Quality Management Systems for Airline Customer Service Call Centers

Introduction:

The task is to create a cutting-edge web platform tailored for enhancing Quality Management Systems (QMS) within airline customer service call centers. This innovative system harnesses the power of Generative AI, utilizing the Lyzr AI API to analyze call recordings, transcribe conversations, and extract valuable insights. By computing a range of Key Performance Indicators (KPIs), the application aims to assess call quality and agent performance, revolutionizing the way airline call centers optimize their operations and customer interactions.

Project Setup :

1.Extracting data from zip file :

```
def extract_audio_files(zip_file_path):
        audio_files=[]
        # Extracting audio files from zip file and storing it in temp_audio folder
        os.makedirs("temp_audio", exist_ok=True)
        with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        for file_name in zip_ref.namelist():
        if file_name.endswith('.mp3'):
        zip_ref.extract(file_name,"temp_audio")
        audio_files.append(os.path.join("temp_audio",file_name))
    return audio_files
```

This function extracts MP3 audio files from a specified ZIP file path, storing them in a temporary directory named "temp_audio". It then returns a list containing the paths to these extracted audio files for further processing within the application.

2.Transcribing the audio:

```
        def transcribe_audio_from_zip(zip_file_path,api_key):
                # Initializing AssemblyAI through API KEY
                aai.settings.api_key=api_key
                transcriber=aai.Transcriber()
          # Extracting audio files from the zip file
          audio_files=extract_audio_files(zip_file_path)
         transcriptions=[]
         # Transcribing each audio file
        for audio_file in audio_files:
            try:
               transcript=transcriber.transcribe(audio_file)
               transcriptions.append(transcript.text)
          except Exception as e:
            print(f"Error transcribing {audio_file}: {e}")
return transcriptions
```

This function transcribes audio files from a provided ZIP file path using the AssemblyAI service, authenticated with the given API key. It first extracts audio files from the ZIP file, then iterates through each file, transcribing its contents. Any encountered errors during transcription are printed, and the function returns a list of transcriptions for further use.

3.Sentimental Analysis:

```
#determining sentiment score
def analyze_sentiment(transcription):
blob=TextBlob(transcription)
sentiment_score=blob.sentiment.polarity
if sentiment_score > 0:
return "POSITIVE"
elif sentiment_score < 0:
return "NEGATIVE"
else:
return "NEUTRAL"
```

This function analyzes the sentiment of a given transcription using TextBlob, a natural language processing library. It calculates the sentiment polarity score of the transcription and categorizes it as "POSITIVE" if the score is positive, "NEGATIVE" if negative, and "NEUTRAL" if the score is zero.

4.Problem Bucketing/Classification:

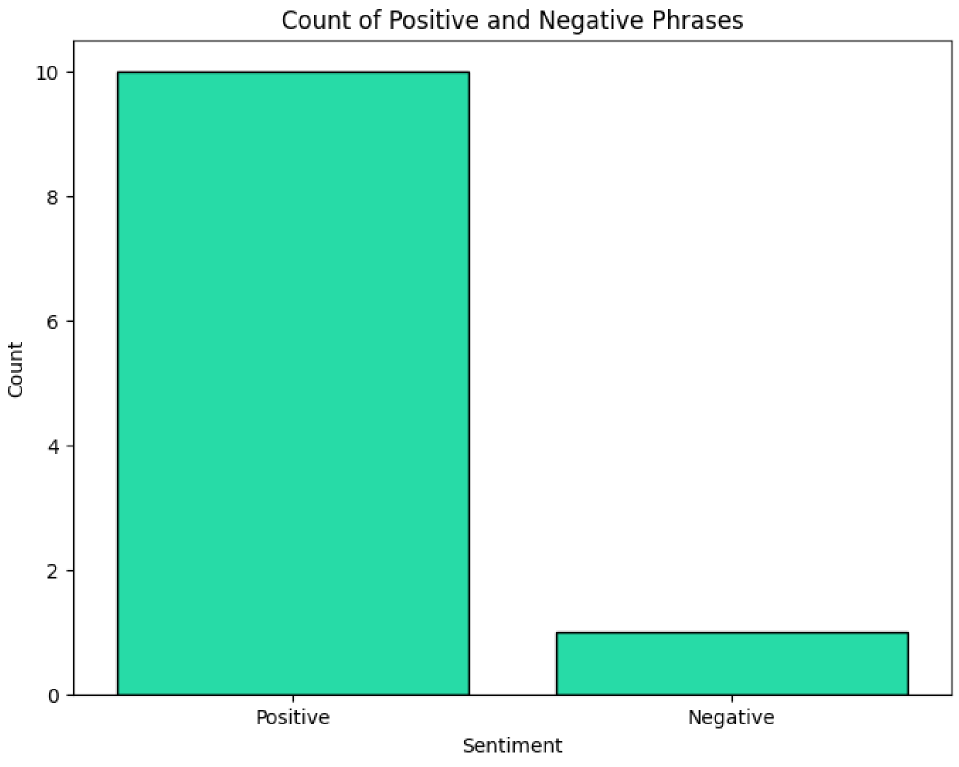
```
def analyze_problem_category(transcription):
#classify ing into different categories
if "cancellation" in transcription.lower() or "refund" in transcription.lower():
return "Cancellation/Refund"
elif "reschedule" in transcription.lower() or "flight delay" in transcription.lower():
return "Reschedule/Flight Delay"
elif "staff behavior" in transcription.lower():
return "Staff Behavior"
else:
return "Miscellaneous"
```

This function categorizes transcriptions based on their content into specific problem categories commonly encountered in airline customer service calls. It identifies keywords such as "cancellation," "refund," "reschedule," "flight delay," and "staff behavior" within the transcription to classify it accordingly. If none of these keywords are found, it categorizes the transcription as "Miscellaneous."

5.Visualizing the transcriptions:

```
def display_classification_results(categories, sentiments, positive_phrases_count,
negative_phrases_count):
    print("Problem Categories:")
    for category, count in categories.items():
        print(f"{category}:{count}")

    print("\nSentiment Analysis:")
    for sentiment, count in sentiments.items():
        print(f"{sentiment}:{count}")

    plt.figure(figsize=(8,6))
    sns.barplot(x=["Positive", "Negative"],
    y=[sentiments["POSITIVE"],sentiments["NEGATIVE"]])
    plt.title("Count of Positive and Negative Phrases")
    plt.xlabel("Sentiment")
    plt.ylabel("Count")
    plt.show()
```

This function displays the classification results of problem categories and sentiment analysis. It iterates through the provided dictionaries of categories and sentiments, printing their counts. Additionally, it visualizes the count of positive and negative sentiments using a bar plot.



6. CALCULATING KEY PERFORMANCE INDICATOR:

6.1 Average sentiment, Resolution time, Agent Performance:

```
def calculate_kpis(categories, sentiments, positive_phrases_count, negative_phrases_count,
total_transcriptions):
total_categories=sum(categories.values())
total_sentiments=sum(sentiments.values())
average_sentiment=(sentiments["POSITIVE"] - sentiments["NEGATIVE"]) /
total_sentiments if total_sentiments != 0 else 0
resolution_time=total_transcriptions / total_categories if total_categories != 0 else 0
agent_performance=sentiments["POSITIVE"] / total_sentiments if total_sentiments != 0
else 0
return average_sentiment, resolution_time, agent_performance
```

This function calculates various Key Performance Indicators (KPIs) for airline call center operations based on provided data. It computes the average sentiment, resolution time per issue, and agent performance metrics. The function returns these KPIs for further analysis and evaluation of call center performance.

6.2 Resolution rate:

```
def calculate_resolution_rate(conversation):
# Defining keywords for resolution
resolution=["resolved", "satisfied", "thank you", "problem solved"]
issues= any(indicator in conversation.lower() for indicator in resolution)
# Calculating resolution rate for positive and negative cases
resolution_rate = 1 if issues else 0
return resolution_rate
```

This function determines the resolution rate of a conversation by identifying specific keywords indicating resolution or satisfaction. It evaluates whether any of these keywords are present in the conversation and assigns a resolution rate accordingly. The function returns a binary indicator representing whether the conversation is resolved or not.

6.3 Transfer Rate:

```
def calculate_transfer_rate(conversation):
# Defining keywords for call transfer
transfer=["transferred", "transfer the call", "transferring you"]
no_transfers=0
segments=conversation.split(".")
# Check each segment for transfer keywords
for segment in segments:
if any(keyword in segment.lower() for keyword in transfer):
no_transfers+=1
total_calls=10
# Calculating the transfer rate
transfer_rate=no_transfers / total_calls
return transfer_rate
```

This function calculates the transfer rate of a conversation by identifying keywords related to call transfer. It splits the conversation into segments and counts the occurrences of transfer keywords. Then, it computes the transfer rate based on the number of transfers detected compared to a predefined total number of calls. The function returns the calculated transfer rate for further analysis.

6.4 Calculate ASA:

```
def calculate_asa(conversation):
    start_time="00:00:00"
    response_time="00:02:30"
    # Calculate Average Speed of Answer
    start_seconds=convert_to_seconds(start_time)
    response_seconds=convert_to_seconds(response_time)
    # Converting ASA from seconds to minutes
    asa_minutes=(response_seconds-start_seconds)/60
    return asa_minutes

#Converting timestamp string to seconds
def convert_to_seconds(timestamp):
    hours,minutes,second=map(int, timestamp.split(":"))
    total_seconds=hours*3600+minutes*60+second
    return total_seconds
```

This function calculates the Average Speed of Answer (ASA) for a conversation by determining the time elapsed between the start of the call and the first response. It converts the timestamps to seconds, computes the difference, and then converts it into minutes. The ASA value in minutes is returned for further analysis.

6.5 Error rate:

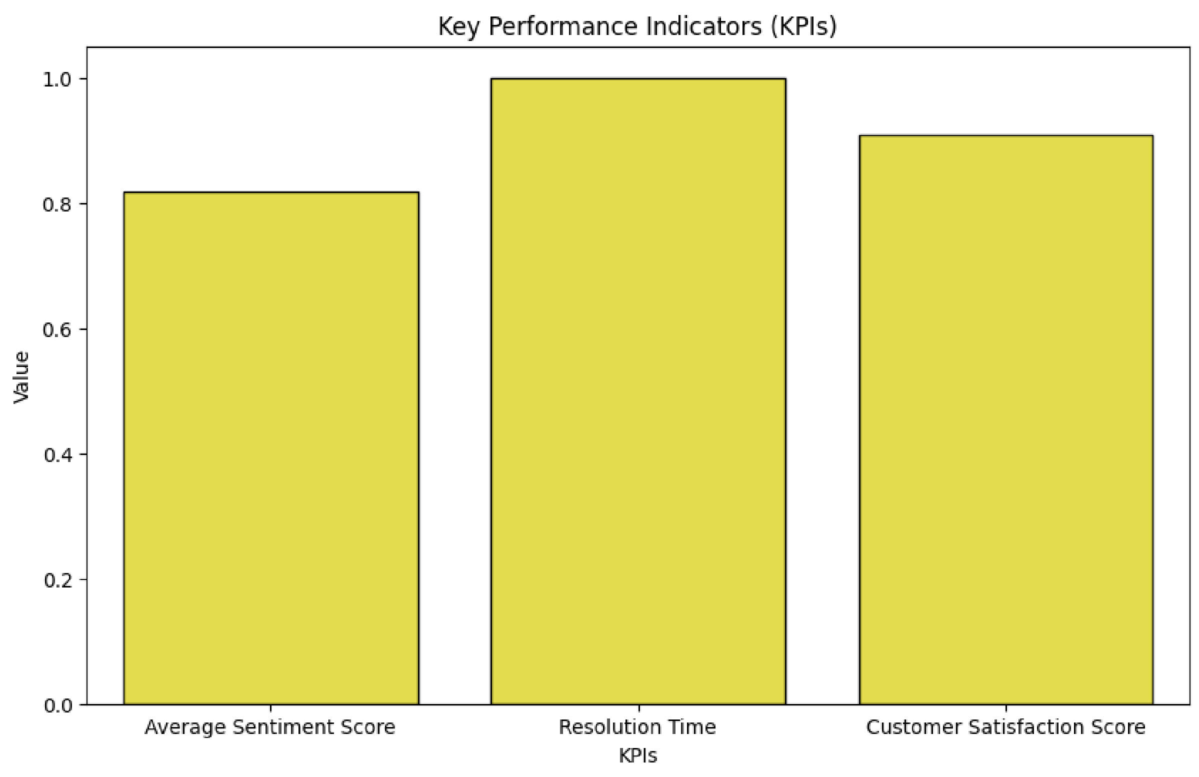
```
def calculate_error_rate(conversation):
# Defining keywords for errors
error_keywords = ["error", "issue", "problem", "mistake", "incorrect", "fault", "apology",
"sorry"]
num_errors = sum(conversation.lower().count(keyword) for keyword in error_keywords)
total_words = len(conversation.split())

# Calculating Error Rate
error_rate = (num_errors / total_words) * 100
return error_rate
```

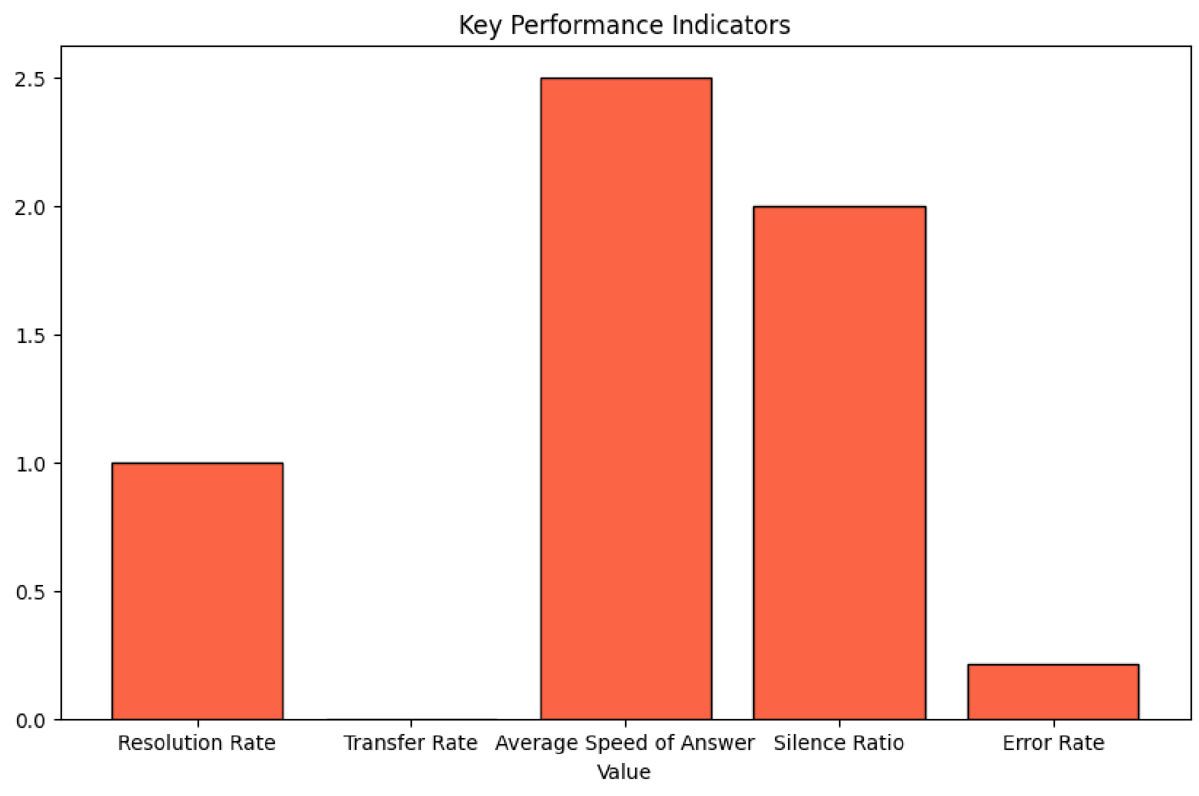
This function computes the error rate of a conversation by identifying specific keywords indicating errors or issues. It calculates the ratio of identified error keywords to the total number of words in the conversation, expressing the error rate as a percentage. The function returns the calculated error rate for further analysis.

6.6 Visualizing the KPI:

```
    def plot_kpis(average_sentiment, resolution_time, agent_performance):
  print("Average Sentiment Score:",average_sentiment)
  print("Resolution Time:",resolution_time)
  print("Agent Performance:",agent_performance)
  kpis = {"Average Sentiment Score": average_sentiment, "Resolution Time":
resolution_time, "Agent Performance": agent_performance}
  plt.figure(figsize=(10, 6))
  sns.barplot(x=list(kpis.keys()), y=list(kpis.values()))
  plt.title("Key Performance Indicators (KPIs)")
  plt.xlabel("KPIs")
  plt.ylabel("Value")
  plt.show()
```

This function displays the Key Performance Indicators (KPIs) - Average Sentiment Score, Resolution Time, and Agent Performance - along with their respective values. It generates a bar plot illustrating these KPIs for visual analysis and comparison. The function offers insights into call center performance based on the provided metrics.


Key Performance Indicators (KPIs)
Value
1.0
0.8
0.6
0.4
0.2
0.0
Average Sentiment Score
Resolution Time
Customer Satisfaction Score
KPIs


Key Performance Indicators
2.5
2.0
1.5
1.0
0.5
0.0
Resolution Rate
Transfer Rate
Average Speed of Answer
Silence Ratio
Error Rate
Value