE)
            if img is not None:
                resized_img = cv2.resize(img, (img_dim, img_dim))  # Resize image
                test_images.append(resized_img)  # Append the resized image to the list

                # Extract label based on the folder name (fractured/not fractured)
                parts = os.path.normpath(file).split(os.sep)
```

```
                label = parts[-2]  # The second-last component is the folder name
                test_labels.append(0 if label == "fractured" else 1)  # Binary conversion

# Convert the test images list to a NumPy array
test_images = np.array(test_images)

# Convert the test labels to a NumPy array
test_labels = np.array(test_labels)

# Expand dimensions of the test images to include the channel dimension
test_images = np.expand_dims(test_images, axis=-1)

# Verify the shape of the test images and labels array
print(f"Test images shape: {test_images.shape}")
print(f"Test labels shape: {test_labels.shape}")
```

```
Test images shape: (506, 64, 64, 1)
Test labels shape: (506,)
```

In [54]:
```
# Generate predictions on test images
test_predictions = model.predict(test_images)
```

In [55]:
```
# Convert probabilities to binary labels (0 or 1)
test_pred_labels = [1 if pred > 0.5 else 0 for pred in test_predictions]
```

In [39]:
```
import sys
!{sys.executable} -m pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\college\anaconda\envs\myenv\lib\site-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\college\anaconda\envs\myenv\lib\site-packages (from scikit-le
arn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in c:\college\anaconda\envs\myenv\lib\site-packages (from scikit-learn)
(1.10.1)
Requirement already satisfied: joblib>=1.1.1 in c:\college\anaconda\envs\myenv\lib\site-packages (from scikit-learn)
(1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\college\anaconda\envs\myenv\lib\site-packages (from scikit-
learn) (3.5.0)
```

In [56]:
```
from sklearn.metrics import accuracy_score

# Calculate test accuracy
```

```
test_accuracy = accuracy_score(test_labels, test_pred_labels)
print(f"Test Accuracy: {test_accuracy}")
```

```
Test Accuracy: 0.8972332015810277
```

In [57]:
```
# Manually compare a few predictions with the true labels
for i in range(500D):
    print(f"Prediction: {test_pred_labels[i]}, Actual: {test_labels[i]}")
```

```
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
```

```
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
```

```
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
```

```
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
```

```
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 1, Actual: 0
Prediction: 1, Actual: 0
Prediction: 0, Actual: 0
Prediction: 1, Actual: 0
Prediction: 0, Actual: 0
Prediction: 1, Actual: 0
Prediction: 1, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
```

```
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 0, Actual: 0
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
```

```
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
```

```
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
```

```
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
```

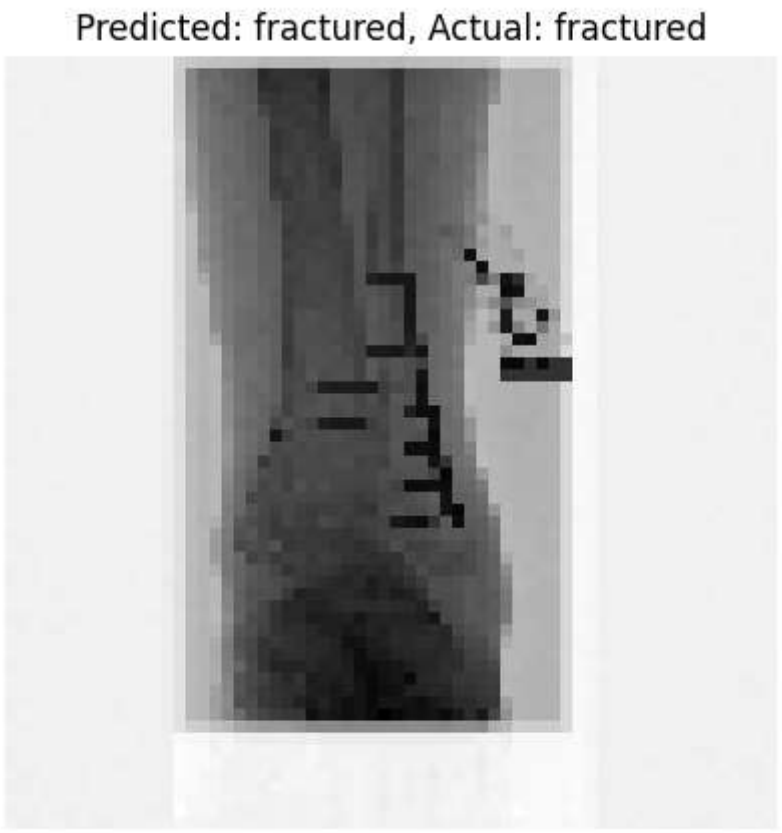
```
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
```

```
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
```

```
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 0, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
Prediction: 1, Actual: 1
```

In [58]:
```
# Plot images with actual and predicted labels
import matplotlib.pyplot as plt
```

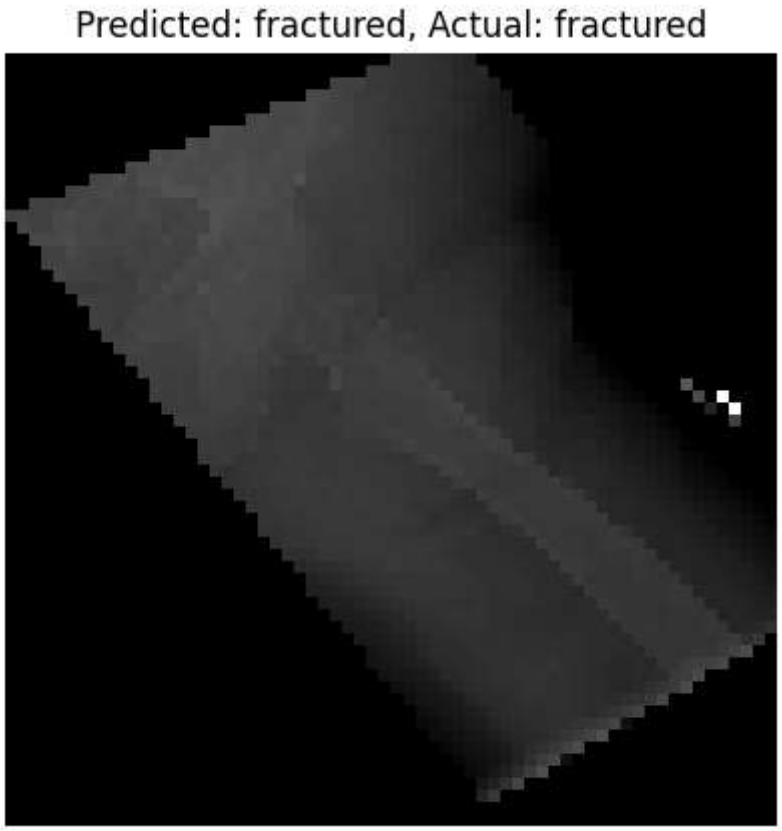
```
# Plot a few sample images along with their predictions and actual labels
for i in range(20):  # You can change the range for more images
    plt.figure(figsize=(5, 5))

    # Display the test image
    plt.imshow(test_images[i].reshape(img_dim, img_dim), cmap='gray')  # Adjusting shape as per the model input

    # Set the title as predicted and actual label
    pred_label = 'fractured' if test_pred_labels[i] == 0 else 'not fractured'
    true_label = 'fractured' if test_labels[i] == 0 else 'not fractured'
    plt.title(f"Predicted: {pred_label}, Actual: {true_label}")

    # Remove axis labels for a cleaner look
    plt.axis('off')

    plt.show()
```



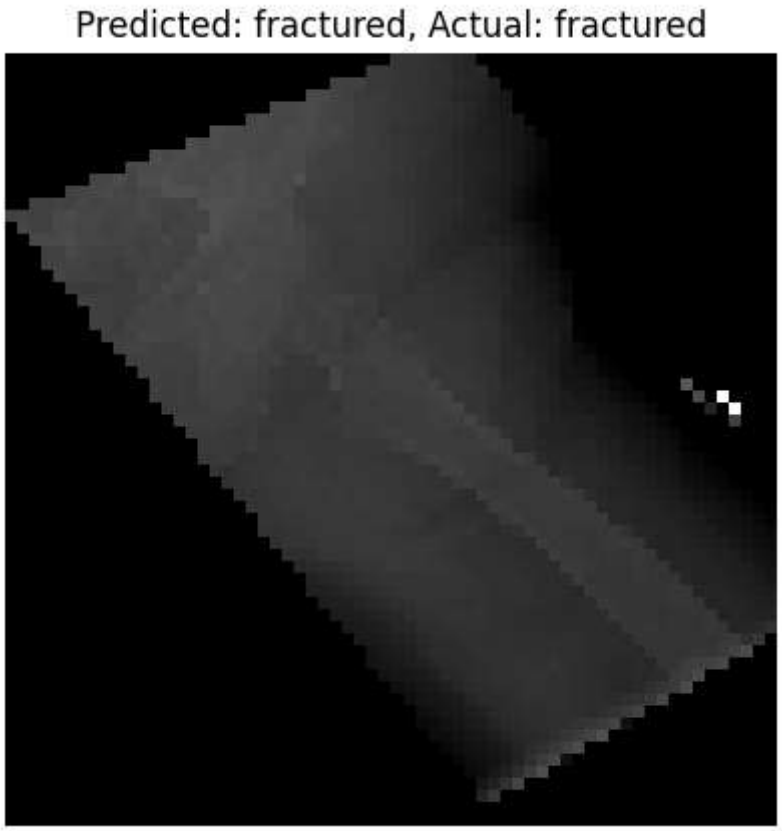

Predicted: fractured, Actual: fractured

