Building a Virtual Assistant Using Deep Learning and LLM to Enhance the Trading Experience

Module 1: Voice Recognition Using Deep Learning

In [3]:

```
# Installing the packages in the requirements file
!pip install -r requirements.txt
```

```
Collecting mysql-connector-python (from -r requirements.txt (line 8))
  Downloading mysql_connector_python-9.0.0-cp310-cp310-manylinux_2_17_x86_64.whl.metadata (2.0 kB)
Collecting pymysql (from -r requirements.txt (line 9))
  Downloading PyMySQL-1.1.1-py3-none-any.whl.metadata (4.4 kB)
Collecting sentence-transformers (from -r requirements.txt (line 10))
  Downloading sentence_transformers-3.0.1-py3-none-any.whl.metadata (10 kB)
Collecting chromadb (from -r requirements.txt (line 11))
  Downloading chromadb-0.5.5-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.284->-r requirements.txt
(line 1)) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.284->-r requiremen
ts.txt (line 1)) (2.0.31)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.284->-r require
ments.txt (line 1)) (3.10.1)
Requirement already satisfied: async-timeout<5.0.0,>=4.0.0 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.284->-r r
equirements.txt (line 1)) (4.0.3)
Collecting dataclasses-json<0.6.0,>=0.5.7 (from langchain==0.0.284->-r requirements.txt (line 1))
  Downloading dataclasses_json-0.5.14-py3-none-any.whl.metadata (22 kB)
Collecting langsmith<0.1.0,>=0.0.21 (from langchain==0.0.284->-r requirements.txt (line 1))
  Downloading langsmith-0.0.92-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: numexpr<3.0.0,>=2.8.4 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.284->-r require
ments.txt (line 1)) (2.10.1)
Requirement already satisfied: numpy<2,>=1 in /usr/local/lib/python3.10/dist-packages (from langchain==0.0.284->-r requirements.txt
(line 1)) (1.26.4)
```

In [ ]:

```
# Installing packages
!pip install git+https://github.com/openai/whisper.git
!pip install ffmpeg
!pip install whisper
!pip install langchain
!pip install google-generativeai
!pip install streamlit
!pip install pyngrok
!pip install langchain_experimental
!pip install mysql-connector-python
```

```
Collecting git+https://github.com/openai/whisper.git
  Cloning https://github.com/openai/whisper.git (https://github.com/openai/whisper.git) to /tmp/pip-req-build-4dopurd4
  Running command git clone --filter=blob:none --quiet https://github.com/openai/whisper.git (https://github.com/openai/whisper.git) /tmp/pip-req-bui
ld-4dopurd4
  Resolved https://github.com/openai/whisper.git (https://github.com/openai/whisper.git) to commit ba3f3cd54b0e5b8ce1ab3de13e32122d0d5f98ab
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from openai-whisper==20231117) (0.60.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from openai-whisper==20231117) (1.26.4)
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from openai-whisper==20231117) (2.3.1+cu121)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from openai-whisper==20231117) (4.66.5)
Requirement already satisfied: more-itertools in /usr/local/lib/python3.10/dist-packages (from openai-whisper==20231117) (10.3.0)
Requirement already satisfied: tiktoken in /usr/local/lib/python3.10/dist-packages (from openai-whisper==20231117) (0.4.0)
Requirement already satisfied: triton<3,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from openai-whisper==20231117) (2.3.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from triton<3,>=2.0.0->openai-whisper==20231117)
(3.15.4)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba->openai-whisper==20
231117) (0.43.0)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken->openai-whisper==20231117)
(2024.5.15)
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.10/dist-packages (from tiktoken->openai-whisper==20231117)
(2.32.3)
```

In [1]:

```python
# Installing the correct version of TensorFlow
!pip install tensorflow==2.10
```

```
Collecting tensorflow==2.10
  Downloading tensorflow-2.10.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.1 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.6.3)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (24.3.25)
Collecting gast<=0.4.0,>=0.2.1 (from tensorflow==2.10)
  Downloading gast-0.4.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.64.1)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (3.11.0)
Collecting keras<2.11,>=2.10.0 (from tensorflow==2.10)
  Downloading keras-2.10.0-py2.py3-none-any.whl.metadata (1.3 kB)
Collecting keras-preprocessing>=1.1.1 (from tensorflow==2.10)
  Downloading Keras_Preprocessing-1.1.2-py2.py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (18.1.1)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.26.4)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (24.1)
Collecting protobuf<3.20,>=3.9.2 (from tensorflow==2.10)
  Downloading protobuf-3.19.6-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (787 bytes)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow==2.10) (1.16.0)
Collecting tensorboard<2.11,>=2.10 (from tensorflow==2.10)
  Downloading tensorboard-2.10.1-py3-none-any.whl.metadata (1.9 kB)
```

In [1]:

```python
# Import required Libraries
import tensorflow as tf
import os
import cv2
import imghdr
import numpy as np
import seaborn as sns
import pandas as pd
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt
import librosa
import librosa.display
from IPython.display import Audio
import IPython.display as ipd
import warnings
warnings.filterwarnings('ignore')
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import tensorflow as tf
# Importing libraries for building CNN models
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Activation, Conv2D, MaxPooling2D, Flatten
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, EarlyStopping
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import TensorBoard
import warnings
from joblib import Parallel, delayed
import os
import matplotlib.pyplot as plt
import numpy as np
from skimage.io import imread
from skimage.transform import resize
```

In [2]:

```
# Downloading the Dataset required for Speaker Identification
!wget https://github.com/Jakobovski/free-spoken-digit-dataset/archive/refs/heads/master.zip
!unzip master.zip
```

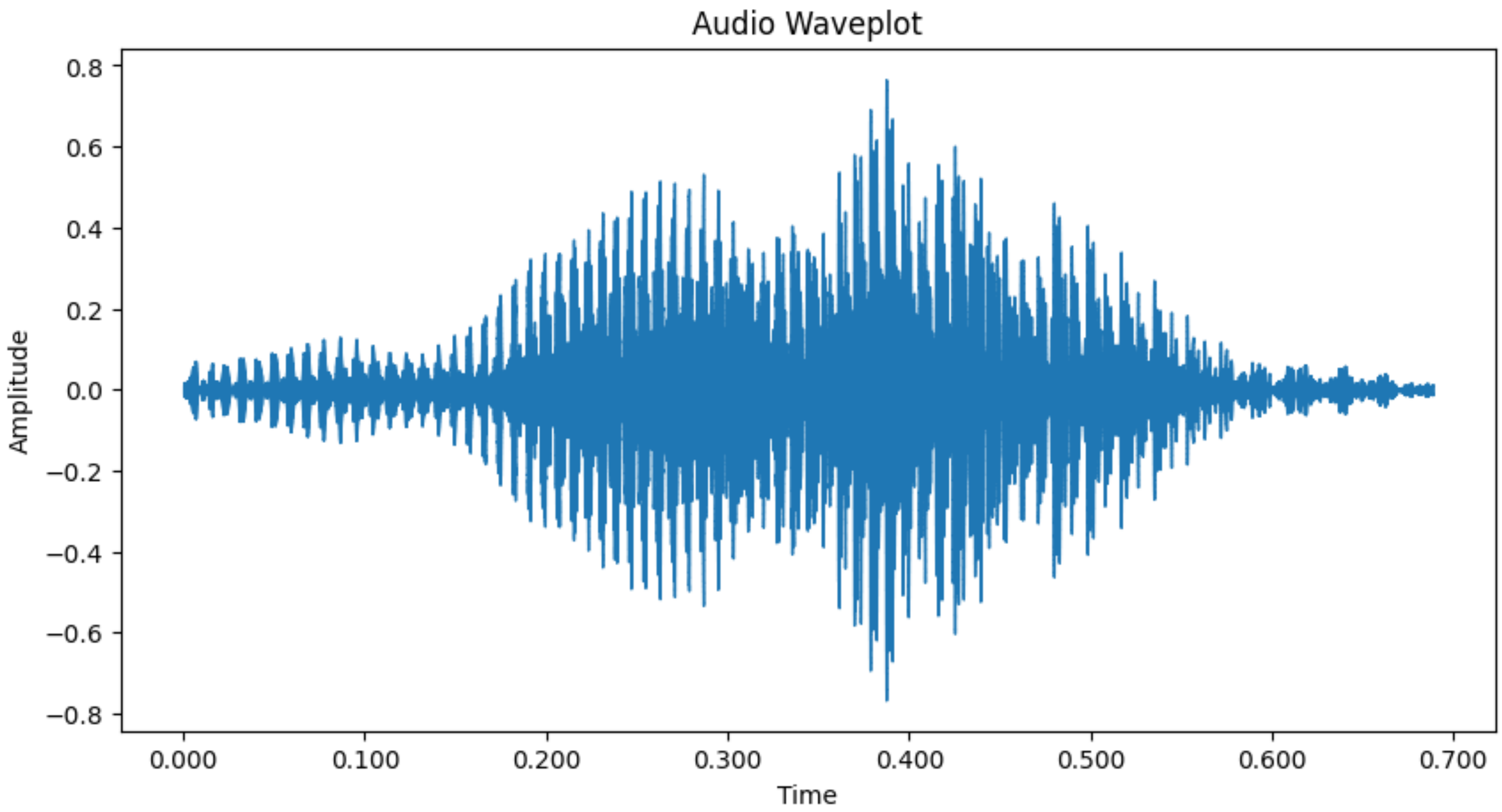
```
--2024-08-07 22:36:20--  https://github.com/Jakobovski/free-spoken-digit-dataset/archive/refs/heads/master.zip (https://github.com/Jakobovski/free-spoken-digit-dataset/archive/refs/heads/master.zip)
Resolving github.com (github.com)... 140.82.116.4
Connecting to github.com (github.com)|140.82.116.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/Jakobovski/free-spoken-digit-dataset/zip/refs/heads/master (https://codeload.github.com/Jakobovski/free-spoken-digit-dataset/zip/refs/heads/master) [following]
--2024-08-07 22:36:20--  https://codeload.github.com/Jakobovski/free-spoken-digit-dataset/zip/refs/heads/master (https://codeload.github.com/Jakobovski/free-spoken-digit-dataset/zip/refs/heads/master)
Resolving codeload.github.com (codeload.github.com)... 140.82.116.10
Connecting to codeload.github.com (codeload.github.com)|140.82.116.10|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/zip]
Saving to: ‘master.zip’

master.zip              [     <=>            ]  15.66M  18.3MB/s    in 0.9s

2024-08-07 22:36:21 (18.3 MB/s) - ‘master.zip’ saved [16422817]

Archive:  master.zip
26eb9aaf76e81b692f806f9140c2d2777410d7a1
   creating: free-spoken-digit-dataset-master/
 extracting: free-spoken-digit-dataset-master/.gitignore
```

