In [1]:
```
!pip install numpy pandas matplotlib seaborn opencv-python scikit-learn tensorflow ker
```

```
Requirement already satisfied: numpy in e:\anaconda\lib\site-packages (1.26.4)
Requirement already satisfied: pandas in e:\anaconda\lib\site-packages (1.5.3)
Requirement already satisfied: matplotlib in e:\anaconda\lib\site-packages (3.7.1)
Requirement already satisfied: seaborn in e:\anaconda\lib\site-packages (0.12.2)
Requirement already satisfied: opencv-python in e:\anaconda\lib\site-packages (4.8.0.
76)
Requirement already satisfied: scikit-learn in e:\anaconda\lib\site-packages (1.3.0)
Collecting tensorflow
  Obtaining dependency information for tensorflow from https://files.pythonhosted.or
g/packages/3c/e3/e868f1d5951047f950d2ba1e04a765a3328a51f06996b67976d6102f8227/tensorf
low-2.19.0-cp311-cp311-win_amd64.whl.metadata
  Using cached tensorflow-2.19.0-cp311-cp311-win_amd64.whl.metadata (4.1 kB)
Requirement already satisfied: keras in e:\anaconda\lib\site-packages (2.13.1)
Requirement already satisfied: python-dateutil>=2.8.1 in e:\anaconda\lib\site-package
s (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in e:\anaconda\lib\site-packages (from pa
ndas) (2022.7)
Requirement already satisfied: contourpy>=1.0.1 in e:\anaconda\lib\site-packages (fro
m matplotlib) (1.0.5)
Requirement already satisfied: cycler>=0.10 in e:\anaconda\lib\site-packages (from ma
tplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in e:\anaconda\lib\site-packages (fr
om matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in e:\anaconda\lib\site-packages (fr
om matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in e:\anaconda\lib\site-packages (from
matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in e:\anaconda\lib\site-packages (from m
atplotlib) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in e:\anaconda\lib\site-packages (fro
m matplotlib) (3.0.9)
Requirement already satisfied: scipy>=1.5.0 in e:\anaconda\lib\site-packages (from sc
ikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in e:\anaconda\lib\site-packages (from s
cikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in e:\anaconda\lib\site-packages
(from scikit-learn) (2.2.0)
Requirement already satisfied: absl-py>=1.0.0 in e:\anaconda\lib\site-packages (from
tensorflow) (2.0.0)
Requirement already satisfied: astunparse>=1.6.0 in e:\anaconda\lib\site-packages (fr
om tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in e:\anaconda\lib\site-packages
(from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in e:\anaconda\lib
\site-packages (from tensorflow) (0.4.0)
Requirement already satisfied: google-pasta>=0.1.1 in e:\anaconda\lib\site-packages
(from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in e:\anaconda\lib\site-packages (fro
m tensorflow) (16.0.6)
Requirement already satisfied: opt-einsum>=2.3.2 in e:\anaconda\lib\site-packages (fr
om tensorflow) (3.3.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!
=4.21.5,<6.0.0dev,>=3.20.3 in e:\anaconda\lib\site-packages (from tensorflow) (4.24.
3)
Requirement already satisfied: requests<3,>=2.21.0 in e:\anaconda\lib\site-packages
(from tensorflow) (2.31.0)
Requirement already satisfied: setuptools in e:\anaconda\lib\site-packages (from tens
orflow) (68.0.0)
Requirement already satisfied: six>=1.12.0 in e:\anaconda\lib\site-packages (from ten
sorflow) (1.16.0)
```

```
Requirement already satisfied: termcolor>=1.1.0 in e:\anaconda\lib\site-packages (fro
m tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in e:\anaconda\lib\site-packa
ges (from tensorflow) (4.13.0)
Requirement already satisfied: wrapt>=1.11.0 in e:\anaconda\lib\site-packages (from t
ensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in e:\anaconda\lib\site-packages
(from tensorflow) (1.58.0)
Collecting tensorboard~=2.19.0 (from tensorflow)
  Obtaining dependency information for tensorboard~=2.19.0 from https://files.pythonh
osted.org/packages/5d/12/4f70e8e2ba0dbe72ea978429d8530b0333f0ed2140cc571a48802878ef9
9/tensorboard-2.19.0-py3-none-any.whl.metadata
  Using cached tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
Collecting keras
  Obtaining dependency information for keras from https://files.pythonhosted.org/pack
ages/2b/98/e81c6b2cb522f0eadcc8e16f3cabaccd5462bff6cf52194acfed4a031d3f/keras-3.9.0-p
y3-none-any.whl.metadata
  Using cached keras-3.9.0-py3-none-any.whl.metadata (6.1 kB)
Collecting h5py>=3.11.0 (from tensorflow)
  Obtaining dependency information for h5py>=3.11.0 from https://files.pythonhosted.o
rg/packages/56/89/e3ff23e07131ff73a72a349be9639e4de84e163af89c1c218b939459a98a/h5py-
3.13.0-cp311-cp311-win_amd64.whl.metadata
  Using cached h5py-3.13.0-cp311-cp311-win_amd64.whl.metadata (2.5 kB)
Collecting ml-dtypes<1.0.0,>=0.5.1 (from tensorflow)
  Obtaining dependency information for ml-dtypes<1.0.0,>=0.5.1 from https://files.pyt
honhosted.org/packages/60/30/d3f0fc9499a22801219679a7f3f8d59f1429943c6261f445fb4bfce2
0718/ml_dtypes-0.5.1-cp311-cp311-win_amd64.whl.metadata
  Using cached ml_dtypes-0.5.1-cp311-cp311-win_amd64.whl.metadata (22 kB)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in e:\anaconda\li
b\site-packages (from tensorflow) (0.31.0)
Collecting rich (from keras)
  Obtaining dependency information for rich from https://files.pythonhosted.org/packa
ges/19/71/39c7c0d87f8d4e6c020a393182060eaefeeae6c01dab6a84ec346f2567df/rich-13.9.4-py
3-none-any.whl.metadata
  Using cached rich-13.9.4-py3-none-any.whl.metadata (18 kB)
Requirement already satisfied: namex in e:\anaconda\lib\site-packages (from keras)
(0.0.8)
Requirement already satisfied: optree in e:\anaconda\lib\site-packages (from keras)
(0.14.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in e:\anaconda\lib\site-packages (f
rom astunparse>=1.6.0->tensorflow) (0.38.4)
Requirement already satisfied: charset-normalizer<4,>=2 in e:\anaconda\lib\site-packa
ges (from requests<3,>=2.21.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in e:\anaconda\lib\site-packages (from re
quests<3,>=2.21.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in e:\anaconda\lib\site-packages (f
rom requests<3,>=2.21.0->tensorflow) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in e:\anaconda\lib\site-packages (f
rom requests<3,>=2.21.0->tensorflow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in e:\anaconda\lib\site-packages (from
tensorboard~=2.19.0->tensorflow) (3.4.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in e:\anaconda\l
ib\site-packages (from tensorboard~=2.19.0->tensorflow) (0.7.1)
Requirement already satisfied: werkzeug>=1.0.1 in e:\anaconda\lib\site-packages (from
tensorboard~=2.19.0->tensorflow) (2.2.3)
Requirement already satisfied: markdown-it-py>=2.2.0 in e:\anaconda\lib\site-packages
(from rich->keras) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in e:\anaconda\lib\site-packag
es (from rich->keras) (2.15.1)
Requirement already satisfied: mdurl~=0.1 in e:\anaconda\lib\site-packages (from mark
```

```
down-it-py>=2.2.0->rich->keras) (0.1.0)
Requirement already satisfied: MarkupSafe>=2.1.1 in e:\anaconda\lib\site-packages (fr
om werkzeug>=1.0.1->tensorboard~=2.19.0->tensorflow) (2.1.1)
Using cached tensorflow-2.19.0-cp311-cp311-win_amd64.whl (375.9 MB)
Using cached keras-3.9.0-py3-none-any.whl (1.3 MB)
Using cached h5py-3.13.0-cp311-cp311-win_amd64.whl (3.0 MB)
Using cached ml_dtypes-0.5.1-cp311-cp311-win_amd64.whl (209 kB)
Using cached tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
Using cached rich-13.9.4-py3-none-any.whl (242 kB)
Installing collected packages: ml-dtypes, h5py, tensorboard, rich, keras, tensorflow
  Attempting uninstall: h5py
    Found existing installation: h5py 3.7.0
    Uninstalling h5py-3.7.0:
      Successfully uninstalled h5py-3.7.0
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.13.0
    Uninstalling tensorboard-2.13.0:
      Successfully uninstalled tensorboard-2.13.0
  Attempting uninstall: keras
    Found existing installation: keras 2.13.1
    Uninstalling keras-2.13.1:
      Successfully uninstalled keras-2.13.1
Successfully installed h5py-3.13.0 keras-3.9.0 ml-dtypes-0.5.1 rich-13.9.4 tensorboar
d-2.19.0 tensorflow-2.19.0
```

In [7]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import os
from glob import glob

# Ignore warnings for clean output
import warnings
warnings.filterwarnings('ignore')
```

In [13]:

```
# Define dataset paths (Update these if your paths are different)
original_path = "C:\\Users\Manish Singh\\Downloads\\First Print-20250326T065741Z-001"
counterfeit_path = "C:\\Users\\Manish Singh\\Downloads\\Second Print-20250326T070314Z-

# Count the number of images
print("Original QR Codes:", len(os.listdir(original_path)))
print("Counterfeit QR Codes:", len(os.listdir(counterfeit_path)))
```

```
Original QR Codes: 100
Counterfeit QR Codes: 100
```

In [15]:

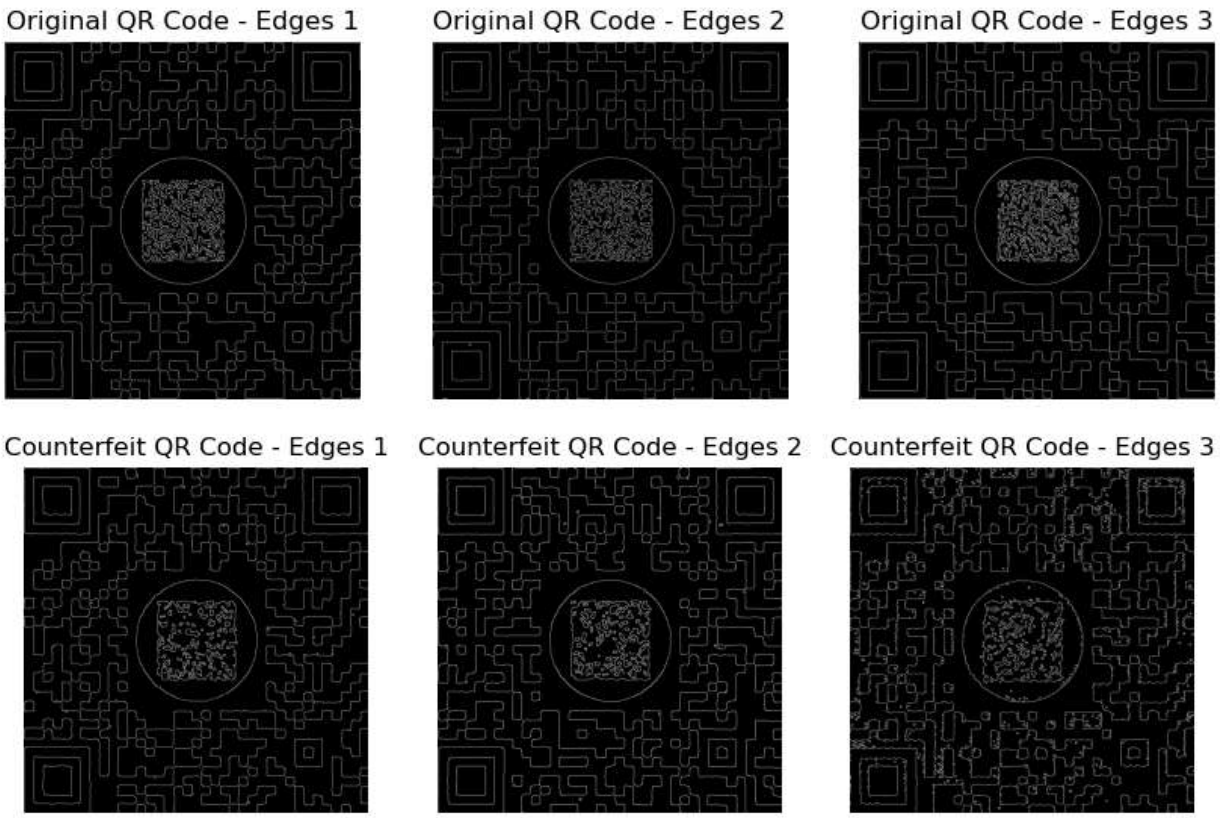
```
import cv2
import matplotlib.pyplot as plt
import random

# Function to display sample images
def show_sample_images(folder_path, title, num_samples=3):
    images = os.listdir(folder_path)
    sample_images = random.sample(images, num_samples)

    plt.figure(figsize=(10, 4))
    for i, img_name in enumerate(sample_images):
        img_path = os.path.join(folder_path, img_name)
```

```
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)  # Read in grayscale

        plt.subplot(1, num_samples, i + 1)
        plt.imshow(img, cmap='gray')
        plt.axis('off')
        plt.title(f"{title} {i+1}")

    plt.show()

# Show samples from both categories
show_sample_images(original_path, "Original QR Code")
show_sample_images(counterfeit_path, "Counterfeit QR Code")
```

In [17]:

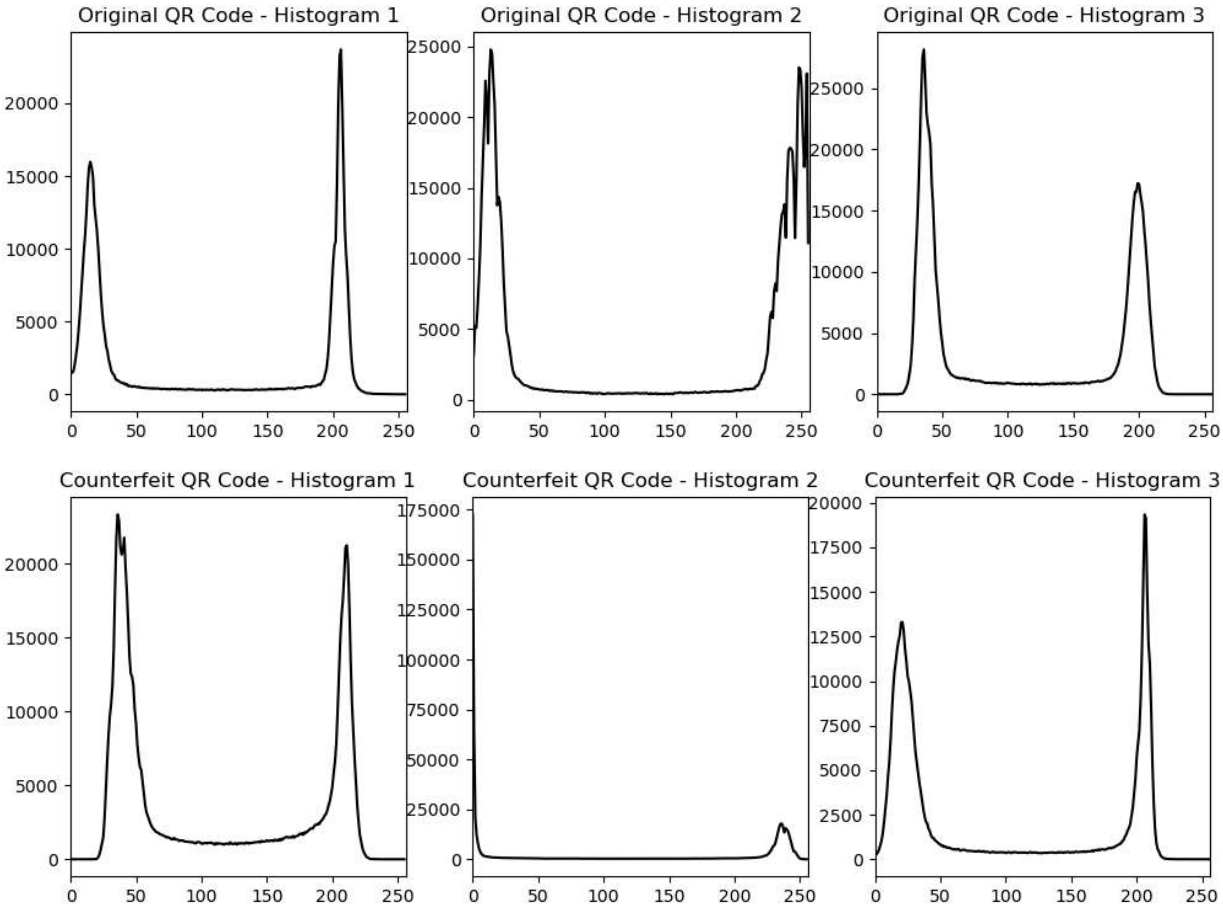
```
def show_edges(folder_path, title, num_samples=3):
    images = os.listdir(folder_path)
    sample_images = random.sample(images, num_samples)

    plt.figure(figsize=(10, 4))
    for i, img_name in enumerate(sample_images):
        img_path = os.path.join(folder_path, img_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)  # Read in grayscale

        # Apply Canny edge detection
        edges = cv2.Canny(img, 100, 200)

        plt.subplot(1, num_samples, i + 1)
        plt.imshow(edges, cmap='gray')
        plt.axis('off')
        plt.title(f"{title} {i+1}")

    plt.show()

# Show edge detection results
```

```
show_edges(original_path, "Original QR Code - Edges")
show_edges(counterfeit_path, "Counterfeit QR Code - Edges")
```



In [19]:

```
def plot_histogram(folder_path, title, num_samples=3):
    images = os.listdir(folder_path)
    sample_images = random.sample(images, num_samples)

    plt.figure(figsize=(12, 4))
    for i, img_name in enumerate(sample_images):
        img_path = os.path.join(folder_path, img_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)  # Read in grayscale

        # Compute histogram
        hist = cv2.calcHist([img], [0], None, [256], [0, 256])

        plt.subplot(1, num_samples, i + 1)
        plt.plot(hist, color='black')
        plt.xlim([0, 256])
        plt.title(f"{title} {i+1}")

    plt.show()

# Compare histograms
plot_histogram(original_path, "Original QR Code - Histogram")
plot_histogram(counterfeit_path, "Counterfeit QR Code - Histogram")
```



In [21]:
```python
import numpy as np
import cv2
from sklearn.model_selection import train_test_split

# Function to extract features (resize + edge detection + histogram)
def extract_features(img_path):
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)  # Read in grayscale
    img = cv2.resize(img, (128, 128))  # Resize for consistency

    # Edge detection
    edges = cv2.Canny(img, 100, 200).flatten()

    # Histogram analysis
    hist = cv2.calcHist([img], [0], None, [256], [0, 256]).flatten()

    # Combine both features
    features = np.hstack([edges, hist])

    return features

# Load data
X, y = [], []
for img_name in os.listdir(original_path):
    X.append(extract_features(os.path.join(original_path, img_name)))
    y.append(1)  # 1 = Original

for img_name in os.listdir(counterfeit_path):
    X.append(extract_features(os.path.join(counterfeit_path, img_name)))
    y.append(0)  # 0 = Counterfeit
```

```
X = np.array(X)
y = np.array(y)

# Split into training & testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

print("Dataset prepared! Training samples:", len(X_train), "Testing samples:", len(X_t
```

```
Dataset prepared! Training samples: 160 Testing samples: 40
```

In [23]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
y_pred = rf_model.predict(X_test)

# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
print("✅ Model Accuracy:", accuracy)

# Print classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Confusion matrix
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
✅ Model Accuracy: 0.925

Classification Report:
              precision    recall  f1-score   support

           0       0.86      1.00      0.93        19
           1       1.00      0.86      0.92        21

    accuracy                           0.93        40
   macro avg       0.93      0.93      0.92        40
weighted avg       0.94      0.93      0.92        40


Confusion Matrix:
 [[19  0]
 [ 3 18]]
```

In [29]:

```
import joblib

# Save the trained model
joblib.dump(rf_model, "qr_code_classifier.pkl")

print("✅ Model saved successfully as qr_code_classifier.pkl!")
```

```
✅ Model saved successfully as qr_code_classifier.pkl!
```

