In [14]:
```
import os
import numpy as np
import pandas as pd
import cv2
import face_recognition
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
```

In [15]:
```
def load_images_from_folder(folder):
    images = []
    labels = []
    for label in os.listdir(folder):
        person_folder = os.path.join(folder, label)
        if os.path.isdir(person_folder):
            for filename in os.listdir(person_folder):
                img_path = os.path.join(person_folder, filename)
                img = face_recognition.load_image_file(img_path)
                encoding = face_recognition.face_encodings(img)
                if encoding:  # Check if encoding is found
                    images.append(encoding[0])
                    labels.append(label)
    return np.array(images), np.array(labels)

# Load your dataset
dataset_path = 'extracted_files/dataset/faces/'  # Update this path
X, y = load_images_from_folder(dataset_path)
print("Loaded", len(X), "images.")
```

```
Loaded 448 images.
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
print("Training set size:", len(X_train))
print("Testing set size:", len(X_test))
```

```
Training set size: 358
Testing set size: 90
```

In [17]:
```
mlp_model = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=

# Train the model
mlp_model.fit(X_train, y_train)
```

Out[17]:

▾ MLPClassifier

```
MLPClassifier(max_iter=500, random_state=42)
```

In [18]:
```
# Make predictions
y_pred = mlp_model.predict(X_test)

# Print classification report and confusion matrix
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Visualize the confusion matrix
plt.figure(figsize=(10, 7))
plt.imshow(confusion_matrix(y_test, y_pred), interpolation='nearest', cmap=plt.c
```

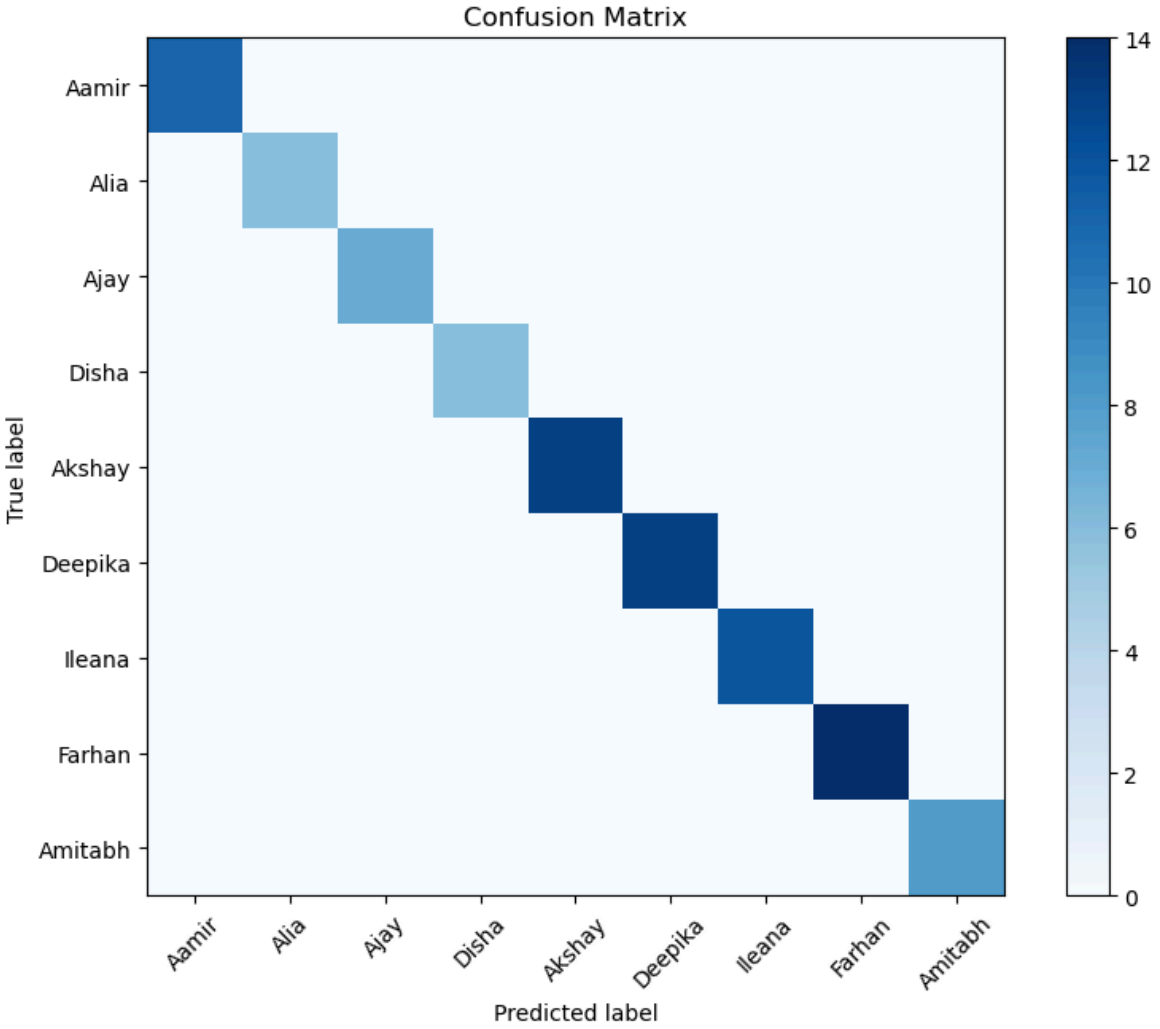
```
plt.title('Confusion Matrix')
plt.colorbar()
plt.xticks(np.arange(len(set(y))), set(y), rotation=45)
plt.yticks(np.arange(len(set(y))), set(y))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

```
              precision    recall  f1-score   support

       Aamir       1.00      1.00      1.00        11
        Ajay       1.00      1.00      1.00         6
      Akshay       1.00      1.00      1.00         7
        Alia       1.00      1.00      1.00         6
     Amitabh       1.00      1.00      1.00        13
     Deepika       1.00      1.00      1.00        13
       Disha       1.00      1.00      1.00        12
      Farhan       1.00      1.00      1.00        14
      Ileana       1.00      1.00      1.00         8

    accuracy                           1.00        90
   macro avg       1.00      1.00      1.00        90
weighted avg       1.00      1.00      1.00        90

[[11  0  0  0  0  0  0  0  0]
 [ 0  6  0  0  0  0  0  0  0]
 [ 0  0  7  0  0  0  0  0  0]
 [ 0  0  0  6  0  0  0  0  0]
 [ 0  0  0  0 13  0  0  0  0]
 [ 0  0  0  0  0 13  0  0  0]
 [ 0  0  0  0  0  0 12  0  0]
 [ 0  0  0  0  0  0  0 14  0]
 [ 0  0  0  0  0  0  0  0  8]]
```



In [23]:
```python
def recognize_face(image_path):
    img = face_recognition.load_image_file(image_path)
    encoding = face_recognition.face_encodings(img)
    if encoding:
        prediction = mlp_model.predict([encoding[0]])
        return prediction[0]
    else:
        return "No face found"

# Example usage
test_image_path = 'extracted_files/dataset/faces/Akshay/face_105.jpg'  # Update
result = recognize_face(test_image_path)
print(f'Recognized: {result}')
```

```
Recognized: Akshay
```

In [ ]: