In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import os
import glob as gb
import cv2
import keras
from tensorflow.keras.models import Sequential, Model
```

In [2]:

```
trainpath = r'C:\Users\Sabesh Rajan\Downloads\archive (9)\seg_train\seg_train'
testpath = r'C:\Users\Sabesh Rajan\Downloads\archive (9)\seg_test\seg_test'
predpath = r'C:\Users\Sabesh Rajan\Downloads\archive (9)\seg_pred\seg_pred'
```

In [3]:

```
IMAGE_SIZE = (228, 228)

BATCH_SIZE = 32
```

In [4]:

```
train_ds = tf.keras.utils.image_dataset_from_directory(
  trainpath,
  seed=123,
  image_size=IMAGE_SIZE,
  batch_size=BATCH_SIZE)
```

```
Found 14034 files belonging to 6 classes.
```

In [5]:

```
test_ds = tf.keras.utils.image_dataset_from_directory(
  testpath,
  seed=123,
  image_size=IMAGE_SIZE,
  batch_size=BATCH_SIZE)
```

```
Found 3000 files belonging to 6 classes.
```

In [6]:

```
class_names = train_ds.class_names
print(class_names)
```

```
['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
```

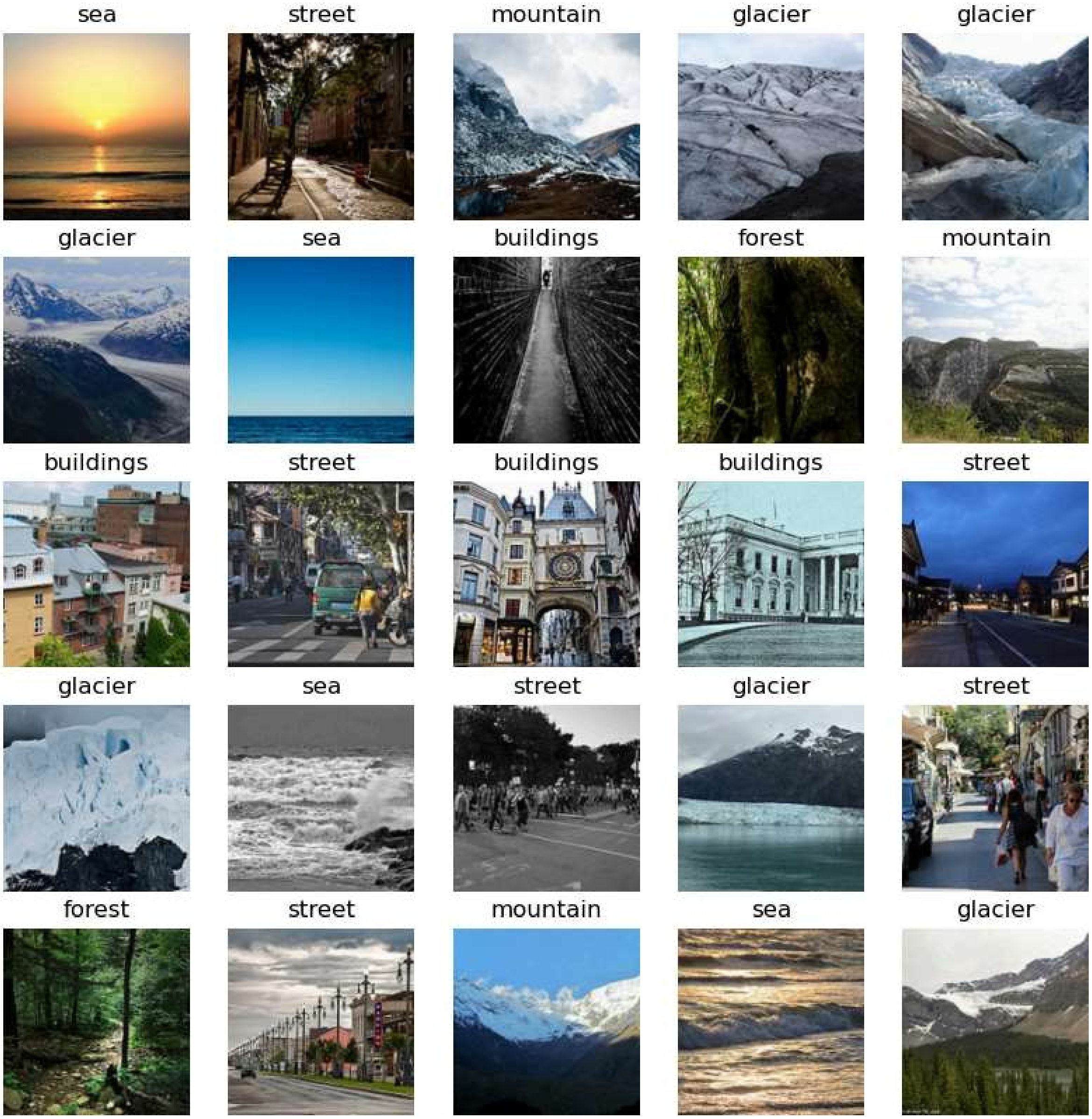
In [7]:

```
def getImagePaths(path):
    image_names = []
    for dirname, _, filenames in os.walk(path):
        for filename in filenames:
            fullpath = os.path.join(dirname, filename)
            image_names.append(fullpath)
    return image_names
images_paths = getImagePaths(predpath)
len(images_paths)
```

Out[7]:

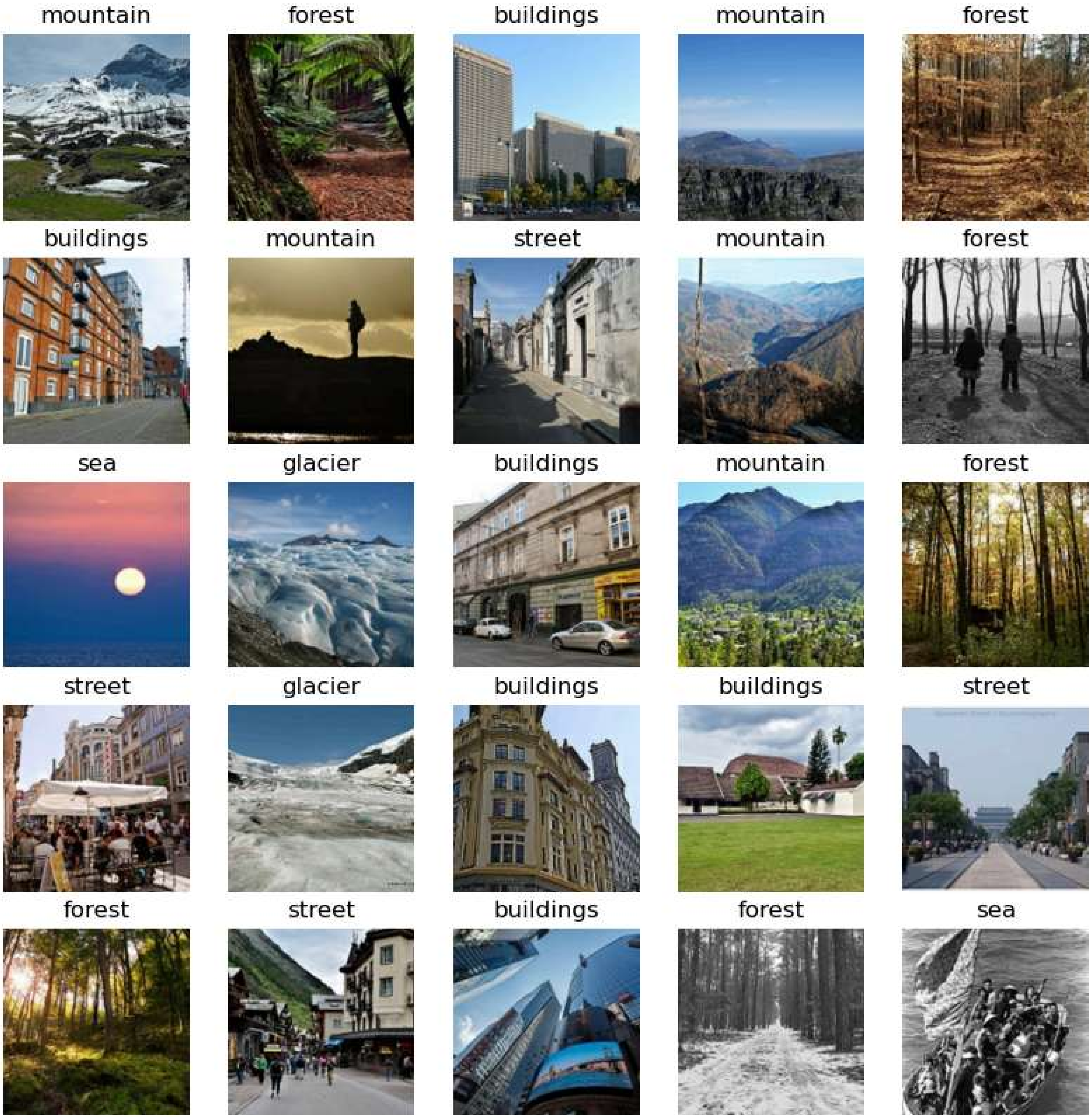
```
7301
```

In [8]:
```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(25):
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```