In [3]:

```
# Creating a variable for a sample file
sample_filepath='/content/free-spoken-digit-dataset-master/recordings/0_jackson_41.wav'
```

In [4]:

```python
# Plotting Waveplot for the sample audio file
plt.figure(figsize=(10,5))
audio_data,sample_rate = librosa.load(sample_filepath)
librosa.display.waveshow(audio_data,sr=sample_rate)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Audio Waveplot')
plt.show()
ipd.Audio(sample_filepath)
```



0:00 / 0:00

In [5]:

```
# Creating a variable to hold the path for Recordings data
recordings_path="/content/free-spoken-digit-dataset-master/recordings"
```

In [6]:

```python
# Finding the unique set of speakers
import shutil
unique_folder_names = set()
for wav_file in os.listdir(recordings_path):
    name = wav_file.split('_')[1]
    unique_folder_names.add(name)
# Printing  the Unique speaker names
print(unique_folder_names)

# Creating  a directory for each speaker
for folder_name in unique_folder_names:
    shutil.rmtree(folder_name, ignore_errors=True)
    os.makedirs(folder_name, exist_ok=True)
```

```
{'jackson', 'lucas', 'george', 'theo', 'nicolas', 'yweweler'}
```

In [7]:

```python
# Printing the list of Authorized user
authorized_users=["jackson","lucas"]
```

In [8]:

```python
# Copying each file  into Respective folders
for folder_name in unique_folder_names:
  for wav_file in os.listdir(recordings_path):
      name = wav_file.split('_')[1]
      if name == folder_name:
        shutil.copyfile(os.path.join(recordings_path, wav_file), os.path.join(name, wav_file))
```

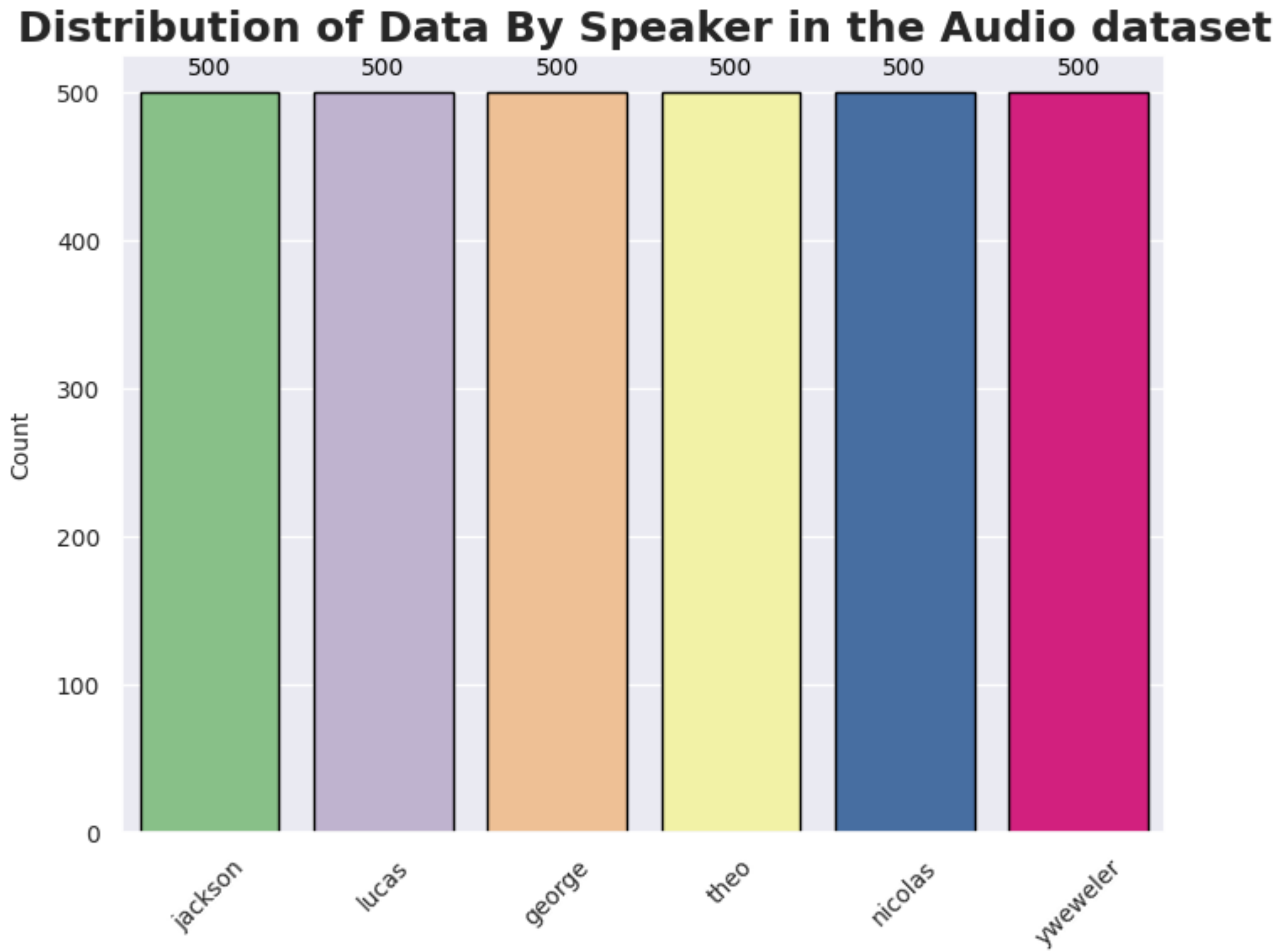
In [9]:

```python
# Counting the number of files available for each speaker
file_count={}
for folders in unique_folder_names:
  file_count[folders]=len(os.listdir(folders))
file_count_df=pd.DataFrame(file_count.items(),columns=['Speaker','File_Count'])
file_count_df
```

	Speaker	File_Count
0	jackson	500
1	lucas	500
2	george	500
3	theo	500
4	nicolas	500
5	yweweler	500

In [10]:

```
# Plotting the Data distribution
plt.figure(figsize=(8, 6))
sns.set(style="darkgrid")
bplot=sns.barplot(x=file_count_df["Speaker"], y=file_count_df["File_Count"], palette='Accent', edge
plt.title('Distribution of Data By Speaker in the Audio dataset', fontsize=18, fontweight='bold')
plt.xlabel('', fontsize=10, labelpad=10)
plt.ylabel('Count', fontsize=10, labelpad=10)
plt.xticks(rotation=45, fontsize=10)
plt.yticks(fontsize=10)
# Add count labels on each bar
for p in bplot.patches:
    bplot.annotate(format(p.get_height(), '.0f'),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha='center', va='center',
                     xytext=(0, 10),
                     textcoords='offset points', fontsize=10, color='black')

plt.show()
```


Distribution of Data By Speaker in the Audio dataset
500
500
500
500
500
500
Count
0
100
200
300
400
500
jackson
lucas
george
theo
nicolas
yweweler

In [11]:

```python
# Creating a combined Directory
combined_base_folder = "combined_dir"
shutil.rmtree(combined_base_folder, ignore_errors=True)
os.makedirs(combined_base_folder, exist_ok=True)
```

In [12]:

```python
# Creating sub folders in the combined dircetory
for folder_name in unique_folder_names:
    shutil.rmtree(os.path.join(combined_base_folder, folder_name), ignore_errors=True)
    os.makedirs(os.path.join(combined_base_folder, folder_name), exist_ok=True)
```

In [13]:

```
import soundfile as sf
# Import numpy for array manipulation
import numpy as np
# Looping through each file and Iteration count
for folder_name in unique_folder_names:
  for file_number in range(50):
    combined_audio_files = []  # Initialize inside the file_number loop
    for wav_file in os.listdir(folder_name):
      # Only selecting the audio files
      if (wav_file.endswith('.wav') and wav_file.split('_')[2].split(".")[0] == str(file_number)):
        # Looping through files for each speaker and Iteration count and combining the files from 6
        for digit in range(10):
          if wav_file.split('_')[0] == str(digit):
            wav_file_path = os.path.join(folder_name, wav_file)
            audio, sample_rate = librosa.load(wav_file_path, sr=None)
            combined_audio_files.append(audio)

    # Check if any audio files were found for this file_number
    if combined_audio_files:
      # Padding the audio files to the same length
      max_length = max([len(audio) for audio in combined_audio_files])
      combined_audio_files = [np.pad(audio, (0, max_length - len(audio))) for audio in combined_auc

      # Stack the audio files into a single NumPy array
      combined_audio_files = np.concatenate(combined_audio_files, axis=0)

      output_file_path = os.path.join(os.path.join(combined_base_folder, folder_name), f"{folder_na
```

```python
        sf.write(output_file_path, combined_audio_files, sample_rate)
    else:
        print(f"No audio files found for folder: {folder_name}, file number: {file_number}")
```

In [14]:

```python
from IPython.display import display, Audio
import os
# Creating a function to play the file
def play_audio(audio_filepath):
    display(Audio(filename=audio_filepath))

# Use os.path.join to create the file path
audio_file = os.path.join("/content/combined_dir/theo", "theo_combined_0.wav")
 # Call the function to play the audio
print(f"Click the play button to listen: {audio_file}")
play_audio(audio_file)
```

```
Click the play button to listen: /content/combined_dir/theo/theo_combined_0.wav
```

0:00 / 0:00

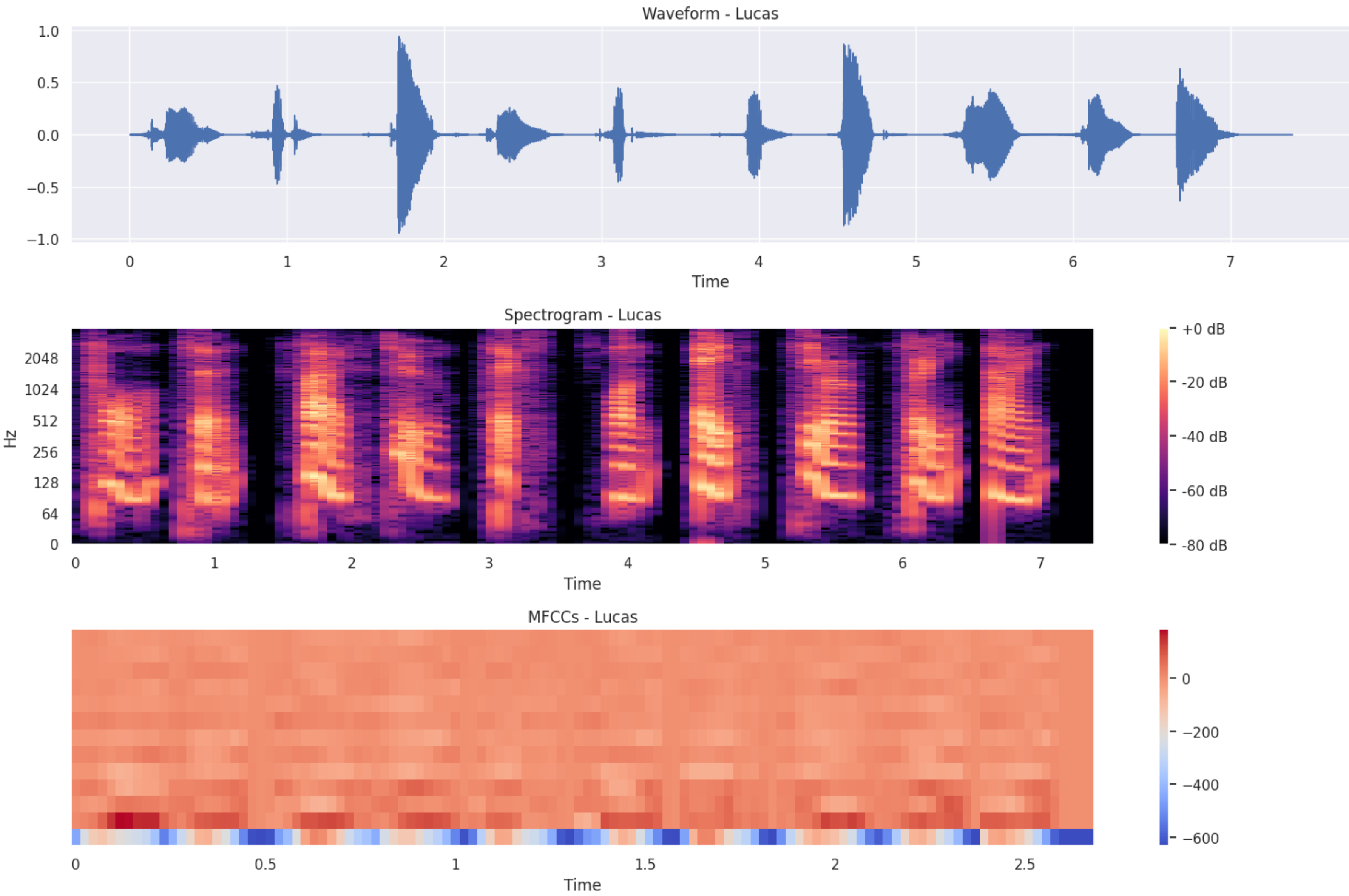
In [15]:

```python
import librosa.display
# Function to plot the waveform, spectrogram, and MFCCs
def audio_plot(audio_filepath, speaker_name):
    # Load audio file
    audio, sample_rate = librosa.load(audio_filepath, sr=None)

    # Plot the waveform
    plt.figure(figsize=(15, 10))
    plt.subplot(3, 1, 1)
    librosa.display.waveshow(audio, sr=sample_rate)
    plt.title(f'Waveform - {speaker_name}')

    # Plot the spectrogram
    plt.subplot(3, 1, 2)
    D = librosa.amplitude_to_db(librosa.stft(audio), ref=np.max)
    librosa.display.specshow(D, sr=sample_rate, x_axis='time', y_axis='log')
    plt.colorbar(format='%+2.0f dB')
    plt.title(f'Spectrogram - {speaker_name}')

    # Plot the MFCCs
    plt.subplot(3, 1, 3)
    mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=13)
    librosa.display.specshow(mfccs, x_axis='time')
    plt.colorbar()
    plt.title(f'MFCCs - {speaker_name}')

    plt.tight_layout()
```

```
plt.show()
```

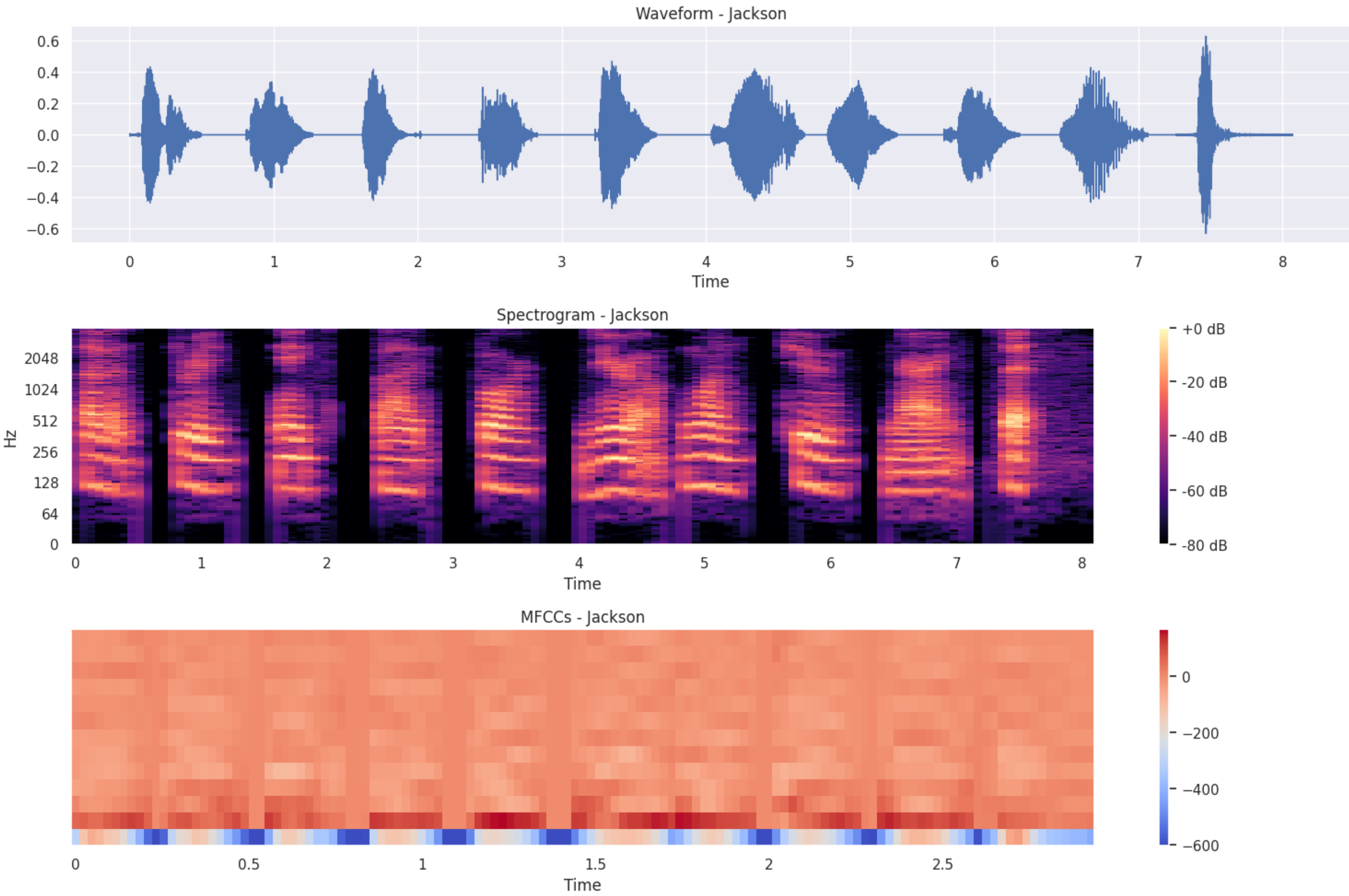
In [16]:

```python
# Plotting various plots for the Speaker Lucas
audio_plot(audio_filepath="/content/combined_dir/lucas/lucas_combined_44.wav", speaker_name="Lucas'
```



In [17]:

```python
# Plotting various plots for the Speaker Jackson
audio_plot(audio_filepath="/content/combined_dir/jackson/jackson_combined_27.wav", speaker_name="Ja
```



In [18]:

```python
# Importing the Libraries
import librosa
import numpy as np
import os
from sklearn.preprocessing import StandardScaler

# Creating a function to extract features from the audio
def extract_features_from_file(audio_filepath):
  # Using Librosa to load the audio file
  audio, sample_rate = librosa.load(audio_filepath, sr=None, duration=1)
  mfcc_audio = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=13)

  # Normalize MFCC features
  mfcc_audio = StandardScaler().fit_transform(mfcc_audio)

  # Pad or truncate MFCCs to a fixed length
  max_length = 16
  if mfcc_audio.shape[1] < max_length:
    mfcc_audio = np.pad(mfcc_audio, ((0, 0), (0, max_length - mfcc_audio.shape[1])), mode='constant
  else:
    mfcc_audio = mfcc_audio[:, :max_length]
  # Taking a transpose of Matrix and returning the array
  mfcc_audio_t=mfcc_audio.T
  return mfcc_audio_t

def extract_audio_features(base_dir):
  # This function will create a numpy array of all files in the specified directory and return a Nu
```

```python
    mfcc_features = []
    speaker_labels = []
    # Looping through each combined file for the speaker
    for speaker_inx, speaker in enumerate(unique_folder_names):
        speaker_folder_path = os.path.join(base_dir, speaker)
        # selecting only the audio files
        for wavefile in os.listdir(speaker_folder_path):
            if wavefile.endswith(".wav"):
                file_path = os.path.join(speaker_folder_path, wavefile)
                mfcc_audio_t=extract_features_from_file(file_path)
                # Creating a final list containing all features extracted from the audio files
                mfcc_features.append(mfcc_audio_t)
                speaker_labels.append(speaker_inx)
    # Returning the final array of features and the labels
    return np.array(mfcc_features), np.array(speaker_labels)

# Extract features and labels
X, y = extract_audio_features(base_dir="combined_dir")
```

In [19]:

```python
# Printing the shape of the array to confirm 300 files, 13 MFCC features
X.shape
```

```
(300, 16, 13)
```

In [20]:

```
# Importing libraries for model building
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.preprocessing import LabelEncoder

# Creating an object for Label encoding
label_encoding = LabelEncoder()
y = label_encoding.fit_transform(y)
# Encoding the target classes
label_encoding.classes_ = np.array(unique_folder_names)

# Split the data into training, validation, and test sets
X_train, X_temporary, y_train, y_temporary = train_test_split(X, y, test_size=0.3, random_state=39
X_val, X_test, y_val, y_test = train_test_split(X_temporary, y_temporary, test_size=0.5, random_st

# Print the shapes of training and validation data
print("Training Data Shape:", X_train.shape)
print("Validation Data Shape:", X_val.shape)
print("Test Data Shape:", X_test.shape)
```

```
Training Data Shape: (210, 16, 13)
Validation Data Shape: (45, 16, 13)
Test Data Shape: (45, 16, 13)
```

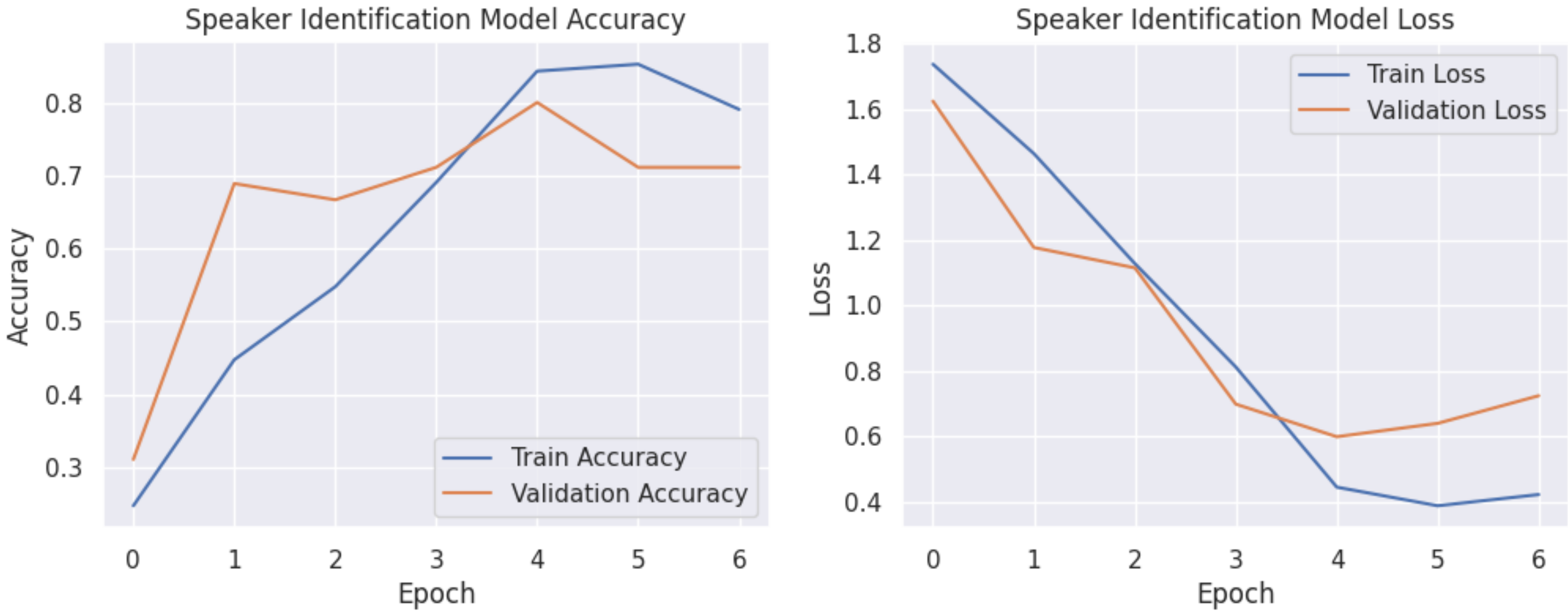
In [21]:

```python
# Define the RNN(LSTM) model
lstm_speaker_model = tf.keras.Sequential([
    tf.keras.layers.LSTM(128, input_shape=(X_train.shape[1], X_train.shape[2])),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(len(unique_folder_names), activation='softmax')
])

# Compile the model
lstm_speaker_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accu

# Define the EarlyStopping callback
earlystop = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)

# Train the model with EarlyStopping
history = lstm_speaker_model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=20, batch

# Check if EarlyStopping triggered
if earlystop.stopped_epoch > 0:
    print("Early stopping triggered at epoch", earlystop.stopped_epoch + 1)
else:
    print("Training completed without early stopping")
```

```
Epoch 1/20
14/14 [==============================] - 11s 176ms/step - loss: 1.7374 - accuracy: 0.2476 - val_loss: 1.6244 - val_accuracy: 0.3111
Epoch 2/20
14/14 [==============================] - 1s 41ms/step - loss: 1.4640 - accuracy: 0.4476 - val_loss: 1.1771 - val_accuracy: 0.6889
Epoch 3/20
14/14 [==============================] - 1s 52ms/step - loss: 1.1278 - accuracy: 0.5476 - val_loss: 1.1147 - val_accuracy: 0.6667
Epoch 4/20
14/14 [==============================] - 0s 25ms/step - loss: 0.8117 - accuracy: 0.6905 - val_loss: 0.6981 - val_accuracy: 0.7111
Epoch 5/20
14/14 [==============================] - 0s 25ms/step - loss: 0.4444 - accuracy: 0.8429 - val_loss: 0.5985 - val_accuracy: 0.8000
Epoch 6/20
14/14 [==============================] - 0s 24ms/step - loss: 0.3875 - accuracy: 0.8524 - val_loss: 0.6394 - val_accuracy: 0.7111
Epoch 7/20
14/14 [==============================] - 0s 25ms/step - loss: 0.4217 - accuracy: 0.7905 - val_loss: 0.7238 - val_accuracy: 0.7111
Early stopping triggered at epoch 7
```

In [22]:

```python
# Plot training & validation accuracy and loss values
plt.figure(figsize=(12, 4))
# Accuracy
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Speaker Identification Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
# Loss
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'],  label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Speaker Identification Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.show()
```



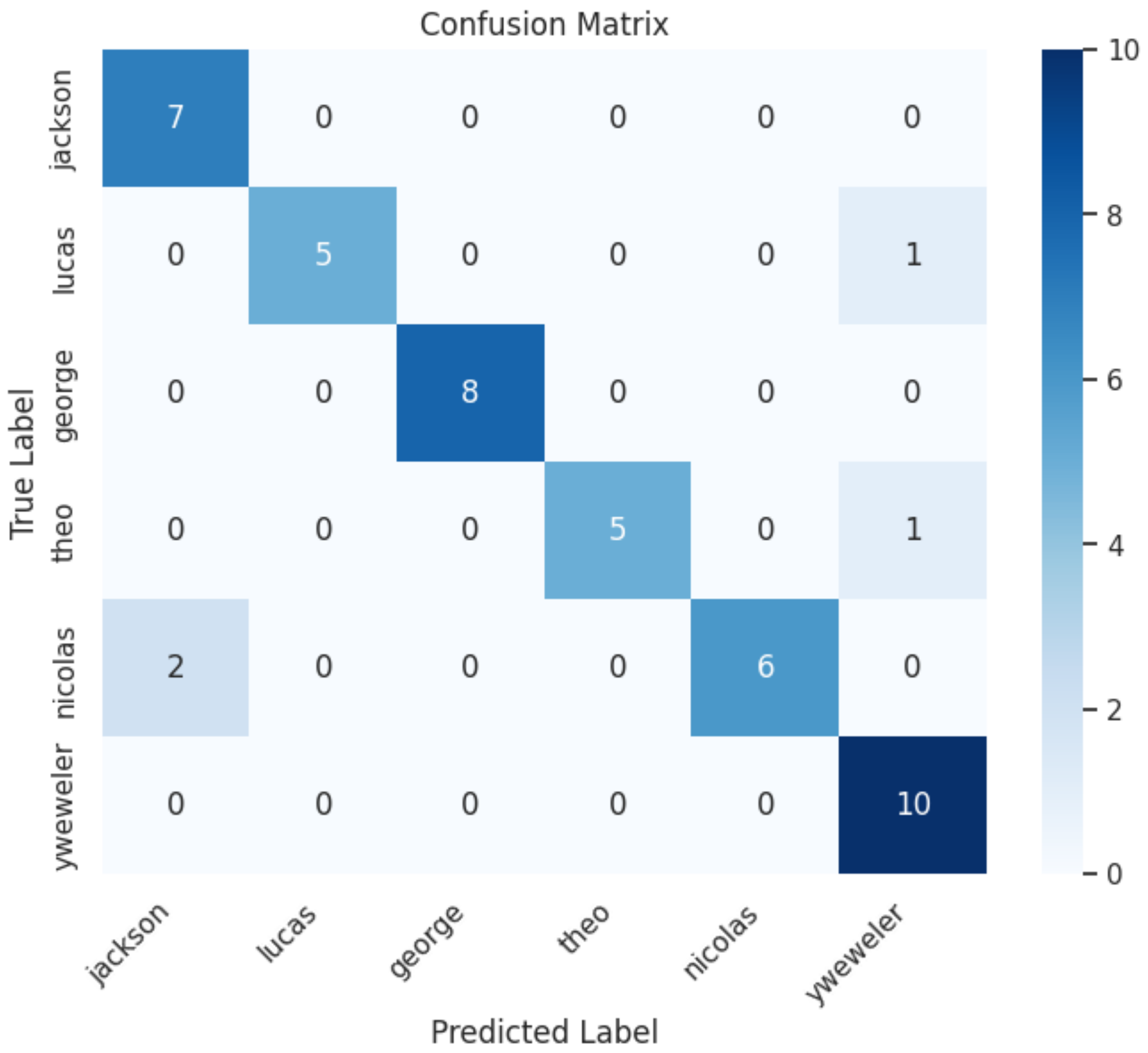
Speaker Identification Model Accuracy
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
Train Accuracy
Validation Accuracy
0
1
2
3
4
5
6
Epoch
Speaker Identification Model Loss
Loss
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
Train Loss
Validation Loss
0
1
2
3
4
5
6
Epoch

In [23]:

```
# Building confusion Matrix
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, f1_score

# Predicting the Results on the test set
y_pred_prob = lstm_speaker_model.predict(X_test)
y_pred = np.argmax(y_pred_prob, axis=1)

# Creating an object for Label encoding
label_encoding = LabelEncoder()
y = label_encoding.fit_transform(y)

# Creating list of classes for Label encoding
label_encoding.classes_ = np.array(list(unique_folder_names))

# Decode labels to original format
y_test_decoded = label_encoding.inverse_transform(y_test)
y_pred_decoded = label_encoding.inverse_transform(y_pred)

# Create a confusion matrix
conf_matrix = confusion_matrix(y_test_decoded, y_pred_decoded, labels=list(unique_folder_names)) #

# Calculate accuracy
accuracy = accuracy_score(y_test_decoded, y_pred_decoded)
print(f"Test Evaluation Accuracy: {accuracy}")
```

```python
# Calculate F1 score
# Convert unique_folder_names to list for f1_score
f1 = f1_score(y_test_decoded, y_pred_decoded, labels=list(unique_folder_names), average='weighted')
print(f"Weighted F1 Score: {f1}")

# Plot the confusion matrix
plt.figure(figsize=(8, 6))
# Convert unique_folder_names to list for heatmap labels
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=list(unique_folder_names),

# Rotate x-axis labels by 45 degrees
plt.xticks(rotation=45, ha="right")

plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
2/2 [==============================] - 1s 13ms/step
Test Evaluation Accuracy: 0.9111111111111111
Weighted F1 Score: 0.9107142857142857
```


Confusion Matrix
True Label
Predicted Label
jackson
lucas
george
theo
nicolas
yweweler
7 0 0 0 0 0
0 5 0 0 0 1
0 0 8 0 0 0
0 0 0 5 0 1
2 0 0 0 6 0
0 0 0 0 0 10
10
8
6
4
2
0

In [24]:

```python
# Creating a function to predict the speaker with the audio file as the input
def predict_speaker(audio_file):
  # Calling the function to call the features
  test_data_t=extract_features_from_file(audio_file)
  test_data=np.expand_dims(test_data_t, axis=0)
  # Predicting the speaker
  test_data_pred_prob = lstm_speaker_model.predict(test_data)
  test_data_pred = np.argmax(test_data_pred_prob, axis=1)
  test_data_label = label_encoding.inverse_transform(test_data_pred)
  actual_speaker=audio_file.split("_")[1]
  print(f"Predicted Speaker: {test_data_label[0]}")
  print(f"Actual Speaker: {actual_speaker}")
  return test_data_label[0]
```

In [25]:

```python
def voice_authenticate_user(audio_file):
  # Creating a function to authenticate the user with audio file as input
  predicted_speaker=predict_speaker(audio_file)
  # Grant access only if the User is an authorized user, else deny
  if predicted_speaker in authorized_users:
    print("Access Granted")
    return True
  else:
    print("Access Denied")
    return False
```

In [26]:

```python
# Testing the audio of the speaker Lucas
test_file_path1 = os.path.join("/content/lucas", "5_lucas_5.wav")
voice_authenticate_flag=voice_authenticate_user(test_file_path1)
print(f"Voice Authentication Flag: {voice_authenticate_flag}")
```

```
1/1 [==============================] - 0s 51ms/step
Predicted Speaker: lucas
Actual Speaker: lucas
Access Granted
Voice Authentication Flag: True
```

In [27]:

```python
# Testing the audio of the speaker George
test_file_path2 = os.path.join("/content/george/", "3_george_45.wav")
voice_authenticate_user(test_file_path2)
```

```
1/1 [==============================] - 0s 60ms/step
Predicted Speaker: george
Actual Speaker: george
Access Denied
```

```
False
```

In [28]:

```
# Testing the audio of the speaker Jackson
test_file_path3 = os.path.join("/content/jackson/", "4_jackson_29.wav")
voice_authenticate_user(test_file_path3)
```

```
1/1 [==============================] - 0s 48ms/step
Predicted Speaker: jackson
Actual Speaker: jackson
Access Granted


True
```

Module 2: Passphrase Identification using Whisper Model

In [29]:

```
 # Importing the Whisper Library
import whisper
whisper_model = whisper.load_model("base")
```

```
100%|██████████████████████████████████████| 139M/139M [00:01<00:00, 77.1MiB/s]
```

In [35]:

```
def extract_text_from_audio(audio_file_path):
  # Extracting the text from the Audio File using Whisper Library
  whisper_extract=whisper_model.transcribe(audio_file_path, fp16=False)
  extracted_text=whisper_extract['text']
  return extracted_text
```

In [36]:

```python
#  Extracting the text from the Audio File using Whisper Library
passphrase_path='/content/passphrase.wav'
passphrase_text=extract_text_from_audio(passphrase_path)
print(f" The Extracted passphrase is : {passphrase_text}")
```

```
The Extracted passphrase is :  blue unicorn
```

In [37]:

```python
passphrase_text=passphrase_text.strip()
```

In [38]:

```python
def passphrase_authenticate_user(passphrase_text):
  # Function to verify if the user is Passphrase authenticated
  if passphrase_text.lower() == r"blue unicorn":
    print("Access Granted")
    return True
  else:
    print("Access Denied")
    return False
```

In [39]:

```python
# Validating the Passphrase authentication flag
passphrase_authenticate_flag= passphrase_authenticate_user(passphrase_text)
print(f"Passphrase Authentication Flag: {passphrase_authenticate_flag}")
```

```
Access Granted
Passphrase Authentication Flag: True
```

Verifying if both Voice Authentication and Passphrase Authentication flags are true.

In [40]:

```
def authenticate_user():
  # This function will check if both Passphrase and Voice are authenticated. It will allow access c
  if passphrase_authenticate_flag and voice_authenticate_flag:
    print("Access Granted")
    return True
  else:
    print("Access Denied")
    return False
```

In [41]:

```
final_authentication_flag=authenticate_user()
print(f"Final Authentication Flag: {final_authentication_flag}")
```

```
Access Granted
Final Authentication Flag: True
```

Module 3: Passphrase Identification using Lip Reading Model

In [42]:

```
mport Libraries for Lip Reading
ort tensorflow as tf
ort os
ort cv2
ort imghdr
ort numpy as np
ort seaborn as sns
ort pandas as pd
ort numpy as np
ort os
ort seaborn as sns
ort matplotlib.pyplot as plt
ort librosa
ort librosa.display
m IPython.display import Audio
ort IPython.display as ipd
ort warnings
nings.filterwarnings('ignore')
m sklearn.preprocessing import StandardScaler, LabelEncoder
m sklearn.model_selection import train_test_split
m sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
ort tensorflow as tf
mporting libraries for building CNN models
m tensorflow.keras.models import Sequential
m tensorflow.keras.layers import LSTM, Dense, Activation, Conv2D, MaxPooling2D, Flatten
m tensorflow.keras.utils import to_categorical
m tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, EarlyStopping
```

```
m tensorflow.keras.models import Sequential
m tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.layers import Conv3D, LSTM, Dense, Dropout, Bidirectional, MaxPool3D, Activation,
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateScheduler
import warnings
from joblib import Parallel, delayed
import os
import numpy as np
from skimage.io import imread
from skimage.transform import resize
from typing import List
import cv2
import imageio
```

In [43]:

```
mporting Video files
et https://spandh.dcs.shef.ac.uk/gridcorpus/s1/video/s1.mpg_vcd.zip
zip s1.mpg_vcd.zip
 s1.mpg_vcd.zip
```

```
--2024-08-07 22:43:08--  https://spandh.dcs.shef.ac.uk/gridcorpus/s1/video/s1.mpg_vcd.zip (https://spandh.dcs.shef.ac.uk/gridcorpus/s1/video/s1.mpg_vcd.zip)
Resolving spandh.dcs.shef.ac.uk (spandh.dcs.shef.ac.uk)... 143.167.8.2
Connecting to spandh.dcs.shef.ac.uk (spandh.dcs.shef.ac.uk)|143.167.8.2|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 422746353 (403M) [application/zip]
Saving to: ‘s1.mpg_vcd.zip’

s1.mpg_vcd.zip      100%[===================>] 403.16M  23.2MB/s    in 19s

2024-08-07 22:43:28 (21.2 MB/s) - ‘s1.mpg_vcd.zip’ saved [422746353/422746353]

Archive:  s1.mpg_vcd.zip
   creating: s1/
  inflating: s1/swio1s.mpg
  inflating: s1/prii9a.mpg
  inflating: s1/sgwp9s.mpg
  inflating: s1/lwws5s.mpg
  inflating: s1/bbal8p.mpg
  inflating: s1/pwwrzp.mpg
  inflating: s1/pwwezn.mpg
  inflating: s1/sgivzn.mpg
  inflating: s1/swwi9s.mpg
  inflating: s1/lgwtzn.mpg
```

In [44]:

```
mporting the Alignment files containing the text
et https://spandh.dcs.shef.ac.uk/gridcorpus/s1/align/s1.tar
r -xvf s1.tar
 s1.tar
```

```
--2024-08-07 22:43:36--  https://spandh.dcs.shef.ac.uk/gridcorpus/s1/align/s1.tar (https://spandh.dcs.shef.ac.uk/gridcorpus/s1/align/s1.tar)
Resolving spandh.dcs.shef.ac.uk (spandh.dcs.shef.ac.uk)... 143.167.8.2
Connecting to spandh.dcs.shef.ac.uk (spandh.dcs.shef.ac.uk)|143.167.8.2|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1034240 (1010K) [application/x-tar]
Saving to: 's1.tar'

s1.tar              100%[===================>]   1010K  1.24MB/s    in 0.8s

2024-08-07 22:43:38 (1.24 MB/s) - 's1.tar' saved [1034240/1034240]

align/
align/bbaf2n.align
align/bbaf3s.align
align/bbaf4p.align
align/bbaf5a.align
align/bbal6n.align
align/bbal7s.align
align/bbal8p.align
align/bbal9a.align
align/bbas1s.align
align/bbas2p.align
align/bbas3a.align
```

In [46]:

```
reating a variable to hold sample video file
ple_video_path="/content/s1/bbaf2n.mpg"
eo_capture = cv2.VideoCapture(sample_video_path)
nt(video_capture.get(cv2.CAP_PROP_FRAME_COUNT))
nt(video_capture.get(cv2.CAP_PROP_FPS))
nt(video_capture.get(cv2.CAP_PROP_FRAME_WIDTH))
nt(video_capture.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

```
75.0
25.0
360.0
288.0
```

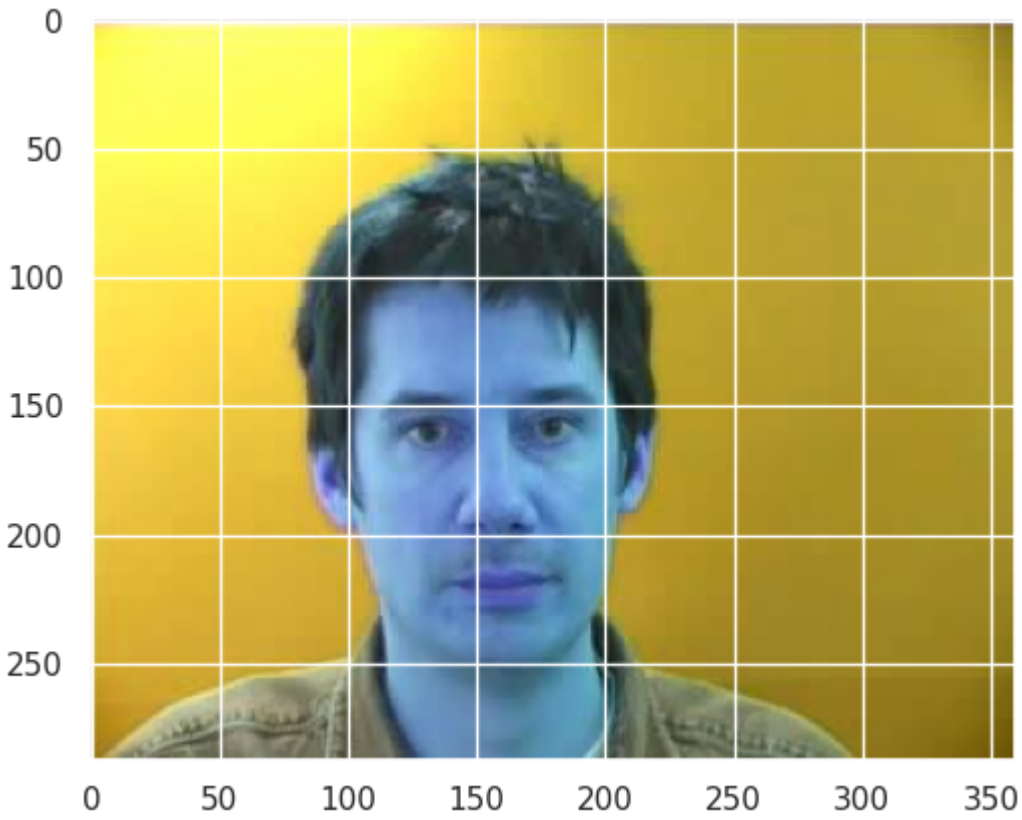
In [47]:

```
rinting the shape of the video frame
ideo_frame=video_capture.read()
nt(video_frame.shape)
```

```
(288, 360, 3)
```

In [48]:

```
lotting the image scanned in the sample video file
geio.imwrite('sample_image_output.jpg', video_frame)
.imshow(imageio.imread('sample_image_output.jpg'))
.show()
```



In [50]:

```
 convert_video_to_frames(video_path):
 The function to scan frame by frame in the video and extract the image and convert into Numpy Array
ideo_capture = cv2.VideoCapture(video_path)
 Creating a list for Storing the Numpy Array
ideo_frames=[]
ideo_frame_count = video_capture.get(cv2.CAP_PROP_FRAME_COUNT)
 Looping through each frame in he video
or frame in range(int(video_frame_count)):
 status, video_frame = video_capture.read()
 # Converting the RGB image to grayscale
 video_frame = tf.image.rgb_to_grayscale(video_frame)
 # Extracting only the mouth portion of the image
 video_frames.append(video_frame[190:236,80:220,:])
 if not status:
   break
ideo_capture.release()
 Standardizing the Numpy array
ean_frames = tf.math.reduce_mean(video_frames)
td_frames = tf.math.reduce_std(tf.cast(video_frames, tf.float32))
tandardized_frames = tf.cast((video_frames - mean_frames), tf.float32) / std_frames
eturn standardized_frames

reating a list of alpha numeric characters to be converted to numbers
hanumeric_list = [alphanum for alphanum in "abcdefghijklmnopqrstuvwxyz'?!123456789 "]
reating a character to number mapping
hanumchar_to_number = tf.keras.layers.StringLookup(vocabulary=alphanumeric_list, oov_token="")
reating a number to character mapping
```

```
ber_to_alphanumchar = tf.keras.layers.StringLookup(vocabulary=alphanumchar_to_number.get_vocabulary(
```

In [51]:

```
aking a sample alignment file and extracting the text from the file
ple_align_path="/content/align/bbaf3s.align"
h open(sample_align_path) as f:
ines = f.readlines()
es = [line.strip() for line in lines]
gnments=[]
canning  through each line in the file and only extracting the text
 line in lines:
f line.split()[2]!="sil":
 print(line.split()[2])
 alignments.append(line.split()[2])
eric_alignments = alphanumchar_to_number(tf.reshape(tf.strings.unicode_split(alignments, input_encod
```

```
bin
blue
at
f
three
soon
```

In [53]:

```
unction to extract the  text from the alignment file
 extract_alignments(alignment_path):
ith open(sample_align_path) as f:
 lines = f.readlines()
ines = [line.strip() for line in lines]
lignments=[]
 Scanning  through each line in the file and only extracting the text
or line in lines:
 if line.split()[2]!="sil":
   alignments.append(line.split()[2])
umeric_alignments = alphanumchar_to_number(tf.reshape(tf.strings.unicode_split(alignments, input_enc
eturn numeric_alignments
```

In [52]:

```
 create_frames_alignments(video_file_path):
""
unction to convert the video files by parsing it frame by frame and
reating a collection of Numpy array. Only the area around the
erson's mouth is used for further processing.
he extracted text from the alignments file will be converted to numbers
""
ideo_file_path = bytes.decode(video_file_path.numpy())
xtracted_annotation=video_file_path.split("/")[-1].split(".")[0]
ideo_file_path=os.path.join("/content/s1",extracted_annotation + ".mpg")
lignment_file_path=os.path.join("/content/align",extracted_annotation + ".align")
td_video_frames=convert_video_to_frames(video_file_path)
um_alignments=extract_alignments(alignment_file_path)
eturn std_video_frames,num_alignments
```

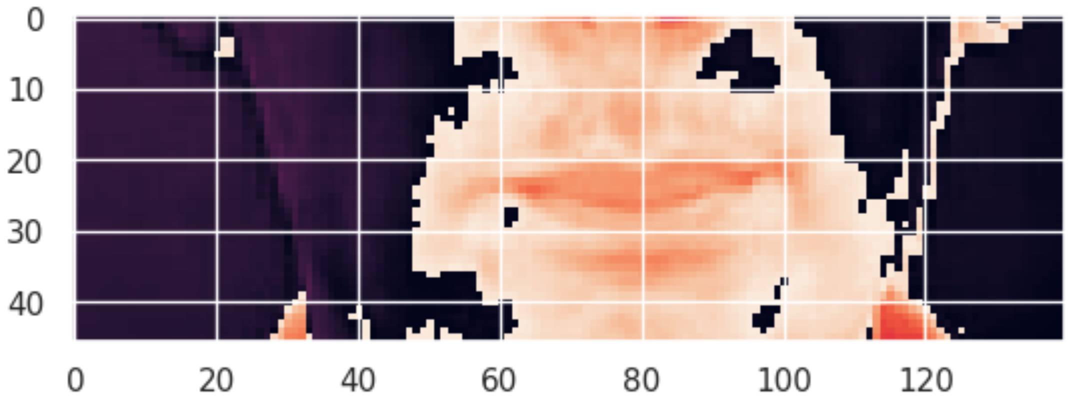
In [54]:

```
alling the function to convert the video and alignments text into Numbers
_video_frames,num_alignments = create_frames_alignments(tf.convert_to_tensor(sample_video_path))
```

In [55]:

```
isplaying the part of the image that will be processed
.imshow(std_video_frames[23])
```

```
<matplotlib.image.AxesImage at 0x7f8a619823e0>
```



In [56]:

```
apply_function(input_video_path):
# Applying the function to videos in the path using Tensor py_function
numeric_result = tf.py_function(create_frames_alignments, [input_video_path], (tf.float32, tf.int64
return numeric_result

electing all files in the path
eo_input = tf.data.Dataset.list_files('/content/s1/*.mpg')
ampling only 100 videos from the path
eo_input = video_input.shuffle(100, reshuffle_each_iteration=False)
pplying the function to all sample videos
eo_input = video_input.map(apply_function)
```

In [57]:

```
adding to include 40 characters in the output.
eo_input = video_input.padded_batch(2, padded_shapes=([75,None,None,None],[40]))
eo_input = video_input.prefetch(tf.data.AUTOTUNE)
plitting the dataset into Train and Test in 80:20 split
inset = video_input.take(80)
tset = video_input.skip(80)

onverting the input into an Iterator
eo_iterator = video_input.as_numpy_iterator()
mes_array = video_iterator.next()
rinting the shape
mes_array[0].shape
```

```
(2, 75, 46, 140, 1)
```

In [58]:

```
reating a model calling the Sequential API
m_model = Sequential()
dding 128 neurons for the first Layer
m_model.add(Conv3D(128, 3, input_shape=(75,46, 140,1), padding='same'))
sing Rectified Linear Unit as Activation
m_model.add(Activation('relu'))
m_model.add(MaxPool3D((1,2,2)))

dding second layer with 256 Neurons
m_model.add(Conv3D(256, 3, padding='same'))
m_model.add(Activation('relu'))
m_model.add(MaxPool3D((1,2,2)))

dding third layer with 75 neurons matching the number of Frames
m_model.add(Conv3D(75, 3, padding='same'))
m_model.add(Activation('relu'))
m_model.add(MaxPool3D((1,2,2)))

lattening the data
m_model.add(TimeDistributed(Flatten()))

dding a Bidirectional LSTM Layer with dropout
m_model.add(Bidirectional(LSTM(128, kernel_initializer='Orthogonal', return_sequences=True)))
m_model.add(Dropout(.5))

m_model.add(Bidirectional(LSTM(128, kernel_initializer='Orthogonal', return_sequences=True)))
m_model.add(Dropout(.5))
```

```
nal Dense Layer with the Softmax as the activation layer
lstm_model.add(Dense(alphanumchar_to_number.vocabulary_size()+1, activation='softmax'))
```

In [59]:

```
rinting the summary
m_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 75, 46, 140, 128)  3584

 activation (Activation)     (None, 75, 46, 140, 128)  0

 max_pooling3d (MaxPooling3D  (None, 75, 23, 70, 128)  0
 )

 conv3d_1 (Conv3D)           (None, 75, 23, 70, 256)   884992

 activation_1 (Activation)   (None, 75, 23, 70, 256)   0

 max_pooling3d_1 (MaxPooling  (None, 75, 11, 35, 256)  0
 3D)

 conv3d_2 (Conv3D)           (None, 75, 11, 35, 75)    518475

 activation_2 (Activation)   (None, 75, 11, 35, 75)    0

 max_pooling3d_2 (MaxPooling  (None, 75, 5, 17, 75)    0
 3D)

 time_distributed (TimeDistr  (None, 75, 6375)         0
 ibuted)

 bidirectional (Bidirectiona  (None, 75, 256)          6660096
 l)

 dropout (Dropout)           (None, 75, 256)           0

 bidirectional_1 (Bidirectio  (None, 75, 256)          394240
 nal)

 dropout_1 (Dropout)         (None, 75, 256)           0

 dense_2 (Dense)             (None, 75, 41)            10537
```

```
=================================================================
Total params: 8,471,924
Trainable params: 8,471,924
Non-trainable params: 0
_________________________________________________________________
```

In [60]:

```
loss_function(real_y, predicted_y):
# Defining the loss function
length_bat = tf.cast(tf.shape(real_y)[0], dtype="int64")
length_inp = tf.cast(tf.shape(predicted_y)[1], dtype="int64")
length_lbl = tf.cast(tf.shape(real_y)[1], dtype="int64")
length_inp = length_inp * tf.ones(shape=(length_bat, 1), dtype="int64")
length_lbl = length_lbl * tf.ones(shape=(length_bat, 1), dtype="int64")
loss = tf.keras.backend.ctc_batch_cost(real_y, predicted_y, length_inp, length_lbl)
return loss

ompliling the Model using Adam Optimizer and Loss Function defined
m_model.compile(optimizer=Adam(learning_rate=0.0001), loss=loss_function)
```

In [61]:

```
efining Call back
lback1 = ModelCheckpoint(os.path.join('lstm_model','chkpoint1.weights.h5'), monitor='loss', save_wei
raining the Model on Train set and Validating on Test set
m_model.fit(trainset, validation_data=testset, epochs=20, callbacks=[callback1])
```

In [75]:

```
reating an iterator of Test set
t_data = testset.as_numpy_iterator()
t_sample = test_data.next()
t = lstm_model.predict(test_sample[0])
```

```
1/1 [==============================] - 18s 18s/step
```

In [76]:

```
rinting the Real Text
.strings.reduce_join([number_to_alphanumchar(wrd) for wrd in sent]) for sent in test_sample[1]]
```

```
[<tf.Tensor: shape=(), dtype=string, numpy=b'inblueatfthreesoon'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'inblueatfthreesoon'>]
```

In [77]:

```
rinting the Predicted Text
oded_value = tf.keras.backend.ctc_decode(yhat, input_length=[75,75], greedy=True)[0][0].numpy()
.strings.reduce_join([number_to_alphanumchar(wrd) for wrd in sent]) for sent in decoded_value]
```

```
[<tf.Tensor: shape=(), dtype=string, numpy=b'bin green with i zero please'>,
 <tf.Tensor: shape=(), dtype=string, numpy=b'bin blue at z six please'>]
```

Module 4: Create Tables and Inserting data into MySQL Database

In [ ]:

```
reating MySQL Table to hold the Portfolio data
ATE TABLE sql5723487.stock_portfolio(
 ticker_symbol VARCHAR(255),
 stock_name VARCHAR(255),
 shares INT,
 current_price_per_share FLOAT,
 purchase_price_per_share FLOAT



reating table to hold Brokerage balance
EATE TABLE sql5723487.brokerage_account (
 balance_amount float



nsert sample data into table
ERT INTO sql5723487.stock_portfolio (ticker_symbol, stock_name,shares,current_price_per_share,purcha
UES
APL', 'Apple' , 20, 180.56, 140.23),
MZN', 'Amazon' , 30, 184.54, 120.53),
SFT', 'Microsoft' , 43, 254.76, 110.66),
SLA', 'Tesla' , 80, 245.76, 40.67),
VDA', 'Nvidia' , 100, 120.67, 23.67),
B', 'Meta' , 40, 180.56, 140.23),
A', 'Boeing' , 50, 180.56, 345.23),
OOG', 'Google' , 40, 180.56, 134.23);

nserting sample data into Account balance table
```

```
ERT INTO sql5723487.brokerage_account(balance_amount)
UES
(1000.00);

commit;

select * from sql5723487.stock_portfolio;
select * from sql5723487.brokerage_account;
```

Module 5: Create Virtual Assistant Using LLM and Langchain

In [78]:

```
mporting the Required Libraries
ort langchain
m langchain.llms import GooglePalm
m langchain.utilities import SQLDatabase
m langchain_experimental.sql import SQLDatabaseChain
m langchain.prompts.prompt import PromptTemplate

ort whisper
ort os
m dotenv import load_dotenv
d_dotenv()
```

```
True
```

In [80]:

```
create_sqlchain():
# Function to create SQL database chain using LLM and LangChain
model_object = GooglePalm(google_api_key=os.environ["GOOGLE_API_KEY"], temperature=0.1)
# Including variables in default template
_DEFAULT_TEMPLATE = """Given an input question, first create a syntactically correct SQL query to r
                    {input}
                    {table_info}
                    {dialect}
                    """


PROMPT = PromptTemplate(
input_variables=["input", "table_info", "dialect"],
template=_DEFAULT_TEMPLATE,
)
# Extracting User name and password from the env file
db_user = os.environ["db_user"]
db_password = os.environ["db_password"]
db_host = os.environ["db_host"]
db_name = os.environ["db_name"]
# creating a database object
db_object = SQLDatabase.from_uri(f"mysql+pymysql://{db_user}:{db_password}@{db_host}/{db_name}",sam
print(db_object.table_info)
database_chain = SQLDatabaseChain.from_llm(model_object, db_object, verbose=True)
return database_chain

abase_chain = create_sqlchain()
```

```
CREATE TABLE brokerage_account (
        balance_amount FLOAT
)ENGINE=InnoDB DEFAULT CHARSET=latin1

/*
3 rows from brokerage_account table:
balance_amount
1000.0
*/


CREATE TABLE stock_portfolio (
        ticker_symbol VARCHAR(255),
        stock_name VARCHAR(255),
        shares INTEGER(11),
        current_price_per_share FLOAT,
        purchase_price_per_share FLOAT
)ENGINE=InnoDB DEFAULT CHARSET=latin1

/*
3 rows from stock_portfolio table:
ticker_symbol   stock_name      shares  current_price_per_share purchase_price_per_share
AAPL    Apple   20      180.56  140.23
AMZN    Amazon  30      184.54  120.53
MSFT    Microsoft       43      254.76  110.66
*/
```

Using Whisper Model to extract the Question asked from Audio files

In [81]:

```
sing Whisper Model to extract the text from Question1
uiry1_path='/content/Question1.wav'
tfolio_qn1=extract_text_from_audio(enquiry1_path)
nt(f" The Question1 is : {portfolio_qn1}")
```

```
The Question1 is :  What is my current balance amount?
```

In [82]:

```
sing Whisper Model to extract the text from Question2
uiry2_path='/content/Question2.wav'
tfolio_qn2=extract_text_from_audio(enquiry2_path)
nt(f" The Question2 is : {portfolio_qn2}")
```

```
The Question2 is :  How many shares of Apple stock are in my portfolio?
```

In [83]:

```
sing Whisper Model to extract the text from Question3
uiry3_path='/content/Question3.wav'
tfolio_qn3=extract_text_from_audio(enquiry3_path)
nt(f" The Question3 is : {portfolio_qn3}")
```

```
The Question3 is :  The what is the current price of Nvidia stock?
```

In [84]:

```
sing Whisper Model to extract the text from Question4
uiry4_path='/content/Question4.wav'
tfolio_qn4=extract_text_from_audio(enquiry4_path)
nt(f" The Question4 is : {portfolio_qn4}")
```

```
The Question4 is :  What is the best performing stock in my portfolio?
```

In [85]:

```
sing Whisper Model to extract the text from Question5
uiry5_path='/content/Question5.wav'
tfolio_qn5=extract_text_from_audio(enquiry5_path)
nt(f" The Question5 is : {portfolio_qn5}")
```

```
The Question5 is :  Do I have sufficient balance in my brokerage account to purchase five shares of Apple's stock?
```

In [86]:

```
alling the SQL Database chain on the Question #1 extracted using the Whisper Library
ck_enquiry1 =database_chain.run(portfolio_qn1)

alling the SQL Database chain on the Question #2 extracted using the Whisper Library
ck_enquiry2 =database_chain.run(portfolio_qn2)

alling the SQL Database chain on the Question #3 extracted using the Whisper Library
ck_enquiry3 =database_chain.run(portfolio_qn3)

alling the SQL Database chain on the Question #4 extracted using the Whisper Library
ck_enquiry4 =database_chain.run(portfolio_qn4)

alling the SQL Database chain on the Question #5 extracted using the Whisper Library
ck_enquiry5 =database_chain.run(portfolio_qn5)
```

```
> Entering new SQLDatabaseChain chain...
 What is my current balance amount?
SQLQuery:SELECT balance_amount FROM brokerage_account
SQLResult: [(1000.0,)]
Answer:1000.0
> Finished chain.


> Entering new SQLDatabaseChain chain...
 How many shares of Apple stock are in my portfolio?
SQLQuery:SELECT shares FROM stock_portfolio WHERE stock_name = 'Apple'
SQLResult: [(20,)]
Answer:20
> Finished chain.


> Entering new SQLDatabaseChain chain...
 The what is the current price of Nvidia stock?
SQLQuery:SELECT current_price_per_share FROM stock_portfolio WHERE ticker_symbol = 'NVDA'
SQLResult: [(120.67,)]
Answer:120.67
> Finished chain.


> Entering new SQLDatabaseChain chain...
 What is the best performing stock in my portfolio?
SQLQuery:SELECT stock_name FROM stock_portfolio ORDER BY (current_price_per_share - purchase_price_per_share) DESC LIMIT 1
SQLResult: [('Tesla',)]
Answer:Tesla
> Finished chain.


> Entering new SQLDatabaseChain chain...
 Do I have sufficient balance in my brokerage account to purchase five shares of Apple's stock?
SQLQuery:SELECT balance_amount FROM brokerage_account WHERE balance_amount >= 5 * (SELECT current_price_per_share FROM stock_portfol
io WHERE ticker_symbol = 'AAPL')
SQLResult: [(1000.0,)]
```

```
Answer:Yes
> Finished chain.
```

Module 6: Streamlit Demo

In [87]:

```
reating a demo.py file
ritefile streamlit_demo.py
mporting the Libraries
ort langchain
m langchain.llms import GooglePalm
m langchain.utilities import SQLDatabase
m langchain_experimental.sql import SQLDatabaseChain
m langchain.prompts.prompt import PromptTemplate
ort streamlit as st
ort whisper
ort os
m dotenv import load_dotenv
d_dotenv()

 create_sqlchain():
 # Function to create SQL database chain using LLM and LangChain
 model_object = GooglePalm(google_api_key=os.environ["GOOGLE_API_KEY"], temperature=0.1)
 # Including variables in default template
 _DEFAULT_TEMPLATE = """Given an input question, first create a syntactically correct SQL query to r
                     {input}
                     {table_info}
                     {dialect}
                     """

 PROMPT = PromptTemplate(
 input_variables=["input", "table_info", "dialect"],
 template=_DEFAULT_TEMPLATE,
```

```
        )
        # Extracting User name and password from the env file
        db_user = os.environ["db_user"]
        db_password = os.environ["db_password"]
        db_host = os.environ["db_host"]
        db_name = os.environ["db_name"]
        # creating a database object
        db_object = SQLDatabase.from_uri(f"mysql+pymysql://{db_user}:{db_password}@{db_host}/{db_name}",sam
        print(db_object.table_info)
        database_chain = SQLDatabaseChain.from_llm(model_object, db_object, verbose=True)
        return database_chain

    # Creating Database Chain
    database_chain=create_sqlchain()
    # Creating the Template
    st.title("Hi! its Alex :man: Your Virtual Assistant! Ask me any question about your stock portfolio :do
    question = st.text_input("Question: ")
    # Question is asked, invoke the SQL Database chain
    if question:
        chain = create_sqlchain()
        response = chain.run(question)
        st.header("Answer")
        st.write(response)
```

```
Writing streamlit_demo.py
```

In [88]:

```
reating a Tunnel using ngrok to show the Streamlit Demo
m pyngrok import ngrok
rok authtoken ngrok_auth_token
_ipython().system('nohup streamlit run streamlit_demo.py &')
nel = ngrok.connect(8501, proto="http")
 = tunnel.public_url
nt(url)
```

```
Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml
nohup: appending output to 'nohup.out'
```

https://3630-104-196-239-104.ngrok-free.app (https://3630-104-196-239-104.ngrok-free.app)