In [2]:
```
# Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image
```

In [3]:
```
data_dir = 'C:/Users/HP/Desktop/python projects/food-101/images/'
data = tf.keras.preprocessing.image_dataset_from_directory(data_dir)
```

```
Found 101000 files belonging to 101 classes.
```

In [4]:
```
# Create an ImageDataGenerator and do Image Augmentation
datagen = ImageDataGenerator(
        rescale = 1./255,
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest',
        validation_split = 0.2)
```

In [5]:
```
height = 228
width = 228
channels = 3
batch_size = 32
img_shape = (height, width, channels)
img_size = (height, width)
```

In [6]:
```
train_data = datagen.flow_from_directory(
    data_dir,
    target_size = img_size,
    batch_size = batch_size,
    class_mode = 'categorical',
    subset = 'training')

val_data = datagen.flow_from_directory(
    data_dir,
    target_size = img_size,
    batch_size = batch_size,
    class_mode='categorical',
    subset = 'validation')
```

```
Found 80800 images belonging to 101 classes.
Found 20200 images belonging to 101 classes.
```

In [7]:
```
num_classes = len(data.class_names)
print('.... Number of Classes : {0} ....'.format(num_classes))
```

```
.... Number of Classes : 101 ....
```

In [8]:
```
#Defing a function to see images
def show_img(data):
    plt.figure(figsize=(15,15))
    for images, labels in data.take(1):
        for i in range(9):
            ax = plt.subplot(3, 3, i + 1)
            ax.imshow(images[i].numpy().astype("uint8"))
            ax.axis("off")

#Plotting the images in dataset
show_img(data)
```

In [9]:
```
# load pre-trained InceptionV3
pre_trained = InceptionV3(weights='imagenet', include_top=False, input_shap

for layer in pre_trained.layers:
    layer.trainable = False
```

In [10]:
```
x = pre_trained.output
x = BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001)(x)
x = Dropout(0.2)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(num_classes, activation='softmax')(x)

model = Model(inputs = pre_trained.input, outputs = predictions)
model.compile(optimizer = Adam(learning_rate=0.001), loss='categorical_cros
```

In [11]:
```
model.summary()
```

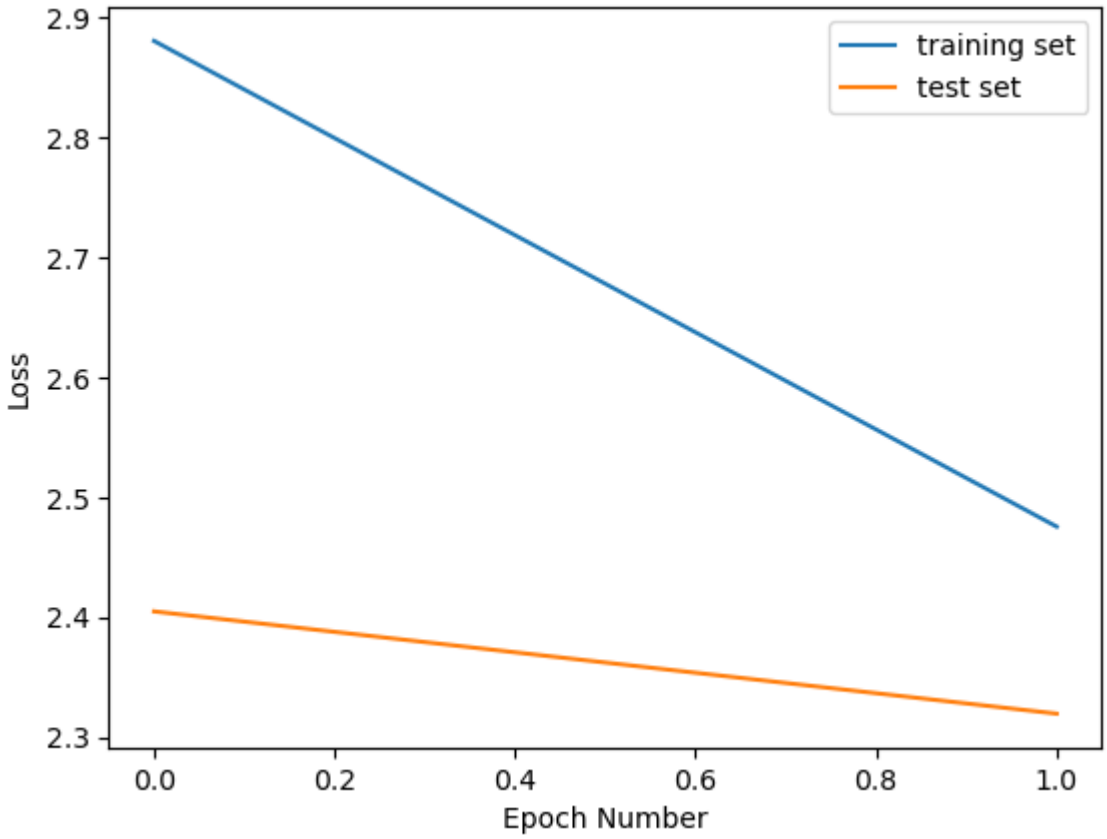
```
 chNormalization)

 batch_normalization_9 (Bat  (None, 26, 26, 96)          288       ['c
onv2d_9[0][0]']
 chNormalization)

 activation_6 (Activation)   (None, 26, 26, 48)          0         ['b
atch_normalization_6[0][0]'
                                                                    ]

 activation_9 (Activation)   (None, 26, 26, 96)          0         ['b
atch_normalization_9[0][0]'
                                                                    ]

 average_pooling2d (Average  (None, 26, 26, 192)         0         ['m
ax_pooling2d_1[0][0]']
 Pooling2D)

 conv2d_5 (Conv2D)           (None, 26, 26, 64)          12288     ['m
```

In [12]:
```
STEP_SIZE_TRAIN = train_data.n // train_data.batch_size
STEP_SIZE_VALID = val_data.n // val_data.batch_size

history = model.fit(train_data,
                    steps_per_epoch = STEP_SIZE_TRAIN,
                    validation_data = val_data,
                    validation_steps = STEP_SIZE_VALID,
                    epochs = 2,
                    verbose = 1)
```

```
Epoch 1/2
2525/2525 [==============================] - 4967s 2s/step - loss: 2.8807
- accuracy: 0.3475 - val_loss: 2.4050 - val_accuracy: 0.4210
Epoch 2/2
2525/2525 [==============================] - 4104s 2s/step - loss: 2.4757
- accuracy: 0.3995 - val_loss: 2.3198 - val_accuracy: 0.4342
```

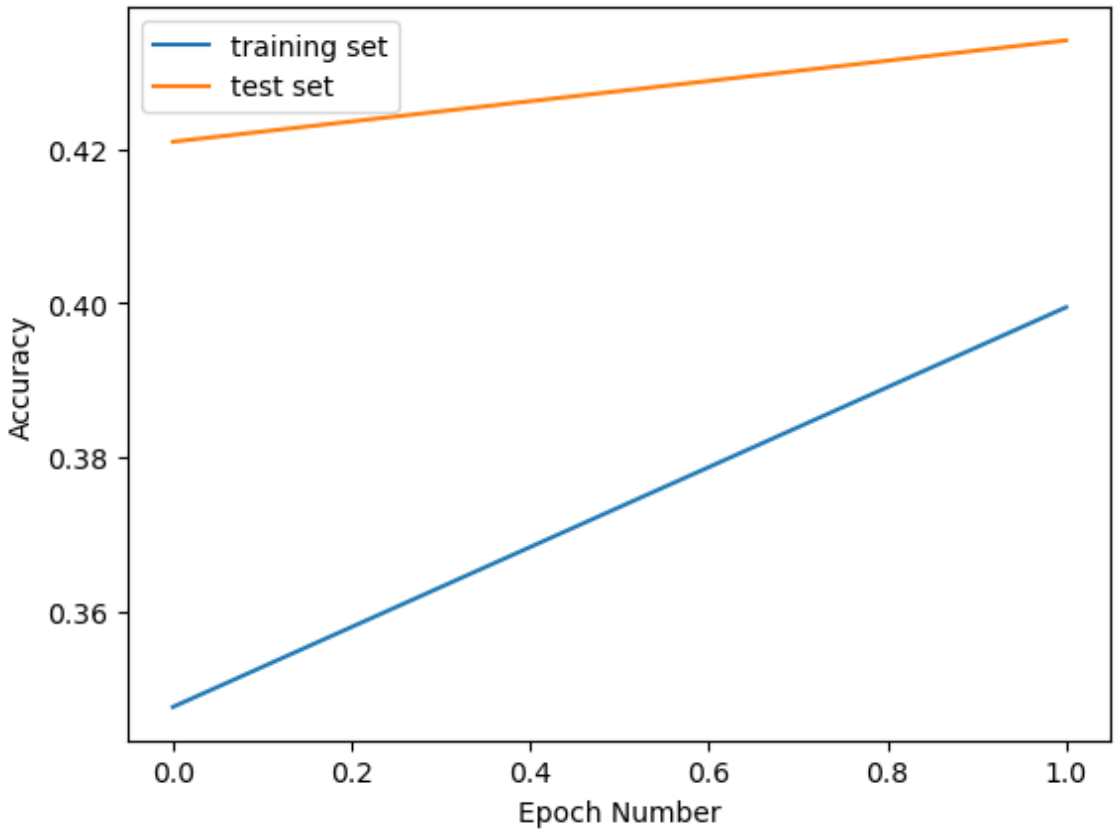
In [13]:
```
plt.xlabel('Epoch Number')
plt.ylabel('Loss')
plt.plot(history.history['loss'], label='training set')
plt.plot(history.history['val_loss'], label='test set')
plt.legend()
```

Out[13]: <matplotlib.legend.Legend at 0x2d3e2287290>



In [14]:
```
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[14]:
```
<matplotlib.legend.Legend at 0x2d3e1dd1dd0>
```



In [15]:
```
model_name = 'food_recognition_inceptionV3.h5'
model.save(model_name, save_format='h5')
```

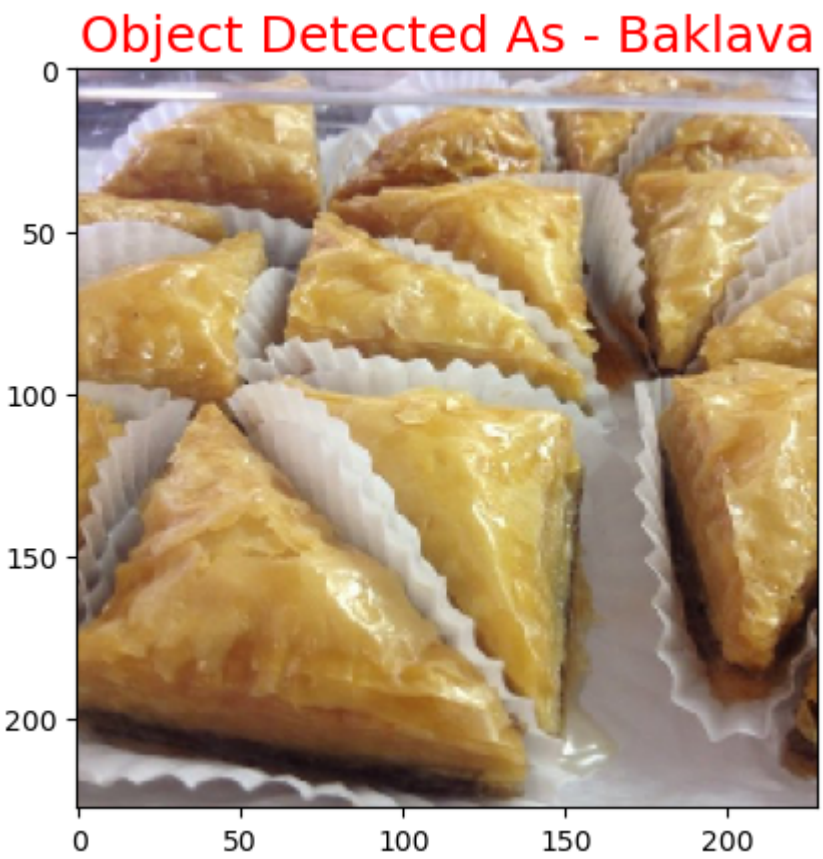
```
C:\Users\HP\anaconda3\Lib\site-packages\keras\src\engine\training.py:3000:
UserWarning: You are saving your model as an HDF5 file via `model.save()`.
This file format is considered legacy. We recommend using instead the nati
ve Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

In [16]:
```
class_map = train_data.class_indices
classes = []
for key in class_map.keys():
    classes.append(key)
```

In [17]:
```
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("Object Detected As - {}".format(str(classes[index]).title())
    plt.imshow(img_array)
```

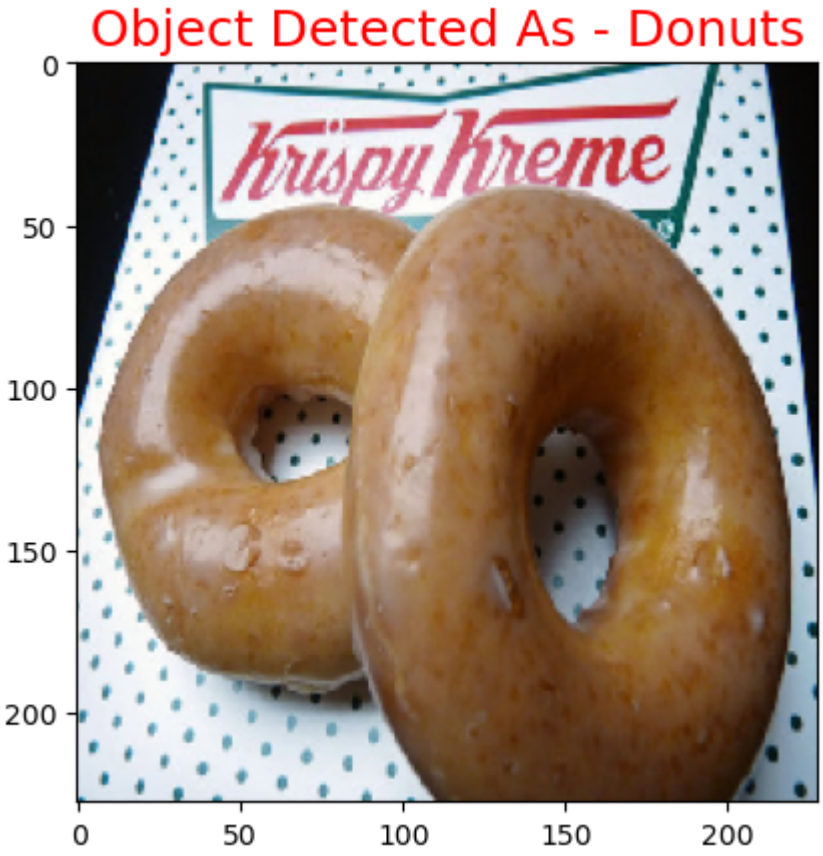
In [19]:
```
predict_image('C:/Users/HP/Desktop/python projects/food-101/images/baklava/
```

```
1/1 [==============================] - 1s 1s/step
```



In [20]:
```
predict_image('C:/Users/HP/Desktop/python projects/food-101/images/donuts/4
```

```
1/1 [==============================] - 0s 58ms/step
```



In [ ]:
```

```