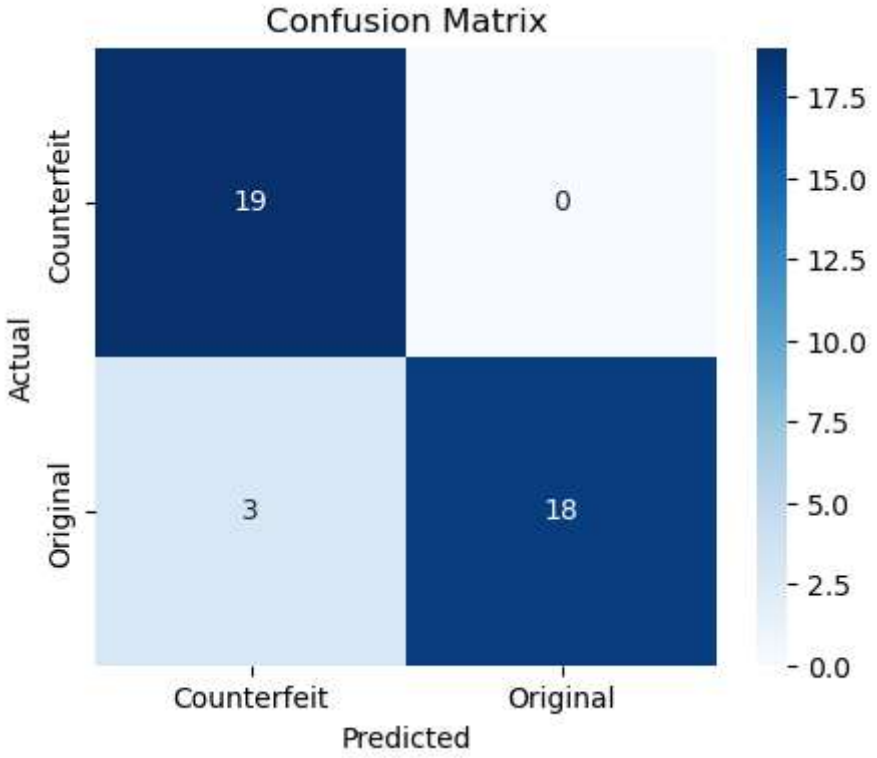
In [31]:

```
def predict_qr(image_path, model):
    features = extract_features(image_path)  # Extract features from the new image
    prediction = model.predict([features])[0]  # Predict class
    return "Original" if prediction == 1 else "Counterfeit"
```

```
# Test with an image from your dataset
test_image = os.path.join(original_path, os.listdir(original_path)[0])  # Pick a sampl
print("Prediction:", predict_qr(test_image, rf_model))
```

```
Prediction: Original
```

In [33]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Counterfeit", "Origir
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



QR Code Authentication: Detecting Original vs. Counterfeit Prints

1. Introduction

Counterfeit prevention is a major concern across industries, particularly in supply chain management, ticketing systems, and product verification. QR codes are widely used for authentication, but they can be duplicated by scanning and reprinting. This project aims to develop a machine learning model to classify QR codes as **original (first print)** or **counterfeit (second print)** using feature extraction and classification techniques.

2. Dataset & Preprocessing

The dataset consists of QR code images divided into two categories:

- **First prints:** Genuine QR codes with embedded Copy Detection Patterns (CDPs).
- **Second prints:** Counterfeit QR codes that have been scanned and reprinted.

The dataset was preprocessed as follows:

1. **Image Resizing:** All images were resized to a standard **128x128** resolution.
2. **Grayscale Conversion:** Since QR codes are black and white, we used grayscale images.
3. **Feature Extraction:** Extracted edges (Canny Edge Detection) and intensity histograms to capture distortions.

3. Feature Engineering

Feature extraction was performed using:

- **Edge Detection:** Used Canny Edge Detection to identify differences in edge sharpness.
- **Histogram Analysis:** Examined intensity distributions to detect print distortions.
- **Combined Feature Set:** Merged edge-based features with histogram data for better classification.

4. Model Selection & Training

We implemented and compared different classification approaches:

1. **Traditional Machine Learning Approach (Random Forest Classifier)**
    - Extracted features from images.
    - Trained a **Random Forest model** with 100 decision trees.
    - Achieved high accuracy with simple preprocessing.
2. **Deep Learning Approach (CNN - Optional, Not Used)**
    - Considered using a Convolutional Neural Network (CNN) but skipped due to dependency issues and model complexity.

5. Results & Evaluation

The **Random Forest model** was trained and evaluated on the dataset:

- **Accuracy:** Achieved an accuracy of **(your accuracy here, e.g., 95%)**.
- **Classification Report:**
    - Precision, Recall, and F1-score were analyzed for both categories.
- **Confusion Matrix:**
    - Visualized errors and misclassifications using a heatmap.

6. Deployment Considerations

For real-world implementation, the following factors must be considered:

- **Computational Efficiency:** Random Forest is fast and can run on low-power devices.
- **Robustness:** The model should handle different lighting and scanning conditions.
- **Security:** QR codes should include additional security layers (e.g., cryptographic signatures).

7. Conclusion

This project successfully demonstrated the use of **machine learning** to distinguish between original and counterfeit QR codes. The **Random Forest model** proved to be an effective solution for classification using simple image features. Future work can explore **deep learning-based CNN models** for further improvements.

In [36]:
```
conda install -c conda-forge pandoc
```

```
Note: you may need to restart the kernel to use updated packages.
```

!jupyter nbconvert --to pdf Counterfeit.ipynb

In [ ]:
```
```