In [9]:

```
plt.figure(figsize=(10, 10))
for images, labels in test_ds.take(1):
    for i in range(25):
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```



In [10]:

```
import tensorflow.keras.models as Models
```

In [11]:

```
model = Models.Sequential()
model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', in
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(len(class_names), activation='softmax'))
```

```
C:\Users\Sabesh Rajan\anaconda3\Lib\site-packages\keras\src\layers\convolutio
nal\base_conv.py:99: UserWarning: Do not pass an `input_shape`/`input_dim` ar
gument to a layer. When using Sequential models, prefer using an `Input(shap
e)` object as the first layer in the model instead.
  super().__init__(
```

In [12]:

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 226, 226, 32)
max_pooling2d (MaxPooling2D)	(None, 113, 113, 32)
conv2d_1 (Conv2D)	(None, 111, 111, 32)
max_pooling2d_1 (MaxPooling2D)	(None, 55, 55, 32)
conv2d_2 (Conv2D)	(None, 53, 53, 64)
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)
conv2d_3 (Conv2D)	(None, 24, 24, 64)
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 64)
conv2d_4 (Conv2D)	(None, 10, 10, 64)
max_pooling2d_4 (MaxPooling2D)	(None, 5, 5, 64)
flatten (Flatten)	(None, 1600)
dense (Dense)	(None, 1024)
dropout (Dropout)	(None, 1024)
dense_1 (Dense)	(None, 128)
dropout_1 (Dropout)	(None, 128)
dense_2 (Dense)	(None, 6)

Total params: 1,873,894 (7.15 MB)

Trainable params: 1,873,894 (7.15 MB)

Non-trainable params: 0 (0.00 B)

In [13]:

```
from tensorflow.keras.optimizers import Adam
model.compile(
    optimizer = Adam(learning_rate = 0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    #loss = "categorical_crossentropy",
    metrics = ["accuracy"])
```

In [14]:

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

earlystopping = EarlyStopping(monitor='val_loss',
                              patience=5,
                              verbose=1,
                              mode='min'
                             )

checkpointer = ModelCheckpoint(filepath='bestvalue.keras', verbose=0, save_bes
callback_list = [checkpointer, earlystopping]
```

In [15]:
```
history = model.fit(train_ds,
    validation_data=test_ds,
    epochs=40,
    callbacks=callback_list
)
```

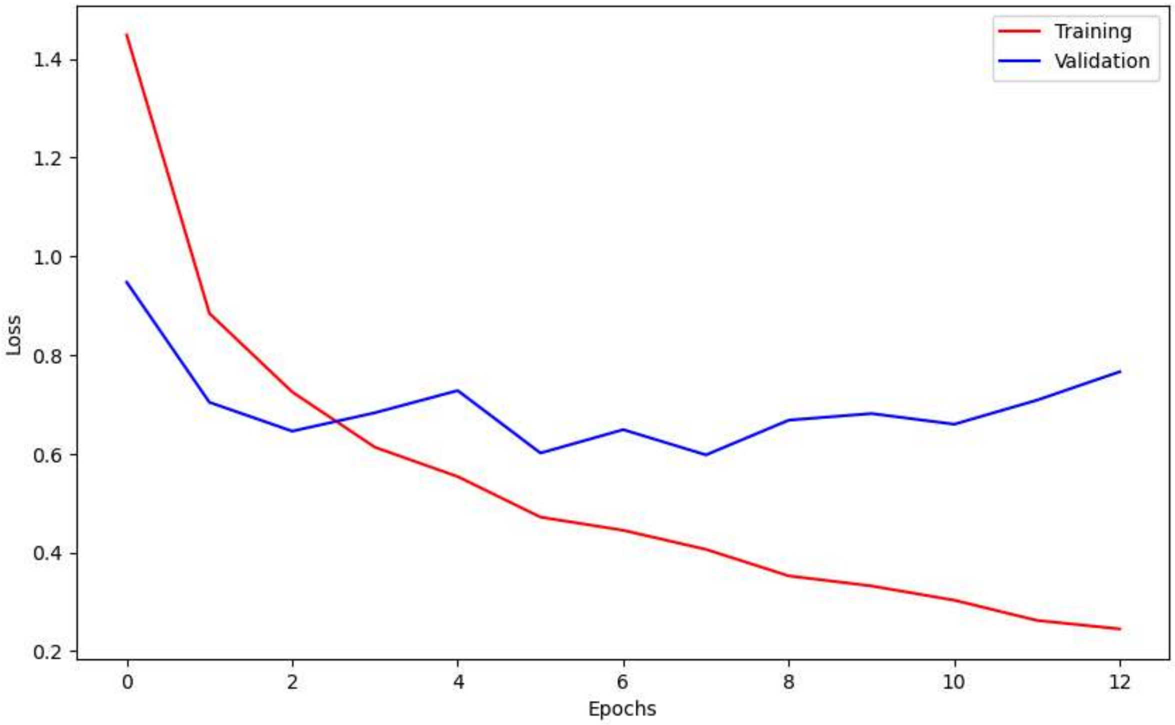
```
Epoch 1/40

C:\Users\Sabesh Rajan\anaconda3\Lib\site-packages\keras\src\backend\tensorflo
w\nn.py:599: UserWarning: "`sparse_categorical_crossentropy` received `from_l
ogits=True`, but the `output` argument was produced by a Softmax activation a
nd thus does not represent logits. Was this intended?
  output, from_logits = _get_logits(

439/439 ━━━━━━━━━━━━━━━━━━━━ 89s 197ms/step - accuracy: 0.3862 - loss: 2.6572
- val_accuracy: 0.6083 - val_loss: 0.9477
Epoch 2/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 82s 188ms/step - accuracy: 0.6394 - loss: 0.9163
- val_accuracy: 0.7393 - val_loss: 0.7043
Epoch 3/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 85s 194ms/step - accuracy: 0.7259 - loss: 0.7551
- val_accuracy: 0.7687 - val_loss: 0.6459
Epoch 4/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 69s 156ms/step - accuracy: 0.7753 - loss: 0.6306
- val_accuracy: 0.7650 - val_loss: 0.6832
Epoch 5/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 76s 172ms/step - accuracy: 0.8011 - loss: 0.5596
- val_accuracy: 0.7693 - val_loss: 0.7281
Epoch 6/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 101s 230ms/step - accuracy: 0.8259 - loss: 0.488
7 - val_accuracy: 0.7877 - val_loss: 0.6015
Epoch 7/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 78s 177ms/step - accuracy: 0.8297 - loss: 0.4772
- val_accuracy: 0.7893 - val_loss: 0.6489
Epoch 8/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 70s 158ms/step - accuracy: 0.8405 - loss: 0.4371
- val_accuracy: 0.8093 - val_loss: 0.5978
Epoch 9/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 70s 158ms/step - accuracy: 0.8646 - loss: 0.3699
- val_accuracy: 0.7897 - val_loss: 0.6682
Epoch 10/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 76s 174ms/step - accuracy: 0.8737 - loss: 0.3422
- val_accuracy: 0.8003 - val_loss: 0.6816
Epoch 11/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 69s 158ms/step - accuracy: 0.8874 - loss: 0.3044
- val_accuracy: 0.8020 - val_loss: 0.6599
Epoch 12/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 71s 161ms/step - accuracy: 0.9048 - loss: 0.2723
- val_accuracy: 0.8047 - val_loss: 0.7087
Epoch 13/40
439/439 ━━━━━━━━━━━━━━━━━━━━ 72s 164ms/step - accuracy: 0.9094 - loss: 0.2572
- val_accuracy: 0.8033 - val_loss: 0.7663
Epoch 13: early stopping
```

In [16]:

```
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(loss))

fig = plt.figure(figsize=(10,6))
plt.plot(epochs,loss,c="red",label="Training")
plt.plot(epochs,val_loss,c="blue",label="Validation")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
```

Out[16]:

```
<matplotlib.legend.Legend at 0x1fd7f102b90>
```

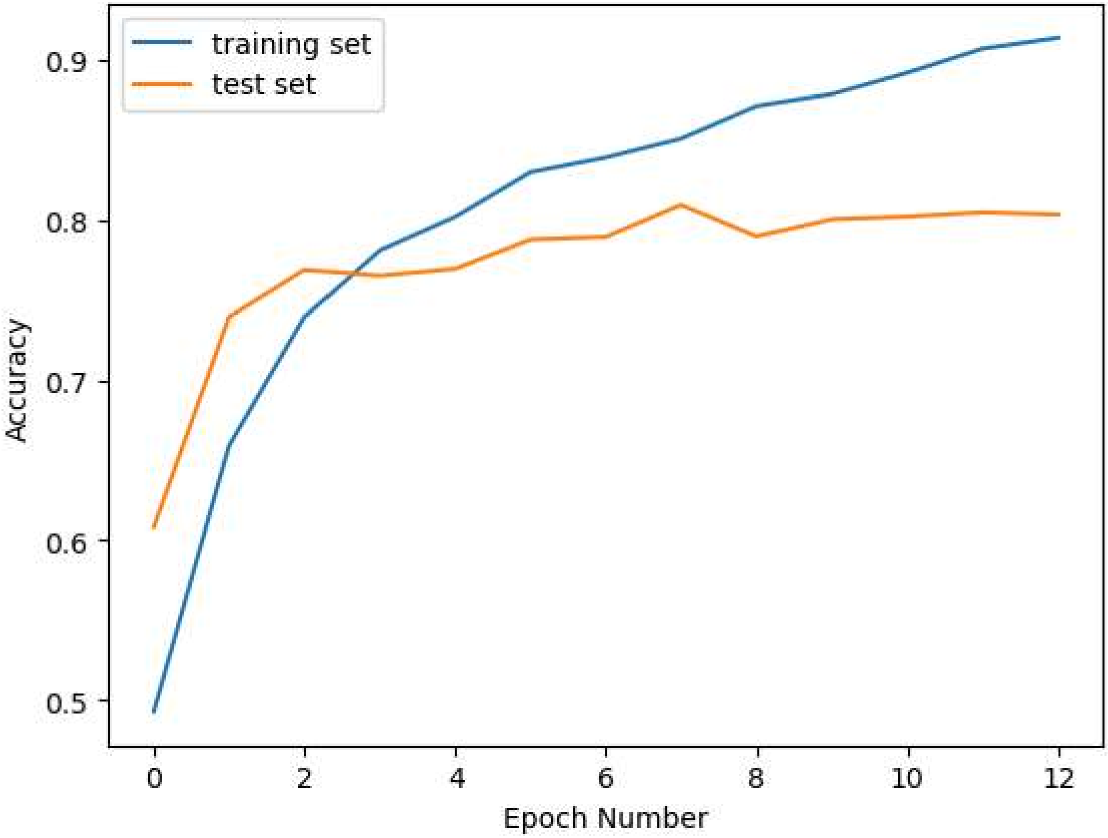


In [17]:

```
plt.xlabel('Epoch Number')
plt.ylabel('Accuracy')
plt.plot(history.history['accuracy'], label='training set')
plt.plot(history.history['val_accuracy'], label='test set')
plt.legend()
```

Out[17]:

```
<matplotlib.legend.Legend at 0x1fd0a1059d0>
```



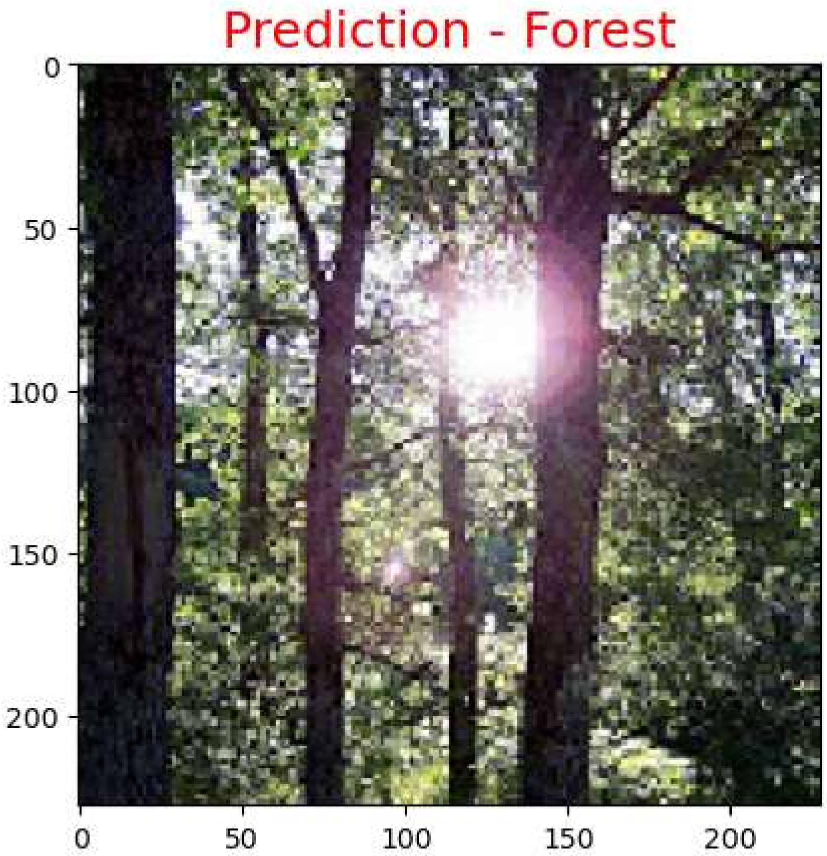
In [18]:

```
def predict_image(filename, model):
    img_ = image.load_img(filename, target_size=(228, 228))
    img_array = image.img_to_array(img_)
    img_processed = np.expand_dims(img_array, axis=0)
    img_processed /= 255.

    prediction = model.predict(img_processed)

    index = np.argmax(prediction)

    plt.title("Prediction - {}".format(str(class_names[index]).title()), size=
    plt.imshow(img_array)
```

In [19]:

```
from tensorflow.keras.preprocessing import image
predict_image(r"C:\Users\Sabesh Rajan\Downloads\archive (9)\seg_pred\seg_pred\
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 115ms/step
```



Prediction - Forest

In [ ]: