In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import NearestNeighbors
import tensorflow as tf
```

In [ ]:
```
# Load CIFAR-10 dataset
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()

# Resize to target shape for ResNet50 input (64x64 images -> for lower memor
# Use method='bilinear' and antialias=True for better quality resizing
x_train_resized = tf.image.resize(x_train, [32, 32], method='bilinear', anti
x_test_resized = tf.image.resize(x_test, [32, 32], method='bilinear', antial


# Preprocess images for ResNet50
x_train_resized = tf.keras.applications.resnet.preprocess_input(x_train_resi
x_test_resized = tf.keras.applications.resnet.preprocess_input(x_test_resize
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.g
z
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 6s 0us/step
```

In [ ]:
```
# Load ResNet50 model without the top layer for feature extraction
base_model = tf.keras.applications.ResNet50(weights='imagenet', include_top=
model = tf.keras.Model(inputs=base_model.input, outputs=base_model.output)

# Extract features for training images
train_features = model.predict(x_train_resized, batch_size=32)
train_features = train_features.reshape(train_features.shape[0], -1)  # Flat

# Extract features for test images
test_features = model.predict(x_test_resized, batch_size=32)
test_features = test_features.reshape(test_features.shape[0], -1)  # Flatter
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applic
ations/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 17s 8ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step
```

In [ ]:
```
# Fit Nearest Neighbors model
neighbors = NearestNeighbors(n_neighbors=5, metric='cosine')
neighbors.fit(train_features)

# Define a function to retrieve similar images
def retrieve_similar_images(query_index):
    query_feature = test_features[query_index].reshape(1, -1)
    distances, indices = neighbors.kneighbors(query_feature)
    # Plot the query image
    plt.figure(figsize=(10, 3))
    plt.subplot(1, 6, 1)
    plt.imshow(x_test[query_index])
    plt.title("Query Image")
    plt.axis('off')

    # Plot the retrieved images
    for i, idx in enumerate(indices[0]):
        plt.subplot(1, 6, i + 2)
        plt.imshow(x_train[idx])
        plt.title(f"Match {i + 1}")
        plt.axis('off')
    plt.show()
```

In [ ]:

```
# Test retrieval for a random test image
query_index = np.random.randint(0, len(x_test))
retrieve_similar_images(query_index)
```