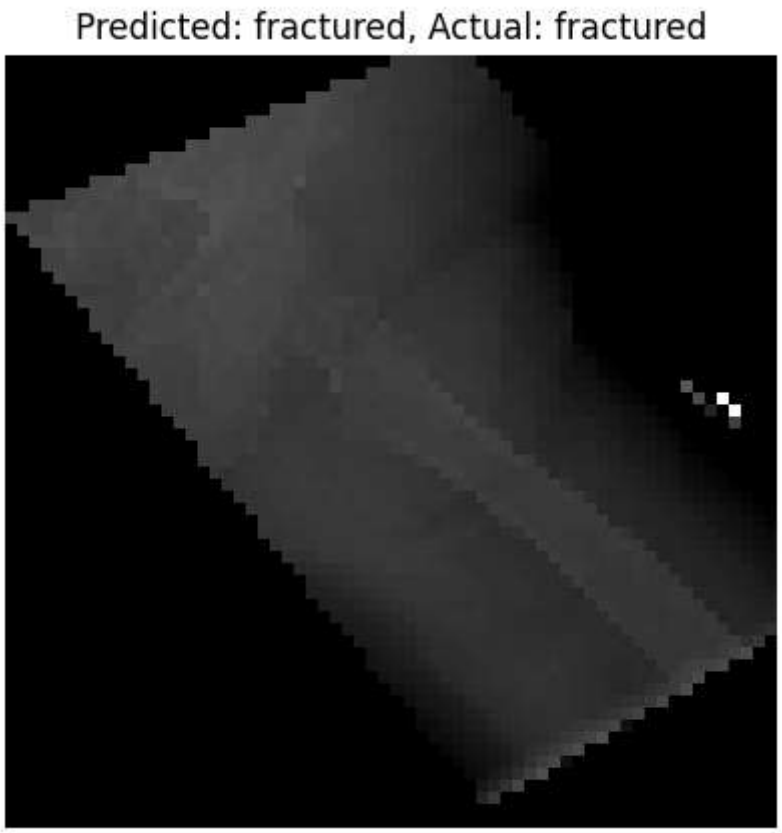

Predicted: fractured, Actual: fractured

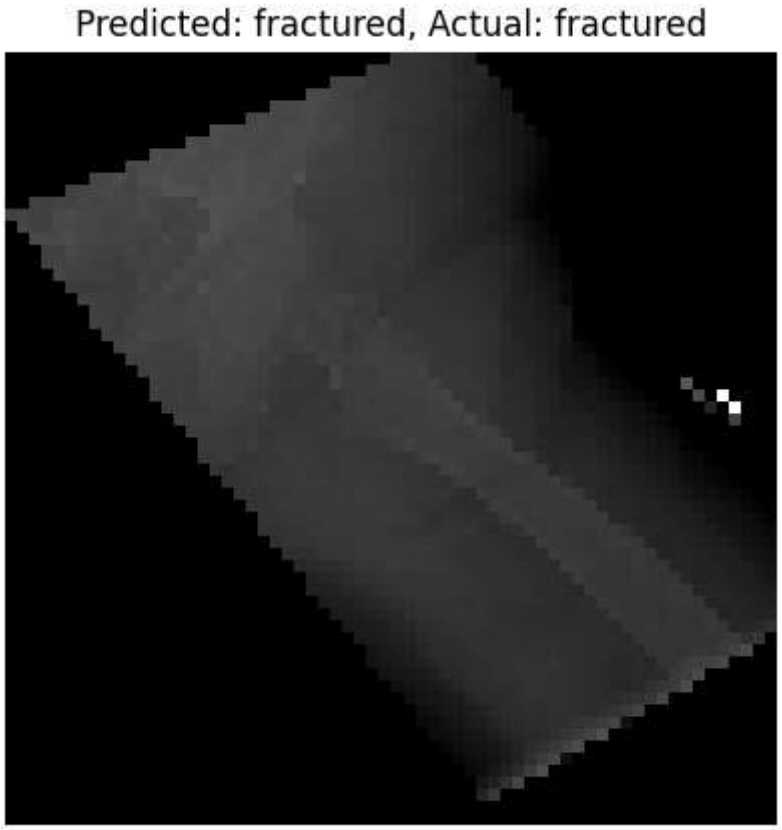

Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured

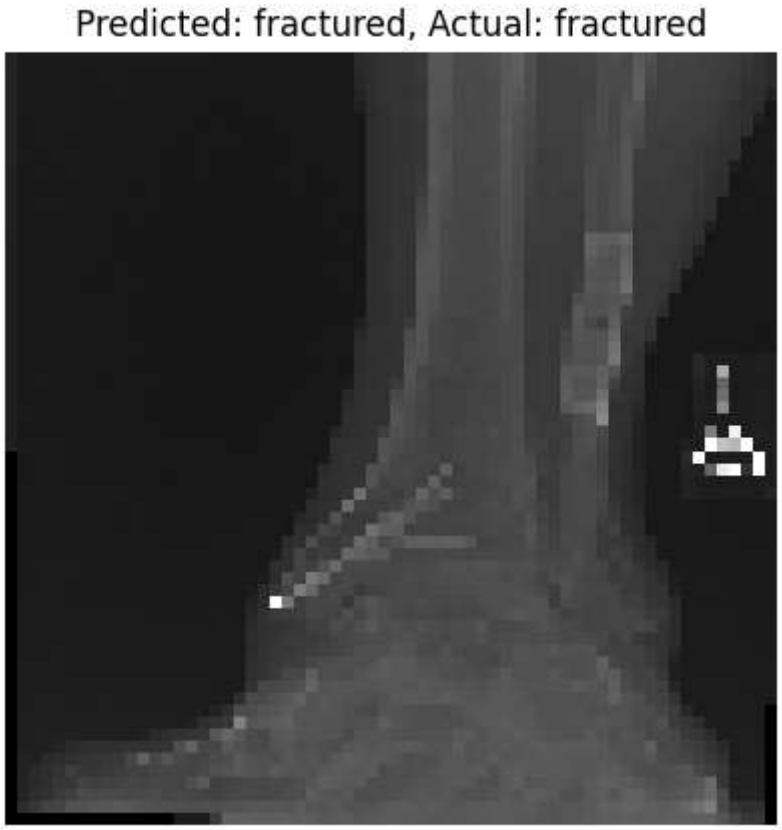

Predicted: fractured, Actual: fractured

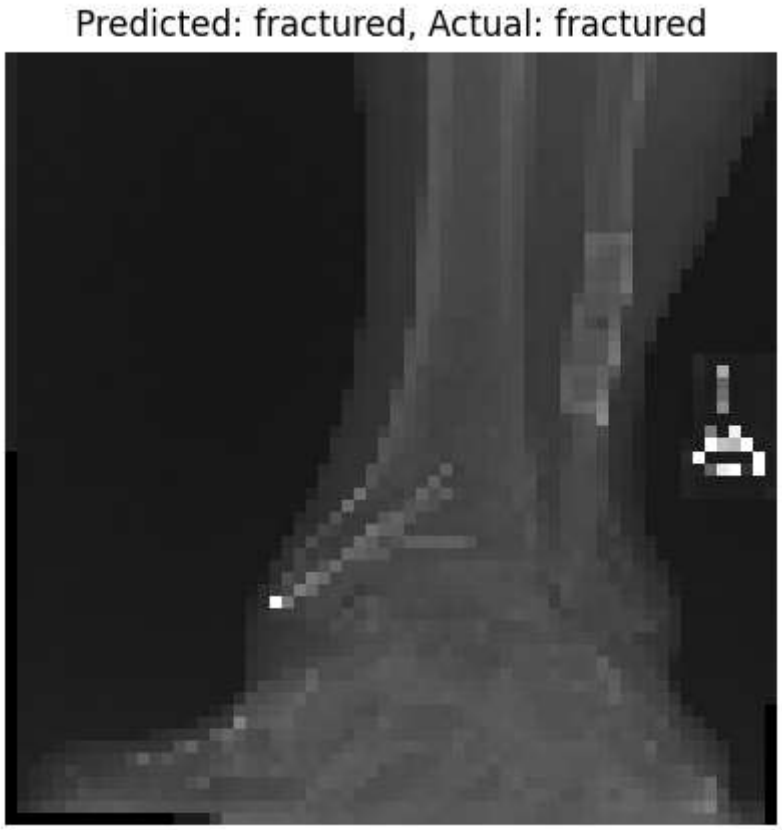

Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured

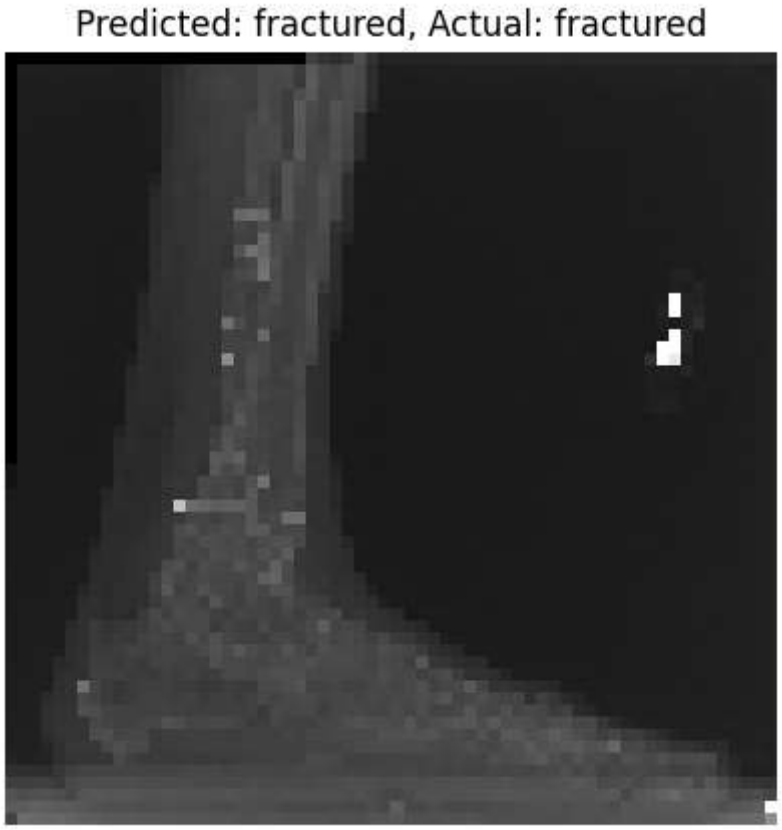

Predicted: fractured, Actual: fractured

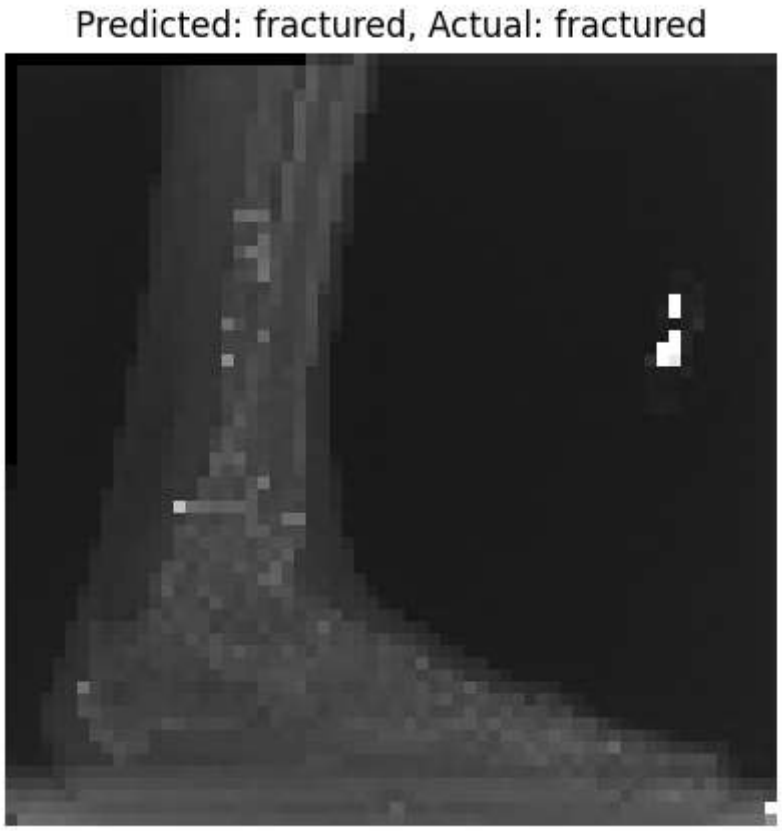

Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured

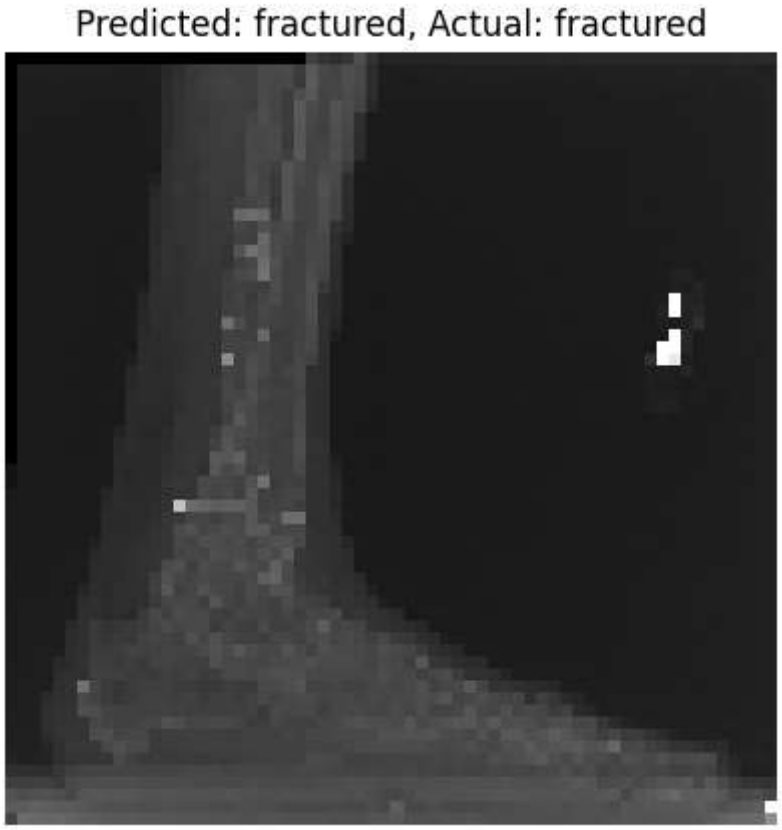

Predicted: fractured, Actual: fractured

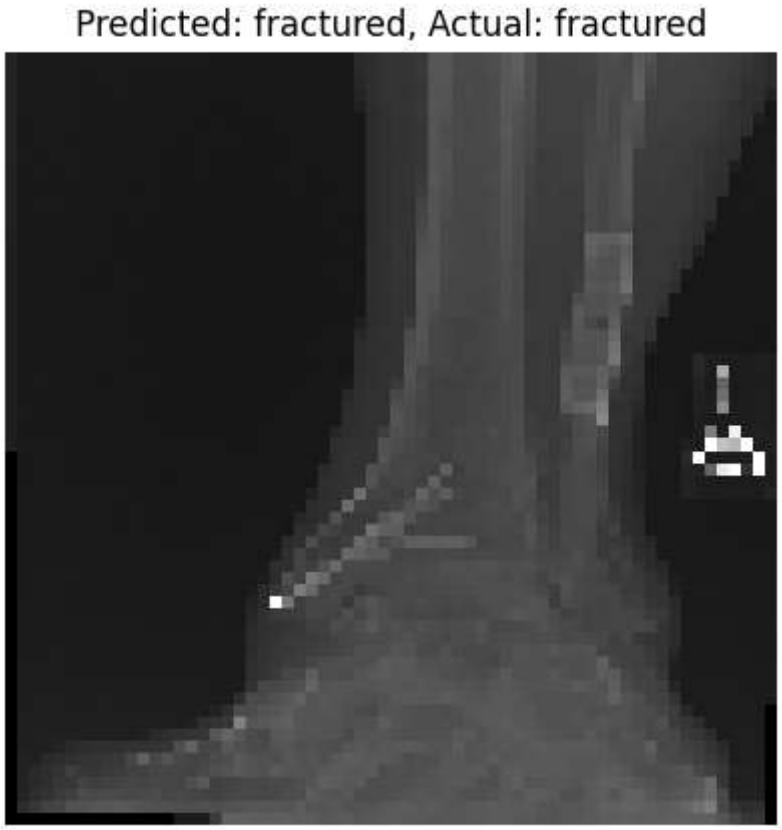

Predicted: fractured, Actual: fractured

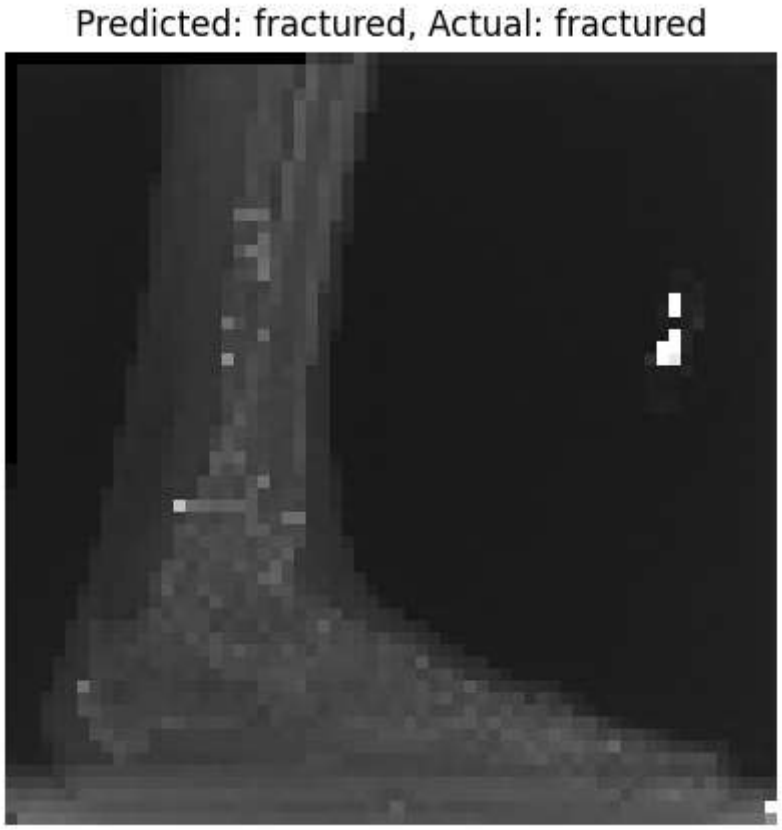

Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured


Predicted: fractured, Actual: fractured

In [59]:
```
from sklearn.metrics import classification_report

# Print detailed classification report
print(classification_report(test_labels, test_pred_labels, target_names=['fractured', 'not fractured']))
```

```
              precision    recall  f1-score   support

    fractured       0.83      0.98      0.90       238
not fractured       0.98      0.82      0.89       268

     accuracy                           0.90       506
    macro avg       0.91      0.90      0.90       506
 weighted avg       0.91      0.90      0.90       506
```

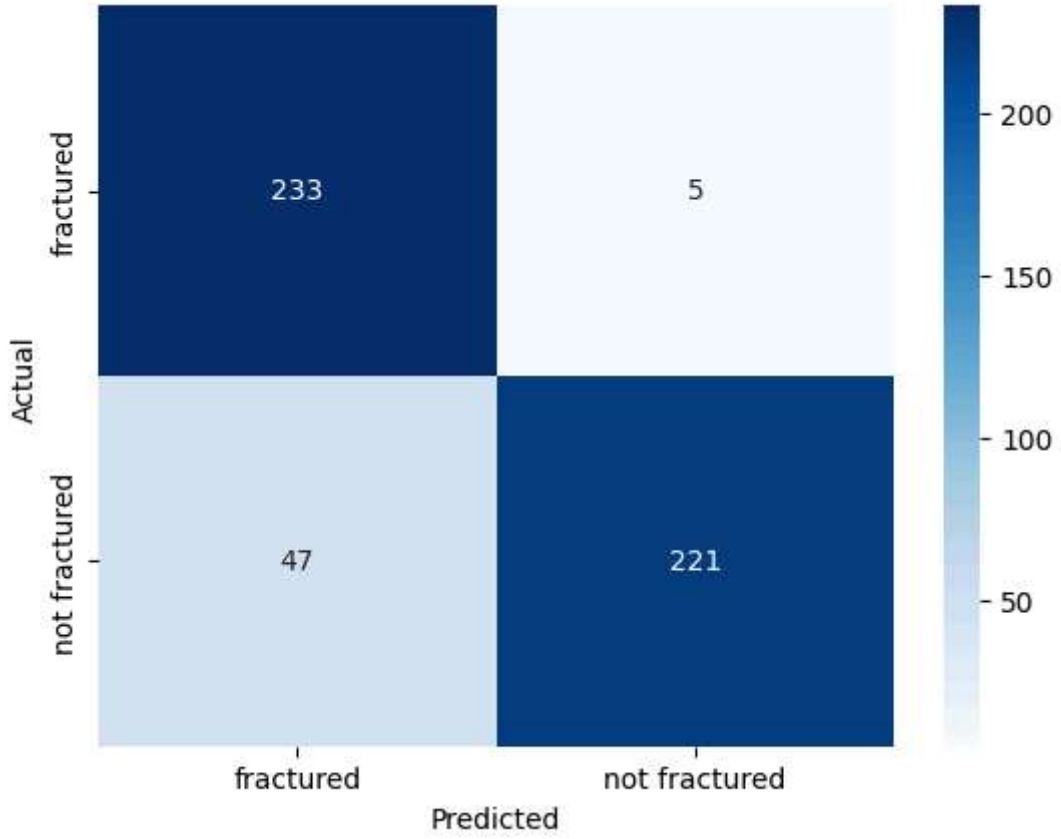
In [44]:
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\college\anaconda\envs\myenv\lib\site-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in c:\college\anaconda\envs\myenv\lib\site-packages (from seabor
n) (1.23.5)
Requirement already satisfied: pandas>=1.2 in c:\college\anaconda\envs\myenv\lib\site-packages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in c:\college\anaconda\envs\myenv\lib\site-packages (from seab
orn) (3.7.5)
Requirement already satisfied: contourpy>=1.0.1 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplotlib!
=3.6.1,>=3.4->seaborn) (1.1.1)
Requirement already satisfied: cycler>=0.10 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplotlib!=3.
6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplotlib!=
3.6.1,>=3.4->seaborn) (24.1)
Requirement already satisfied: pillow>=6.2.0 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplotlib!=3.
6.1,>=3.4->seaborn) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplotlib!
=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\college\anaconda\envs\myenv\lib\site-packages (from matplot
lib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\college\anaconda\envs\myenv\lib\site-packages (from m
atplotlib!=3.6.1,>=3.4->seaborn) (6.4.0)
Requirement already satisfied: pytz>=2020.1 in c:\college\anaconda\envs\myenv\lib\site-packages (from pandas>=1.2->se
aborn) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in c:\college\anaconda\envs\myenv\lib\site-packages (from pandas>=1.2->
seaborn) (2023.3)
Requirement already satisfied: zipp>=3.1.0 in c:\college\anaconda\envs\myenv\lib\site-packages (from importlib-resour
ces>=3.2.0->matplotlib!=3.6.1,>=3.4->seaborn) (3.17.0)
Requirement already satisfied: six>=1.5 in c:\college\anaconda\envs\myenv\lib\site-packages (from python-dateutil>=2.
7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
```

In [60]:

```python
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Generate confusion matrix
cm = confusion_matrix(test_labels, test_pred_labels)

# Plot confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['fractured', 'not fractured'], yticklabels=['fracture
```

```
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



In [61]:
```
def load_unseen_images(data_dir, img_dim=64):
    unseen_images = []

    # Walk through the directory and process each image
    for file in os.listdir(data_dir):
        if file.endswith(('.jpg', '.png')):  # Only load image files
            file_path = os.path.join(data_dir, file)
            img = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
            if img is not None:
                img_resized = cv2.resize(img, (img_dim, img_dim))  # Resize image
                unseen_images.append(img_resized)  # Add resized image to the list
```

```
    unseen_images = np.array(unseen_images)
    unseen_images = np.expand_dims(unseen_images, axis=-1)  # Add channel dimension

    print(f"Loaded {unseen_images.shape[0]} unseen images")
    return unseen_images
```

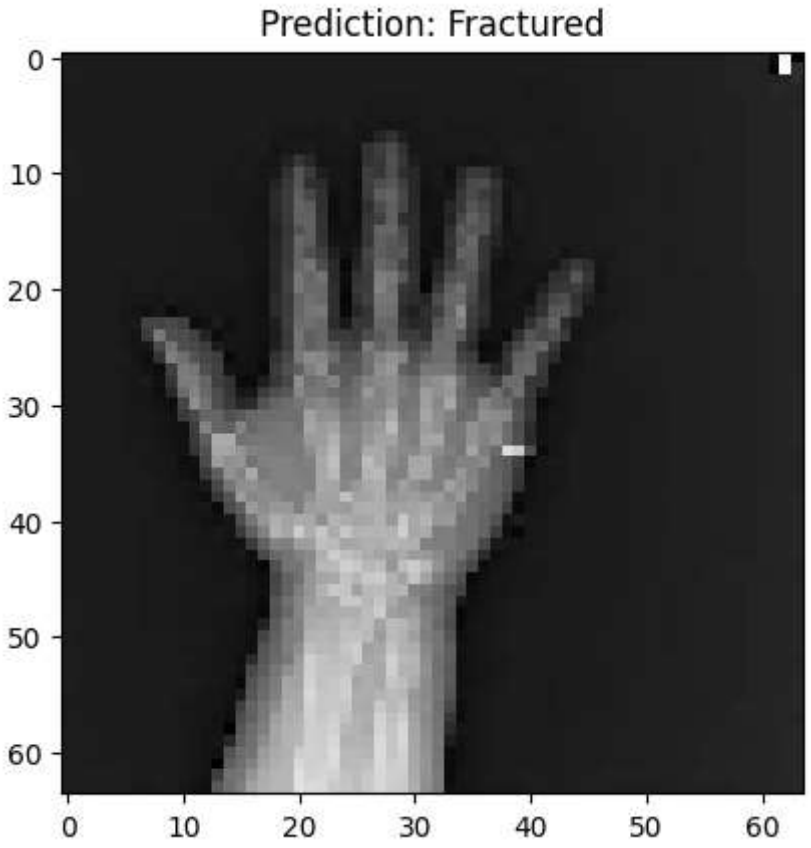
In [103…

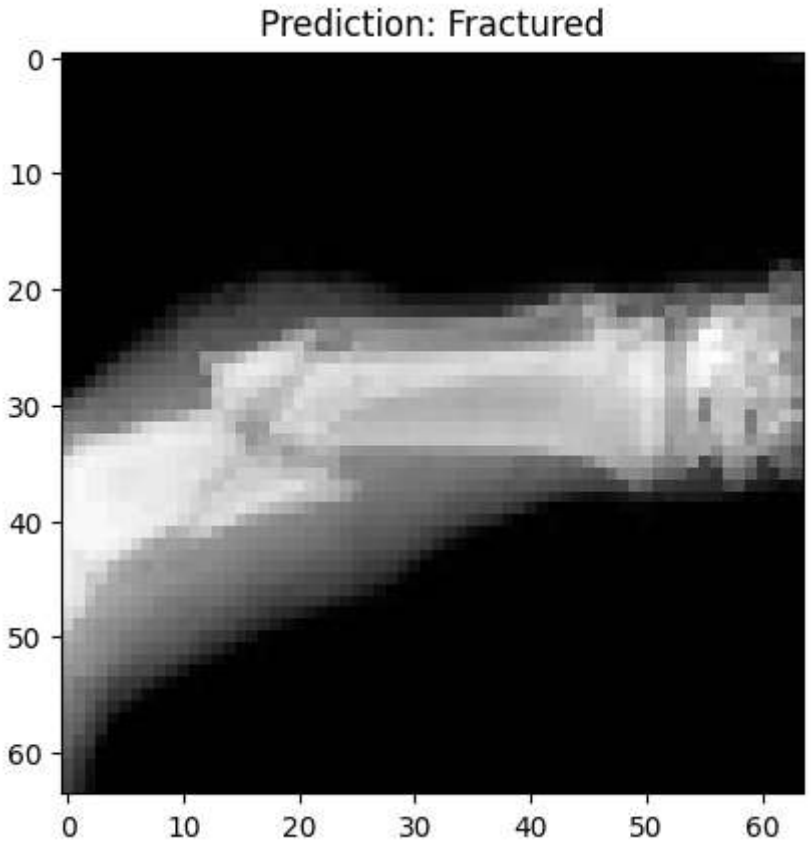
```
# Load unseen images from the folder
unseen_images = load_unseen_images('C:/College/Final Sem/Emerging Trends In Data Technology/Project/Bone_Fracture/uns

# Use the trained model to predict the labels of unseen images
unseen_predictions = model.predict(unseen_images)

# Print predictions
for i, prediction in enumerate(unseen_predictions):
    print(f"Image {i+1}: {'Fractured' if prediction < 0.5 else 'Not Fractured'}")
```

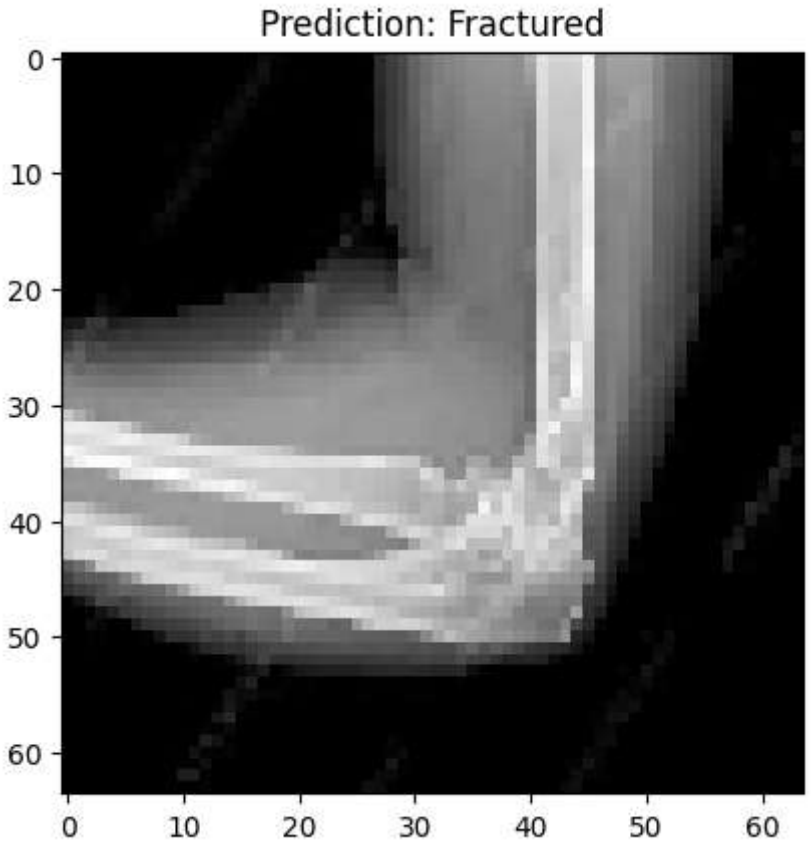
```
Loaded 8 unseen images
Image 1: Fractured
Image 2: Fractured
Image 3: Fractured
Image 4: Fractured
Image 5: Fractured
Image 6: Fractured
Image 7: Not Fractured
Image 8: Not Fractured
```

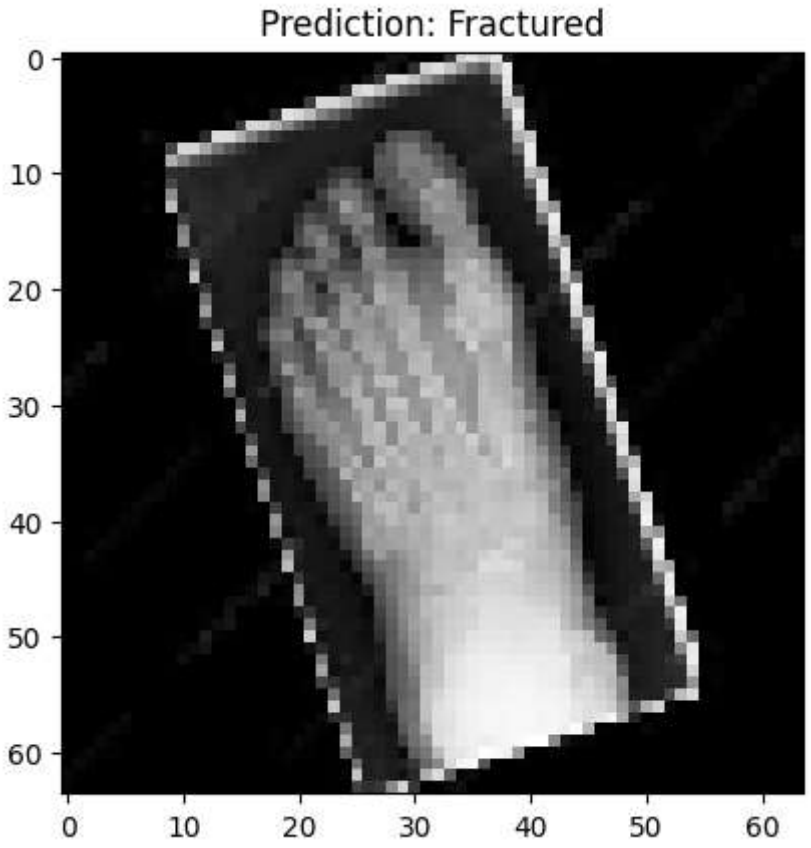
In [63]:

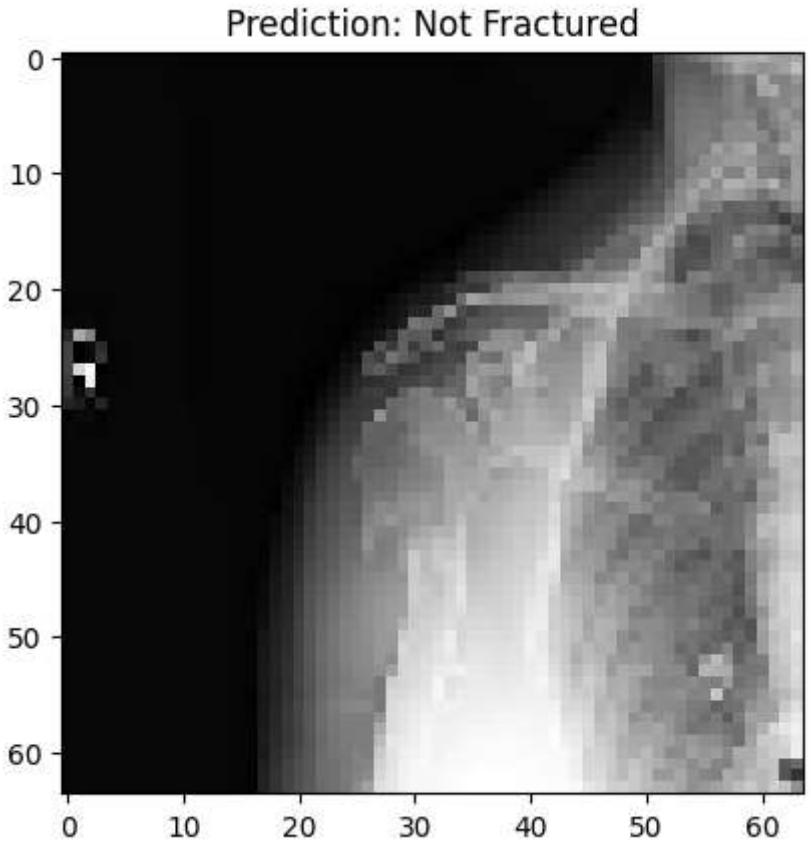
```
import matplotlib.pyplot as plt

# Visualize unseen images and their predictions
for i in range(len(unseen_images)):
    plt.imshow(unseen_images[i].reshape(img_dim, img_dim), cmap='gray')
    plt.title(f"Prediction: {'Fractured' if unseen_predictions[i] < 0.5 else 'Not Fractured'}")
    plt.show()
```

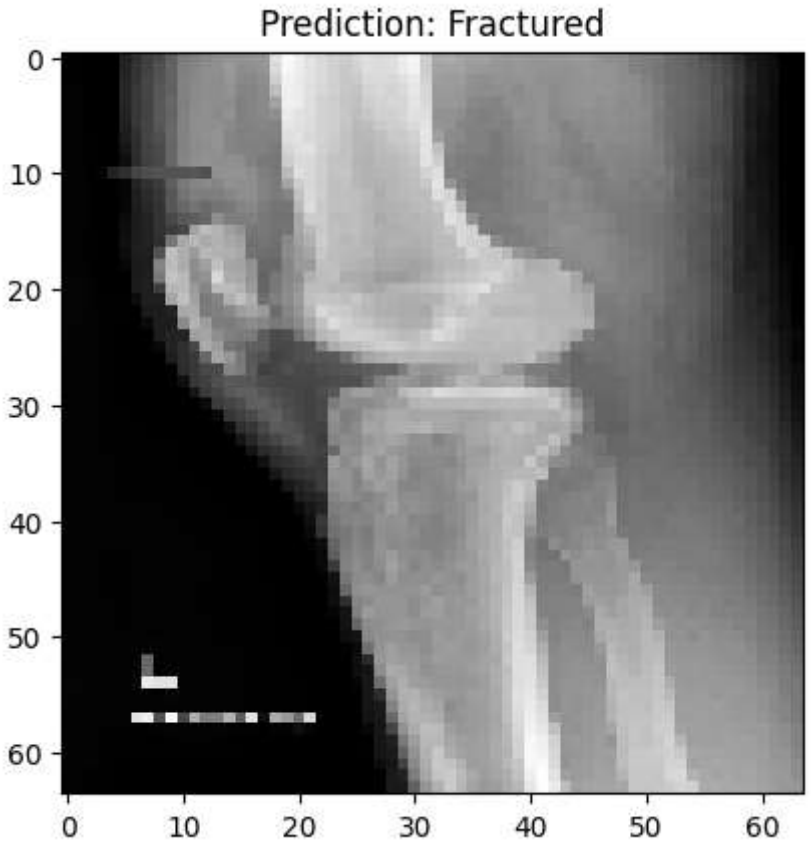

Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60

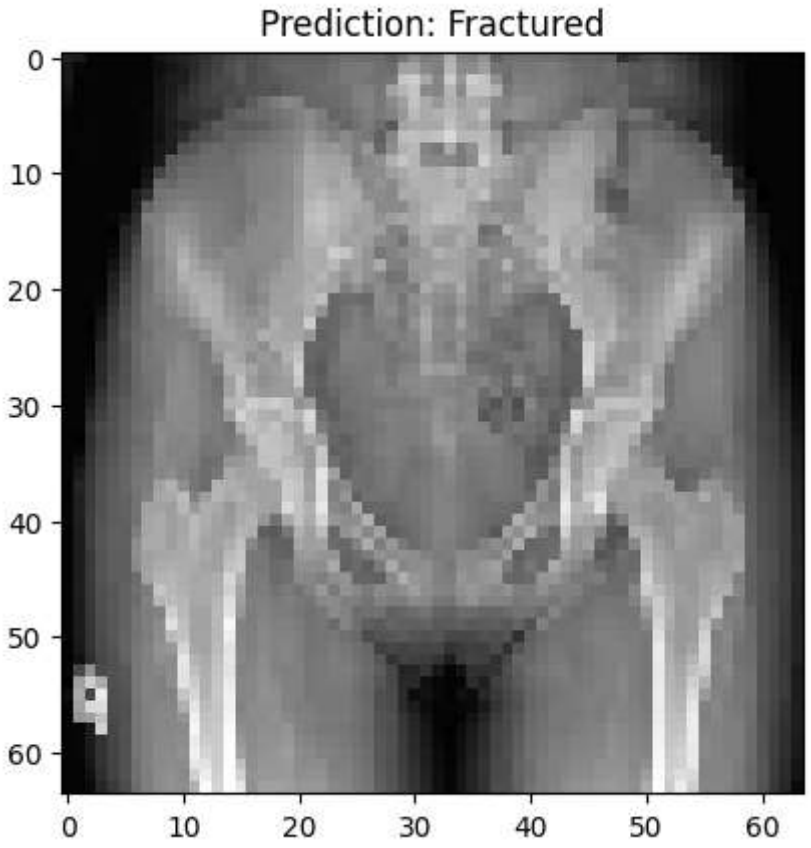

Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60

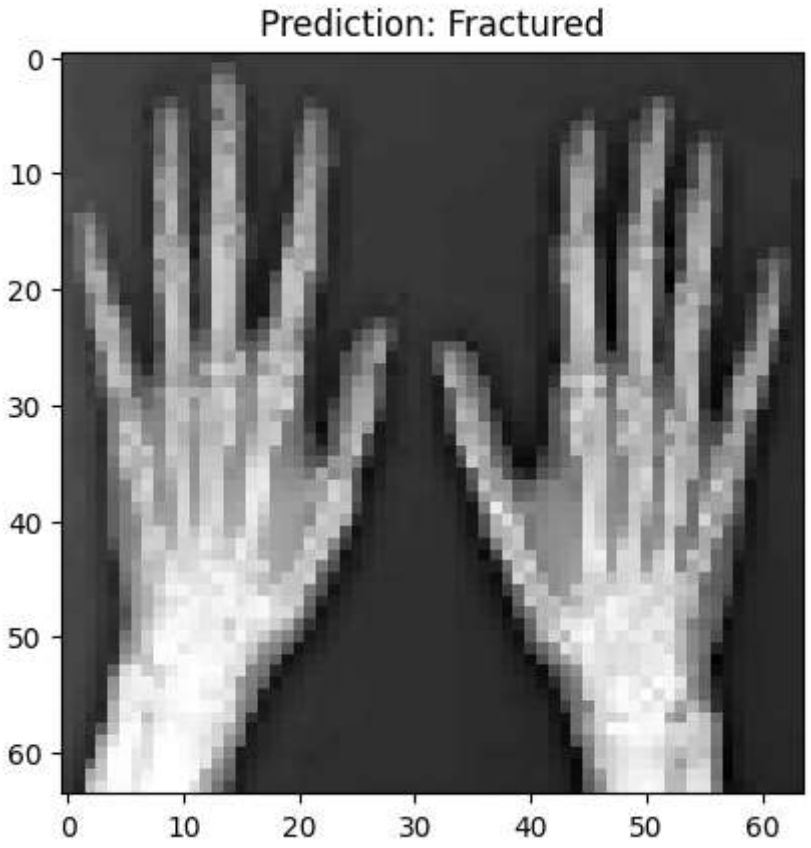

Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60

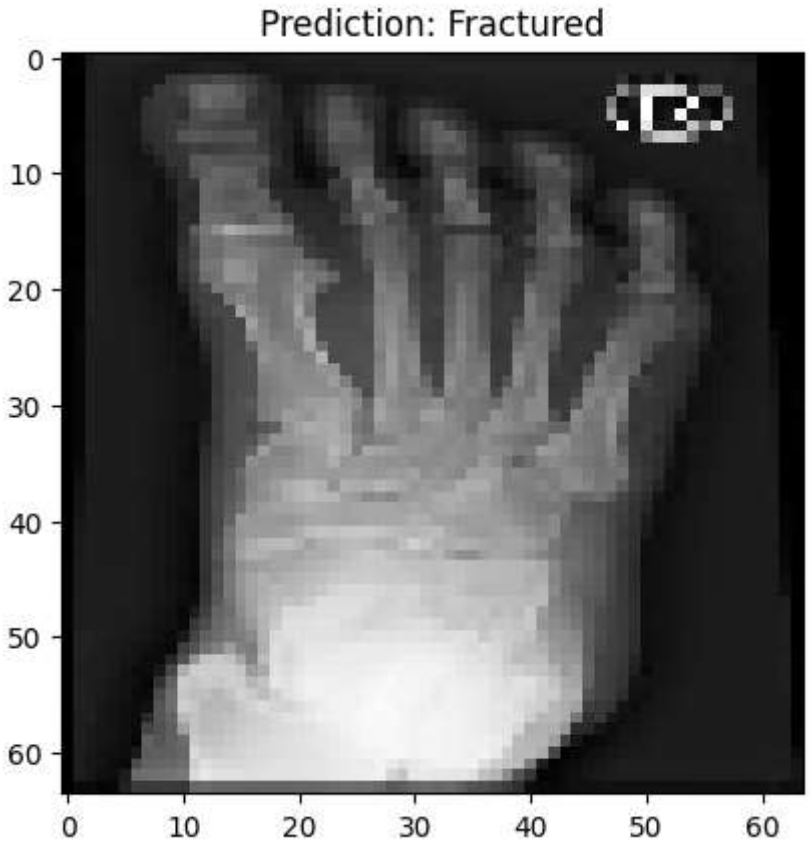

Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60

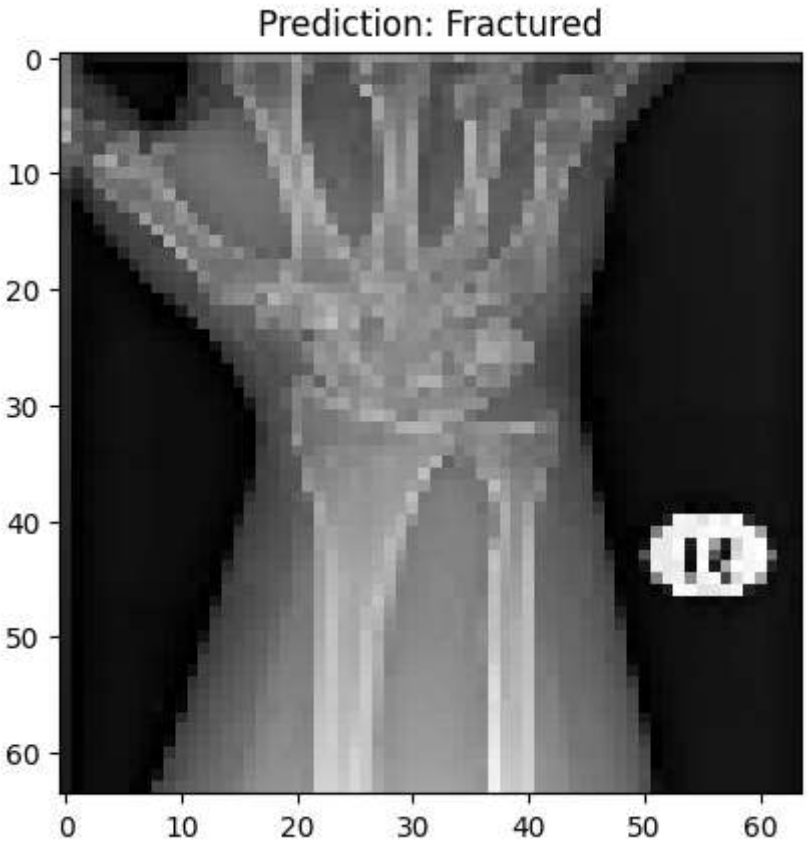

Prediction: Not Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60


Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60


Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60


Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60


Prediction: Fractured
0
10
20
30
40
50
60
0
10
20
30
40
50
60



In [ ]: