PROJECT TITLE

MEETING SUMMARIZER

1. **Kiruthika. B - Register No: 2034025**
2. **Rithanya. R.B - Register No: 2034040**
3. **Sasruthisri. K - Register No: 2034042**

DEPARTMENT OF COMPUTING (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

COIMBATORE INSTITUTE OF TECHNOLOGY

(Autonomous Institution affiliated to Anna University)

COIMBATORE – 641014

Natural Language Processing Lab

MEETING SUMMARIZER

PROBLEM STATEMENT:

Efficient Meeting Summarization System: Develop a system capable of automatically summarizing audio recordings of meetings to extract key points and generate concise summaries. The system should employ speech recognition techniques to transcribe audio to text accurately, utilize natural language processing for text summarization and title generation, and integrate a conversational AI module to provide answers to queries related to the meeting content. The objective is to streamline the process of extracting actionable insights from meetings and enhance productivity.

DESCRIPTION OF DATASET:

- **Size**: The dataset comprises approximately 284MB of audio recordings.
- **Content**: The audio recordings capture various meetings or discussions relevant to the project's context. Each audio file likely corresponds to a single meeting session or a segment thereof, covering topics related to the project's domain.
- **Format**: The audio files are stored in a specific format, including MP3 or WAV. These formats are commonly used for digital audio storage and are compatible with a wide range of audio processing tools and libraries.
- **Metadata**: Each audio file may be accompanied by additional metadata, such as timestamps, speaker information, or any other contextual details. These metadata elements provide valuable context for understanding the content and context of each meeting recording.

DATA PREPROCESSING:

- **Conversion to WAV Format**: If the original audio files were in MP3 format, they might have been converted to the WAV format. This conversion ensures compatibility with various audio processing tools and libraries.
- **Normalization**: Audio normalization techniques may have been applied to standardize the audio levels across different recordings. Normalization helps mitigate variations in volume and ensures consistent audio quality.
- **Sampling Rate Adjustment**: The audio recordings might have been resampled to a specific sampling rate to ensure uniformity. Resampling is essential for compatibility with speech recognition and other audio processing algorithms.
- **Segmentation**: Lengthy audio recordings may have been segmented into smaller segments for easier processing. Segmenting the recordings facilitates tasks such as transcription and analysis of individual meeting segments.
- **Transcription**: Automatic speech recognition (ASR) systems may have been used to transcribe the audio recordings into text. These transcriptions serve as the basis for further analysis and processing in the project.
- **Text Cleaning**: The transcribed text data may have undergone cleaning and preprocessing steps to remove noise, punctuation, and other non-textual elements. Cleaning the text improves the accuracy of subsequent natural language processing tasks.

- **Tokenization**: The preprocessed text data may have been tokenized into individual words or tokens for analysis. Tokenization enables tasks such as text summarization, topic modeling, and sentiment analysis.

IDEOLOGY:

1. **Efficiency and Productivity Enhancement**: The project revolves around enhancing efficiency and productivity in organizational settings. The aim is to streamline the process of extracting actionable insights from meetings, saving time and resources for stakeholders.
2. **Access to Information**: Central to the project's ideology is the belief in democratizing access to meeting content. By converting audio recordings into summarized text and providing a chatbot interface for querying meeting content, individuals can quickly retrieve relevant information, regardless of their role or expertise.
3. **Facilitating Collaboration and Decision-making**: The project emphasizes the importance of facilitating collaboration and informed decision-making within organizations. By providing concise meeting summaries and answering questions related to meeting content, team members can stay informed and aligned, leading to more effective collaboration and decision-making processes.
4. **Continuous Improvement**: The project is grounded in the principle of continuous improvement. The goal is to develop scalable and adaptable meeting summarization solutions that evolve alongside organizational needs and technological advancements, fostering a culture of continuous learning and enhancement.

NOVELTY:

1. **Integration of Audio-to-Text Conversion and Summarization**: One key novelty lies in the seamless integration of audio-to-text conversion and summarization techniques. By automating the transcription process and generating concise summaries from meeting recordings, a comprehensive solution for extracting key insights from audio content is offered.
2. **Dynamic Title Generation**: The project introduces a novel approach to title generation for meeting summaries. By leveraging pre-trained models and advanced natural language processing techniques, informative and relevant titles that capture the essence of each meeting are generated, providing users with quick insights into the content.
3. **Interactive Chatbot Interface**: The integration of a chatbot module adds a novel interactive dimension. Users can engage with the system in natural language, querying meeting content and receiving instant responses. This real-time interaction enhances user experience and facilitates efficient information retrieval.
4. **Adaptability and Customization**: The project offers a novel degree of adaptability and customization, allowing users to tailor summarization parameters and preferences to their specific needs. This flexibility ensures that the summarization output aligns closely with the requirements and preferences of different users and organizational contexts.

MODEL BUILDING:

1. Data Preparation:

- Dataset Collection: Gather audio recordings of meetings along with their transcripts or ground truth summaries.
- Data Cleaning: Preprocess the audio files (e.g., normalization, noise reduction) and clean the text data (e.g., remove punctuation, handle special characters).
- Data Splitting: Divide the dataset into training, validation, and test sets to evaluate the model's performance.

2. Model Development:

- Feature Extraction: Extract relevant features from the audio data, such as spectrograms or MFCCs (Mel-Frequency Cepstral Coefficients), to represent the audio content.
- Text Processing: Preprocess the text data by tokenizing, padding sequences, and encoding the text for input into the model.
- Model Architecture: Design the architecture of the summarization model. This could involve using recurrent neural networks (RNNs), convolutional neural networks (CNNs), transformers, or a combination of these architectures.
- Training: Train the model on the training data using appropriate loss functions and optimization algorithms. Monitor the model's performance on the validation set to prevent overfitting.
- Hyperparameter Tuning: Fine-tune the model hyperparameters to optimize performance.

3. Evaluation:

- Metric Selection: Choose appropriate evaluation metrics for assessing the model's performance, such as ROUGE scores (Recall-Oriented Understudy for Gisting Evaluation) for text summarization tasks.
- Evaluation on Test Set: Evaluate the trained model on the test set to measure its summarization quality and generalization ability.
- Qualitative Assessment: Conduct qualitative analysis by inspecting example summaries generated by the model to identify strengths and weaknesses.

4. Deployment:

- Model Serialization: Serialize the trained model and save its weights and architecture for deployment.
- API Development (Optional): Develop an API for serving the model predictions, allowing users to interact with the summarization system programmatically.
- User Interface Development: Build a user-friendly interface for users to upload audio recordings or input text and receive summarized outputs.
- Scalability and Performance Optimization: Ensure the deployed system can handle multiple requests efficiently and optimize its performance for real-time or batch processing scenarios.

5. Continuous Improvement:

- Feedback Loop: Collect feedback from users and stakeholders to identify areas for improvement in the summarization system.
- Model Iteration: Iteratively refine the model based on feedback, incorporate new data, and update the system to enhance its performance and usability.

MODULES:

1. Audio-to-Text Conversion Module:

The Audio-to-Text Conversion module is responsible for converting audio recordings of meetings into text transcripts. This process involves utilizing automatic speech recognition (ASR) techniques to transcribe spoken words into written text, making the meeting content accessible for further analysis and summarization.

Code:

```
#@title Directory containing the .wav files after getting total data
folder_path                                                          =
"/content/drive/MyDrive/kitti/ourproject/dataset/audio_data/wavfiles"

# Define schema for DataFrame
schema = StructType([
    StructField("audio_content", ArrayType(FloatType())),
    StructField("sampling_rate", LongType())
])

# List to store data from all files
data_list = []

# Iterate over files in the folder
for file_name in os.listdir(folder_path):
    if file_name.endswith(".wav"):
        # Construct the full file path
        file_path = os.path.join(folder_path, file_name)
```

```
        # Load the audio file and convert to floats
        data, sampling_rate = librosa.load(file_path, sr=16000)
        data = [float(x) for x in data]

        # Append data to list
        data_list.append((data, sampling_rate))

# Create a DataFrame from the list
df = pd.DataFrame(data_list, columns=["audio_content",
"sampling_rate"])
spark_df = spark.createDataFrame(df, schema)

# Show the DataFrame schema
spark_df.printSchema()

# Show the first few rows of the DataFrame
spark_df.show()
audio_assembler = AudioAssembler() \
    .setInputCol("audio_content") \
    .setOutputCol("audio_assembler")

speech_to_text = WhisperForCTC \
    .pretrained()\
    .setInputCols("audio_assembler") \
    .setOutputCol("text")

pipeline = Pipeline(stages=[
  audio_assembler,
  speech_to_text,
])

pipelineDF = pipeline.fit(spark_df).transform(spark_df)
text_df = pipelineDF.select("text.result", "text.metadata")
text_df.show(truncate=False)
```

OUTPUT

```
+--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
---------------------------------------+------------------------------
--+
|result
|metadata                             |
+--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
--------------------------------------------------------------------------
---------------------------------------+------------------------------
--+
|[ Hello, this is to test my project for NLP on meeting some a riser. I
hope this works well and I am just talking a lot to bring in more words
so I get to see the output more better. Thank you.]
|[{length -> 309248, audio -> 0}] |
|[ In this project, our goal is to use NLP to create a smart chatbot
that can answer questions and the chatbot will use advanced NLP methods
to understand what users ask find the right information and give
accurate answers quickly.]
|[{length -> 290134, audio -> 0}] |
|[ What is text summarization? Different from other natural language
process models. Text summarization requires fully understanding, attech
semantic information which is consistent with the key points of human
summarization and aims to compress documents into shorter text that
summarizes the essential information of the source text.]
|[{length -> 379806, audio -> 0}] |
|[ Abstractive Tech Summerization Models based on semantic meaning
often utilized advanced NLP techniques and deep learning architectures.
One popular architecture for the stats is the transformer model,
particularly variance like bird and GPT. Below is an example called
using hugging face transformers, library to perform abstractive tech
summarization with bar model which specifically designed for]
|[{length -> 536926, audio -> 0}] |
|[ I am going through some things with my feelings and myself. My Bali
is deep and I do not think but think about how I am good and how I
should be here. I have tried and never attempted for committing any pad
things, bad decisions and I wanted to fix all my issues by myself but I
never get it around of it.]
|[{length -> 414046, audio -> 0}] |
|[ Abstractive summarization involves generating a summary that
contains a new faces, sentences or word not present in the source
document. It aims to produce a conscious representation of original
text in the mode human like manor. Unlike the extractive method, the
abstractive summarization involves understanding the meaning of text
```

and paraphrasing it creates a summary. Abstractive method often uses a natural language processing, techniques such as machine, machine translation, text, generation and cinematic analysis to further]|[{length -> 522206, audio -> 0}] |
[How about dealing with abstractive methods? So, the abstractive text demonstration methods comes with example like sequence sequence models or transformer based models. These models often based on neural networks, they take the input text and generated summary in a sequence sequence manner, they have an encoder and decoder architecture where the encoder process the input text and decoder generates the summary.]	[{length -> 389726, audio -> 0}]
[What is transformer based models? These architectures like transformer architectures such as bidirectional encoder representation from transformer's birth or generative pre-trained models, how shown remarkable performance and abstract or summarization paths. These models can generate coherent and contextually relevant summaries by leveraging large pre-trained language models.]	[{length -> 395806, audio -> 0}]
[Roces are flowers that grow on bushes with short thrones or prickles, let's protect the plant. They have glossy green leaves with two-third edges and coming mini colors in sizes. Roces can be shops, climbers or mini-chapart plants. They are native to China but are non-grown all over the world. Roces are in season from its spring to autumn and the plant goes dormant in the winter. During summer water should be given inside the rose plant every day.]	[{length -> 506846, audio -> 0}]
[What is this?]	[{length -> 144606, audio -> 0}]
[Here I am going to give a comparison between the approach using spacing and the approach using transformers. First we will see a space approach. A space approach is relatively lightweight which is comparing to transformers and it is optimized for speed and it is more as a generally faster to process the text comparing to transformers.]	[{length -> 1131520, audio -> 0}]
[Then coming to transformers approach, there are some pros in trans bombers particularly models like bird and bird or the state of the art architectures that have demonstrated superior performance on the wide range of NLP tasks including question answering. They can capture complex linguistic patterns and the context, dependency more effectively than traditional models like species. Then pre-trained models]	[{length -> 1754454, audio -> 0}]
[In summary, the Spacey approach offers a lightweight and first solution for NLP task making it suitable for real-time application and it provides linguistic features like part of speech tagging that is POS tagging and named entity regulation out of the box. However, it context understanding may be limited due to reliance on individual's illnesses and it employs role-based mother which can struggle]	[{length -> 1269760, audio -> 0}]
+--

--+-------------------------------
--+

EVALUATION:

```
from nltk.translate.bleu_score import sentence_bleu
from nltk.translate.bleu_score import SmoothingFunction
import jiwer

# Read the generated text and ground truth text
with open("final_All_data.txt", "r") as f:
    generated_text = f.read().strip()

with open("ground_truth.txt", "r") as f:
    ground_truth_text = f.read().strip()

# BLEU score
def compute_bleu_score(candidate, reference, n=1):
    # Tokenize the text
    candidate_tokens = candidate.split()
    reference_tokens = reference.split()

    # Compute BLEU score
    smoothing_function = SmoothingFunction().method4
     bleu_score = sentence_bleu([reference_tokens], candidate_tokens,
smoothing_function=smoothing_function, weights=(1/n,)*(n))
    return bleu_score

bleu_1_score = compute_bleu_score(generated_text, ground_truth_text,
n=1)
bleu_2_score = compute_bleu_score(generated_text, ground_truth_text,
n=2)
bleu_3_score = compute_bleu_score(generated_text, ground_truth_text,
n=3)

print("BLEU-1 Score:", bleu_1_score)
print("BLEU-2 Score:", bleu_2_score)
print("BLEU-3 Score:", bleu_3_score)

bleu_score = compute_bleu_score(generated_text, ground_truth_text)
print("BLEU Score:", bleu_score)

# Word Error Rate (WER)
def compute_wer(candidate, reference):
    # Compute WER
    wer = jiwer.wer(reference, candidate)
    return wer

wer = compute_wer(generated_text, ground_truth_text)
print("Word Error Rate (WER):", wer)
```

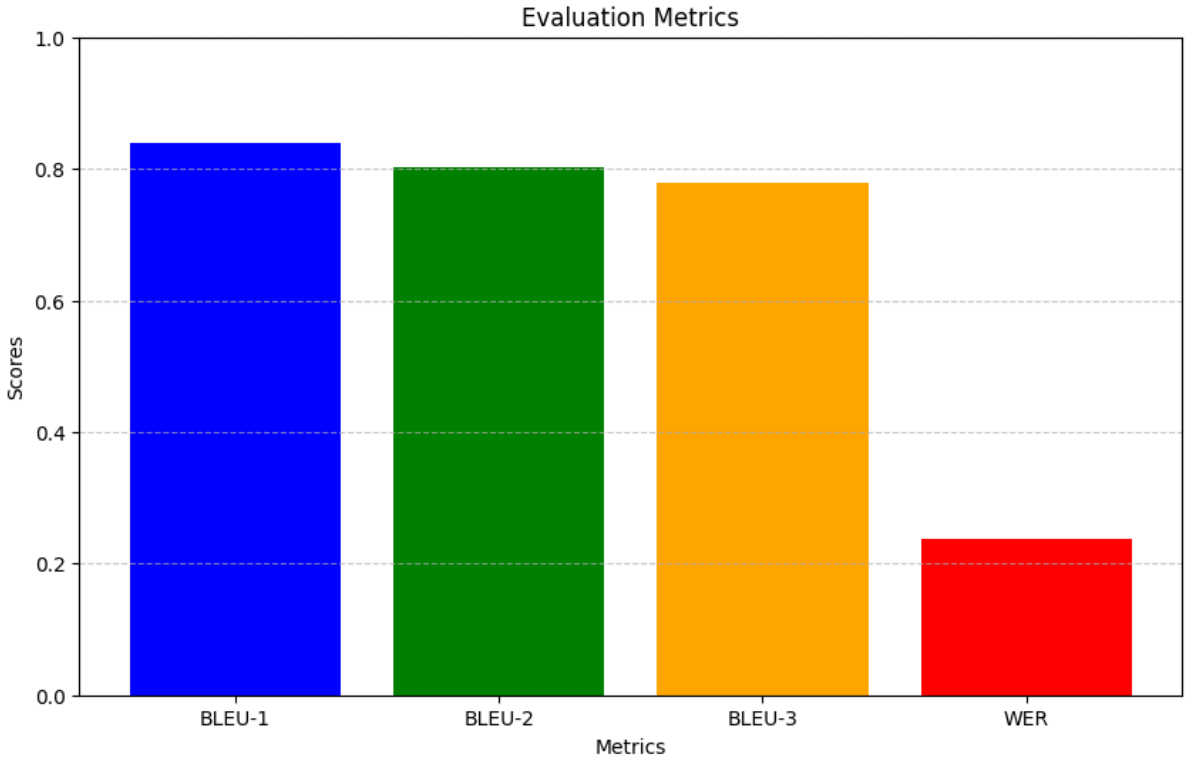
```
import matplotlib.pyplot as plt

# Data
metrics = ['BLEU-1', 'BLEU-2', 'BLEU-3', 'WER']
scores = [bleu_1_score, bleu_2_score, bleu_3_score, wer]

# Plotting
plt.figure(figsize=(10, 6))
plt.bar(metrics, scores, color=['blue', 'green', 'orange', 'red'])
plt.xlabel('Metrics')
plt.ylabel('Scores')
plt.title('Evaluation Metrics')
plt.ylim(0, 1)  # Set y-axis limit to range from 0 to 1 for BLEU scores
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

OUTPUT

BLEU-1 Score: 0.839943342776204
BLEU-2 Score: 0.8028401256125111
BLEU-3 Score: 0.7788342843910555
BLEU Score: 0.839943342776204
Word Error Rate (WER): 0.23659305993690852



2. Text Summarization Module

The Text Summarization module generates a concise summary of the meeting transcript. This summary captures the essential points discussed during the meeting, enabling stakeholders to quickly grasp the key insights without the need to review the entire transcript.

CODE:

```
import spacy
from spacy.lang.en.stop_words import STOP_WORDS
from heapq import nlargest

# Load the English language model
nlp = spacy.load("en_core_web_sm")

# Function to generate summary
def generate_summary(text, num_sentences=3):
    # Parse the input text using spaCy
    doc = nlp(text)

    # Calculate word frequency
    word_frequencies = {}
    for word in doc:
        if word.text.lower() not in STOP_WORDS:
            if word.text.lower() not in word_frequencies.keys():
                word_frequencies[word.text.lower()] = 1
            else:
                word_frequencies[word.text.lower()] += 1

    # Normalize word frequencies
    max_frequency = max(word_frequencies.values())
    for word in word_frequencies.keys():
        word_frequencies[word] = word_frequencies[word] / max_frequency

    # Calculate sentence scores based on word frequencies
    sentence_scores = {}
    for sent in doc.sents:
        for word in sent:
            if word.text.lower() in word_frequencies.keys():
                if sent not in sentence_scores.keys():
                                                sentence_scores[sent]    =
word_frequencies[word.text.lower()]
                else:
                                               sentence_scores[sent]    +=
word_frequencies[word.text.lower()]
```

```
    # Get top N sentences based on scores
        summary_sentences = nlargest(num_sentences, sentence_scores,
key=sentence_scores.get)
    summary = ' '.join([sent.text for sent in summary_sentences])
    return summary

# Read the input file
with open("final_All_data.txt", "r") as file:
    input_text = file.read()

# Generate summary
summary = generate_summary(input_text)

# Format the summary as a single text paragraph
formatted_summary = summary.replace('\n', ' ')

print("Summary:")
print(formatted_summary)
```

OUTPUT:

```
Summary:
These models often based on neural networks, they take the input text
and generated summary in a sequence sequence manner, they have an
encoder and decoder architecture where the encoder process the input
text and decoder generates the summary.']  ] [' In this project, our
goal is to use NLP to create a smart chatbot that can answer questions
and the chatbot will use advanced NLP methods to understand what users
ask find the right information and give accurate answers quickly.']
Then pre-trained models'] [" In summary, the Spacey approach offers a
lightweight and first solution for NLP task making it suitable for
real-time application and it provides linguistic features like part of
speech tagging that is POS tagging and named entity regulation out of
the box.
```

3. Title Generation Module

The Title Generation module is responsible for generating a relevant title for the meeting summary. This title succinctly represents the main focus or outcome of the meeting, providing a clear and informative heading for the summarized content.

CODE:

```
import torch
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

# Load pre-trained tokenizer and model
tokenizer = AutoTokenizer.from_pretrained("czearing/article-title-generator")
model = AutoModelForSeq2SeqLM.from_pretrained("czearing/article-title-generator")

# Define a function to generate title from input text
def generate_title_from_file(file_path):
    # Read text from file
    with open(file_path, 'r') as file:
        input_text = file.read()

    # Tokenize input text
    inputs = tokenizer(input_text, return_tensors="pt", max_length=512, truncation=True)

    # Generate title from input text
    with torch.no_grad():
        output = model.generate(**inputs, max_length=50, num_beams=5, early_stopping=True)

    # Decode generated title
    generated_title = tokenizer.decode(output[0], skip_special_tokens=True)

    return generated_title


file_path = "/content/final_All_data.txt"
generated_title = generate_title_from_file(file_path)
print("Generated Title:", generated_title)
```

OUTPUT:

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100% 2.35k/2.35k [00:00<00:00, 42.9kB/s]
tokenizer.json: 100% 2.42M/2.42M [00:00<00:00, 11.7MB/s]
special_tokens_map.json: 100% 2.20k/2.20k [00:00<00:00, 47.7kB/s]
config.json: 100% 1.48k/1.48k [00:00<00:00, 102kB/s]
pytorch_model.bin: 100% 892M/892M [00:12<00:00, 55.8MB/s]
Generated Title: Abstractive Tech Summarization
```

4. Chatbot Module

The Chatbot Module allows users to interact with the system by asking questions related to the meeting content. It leverages natural language processing techniques to understand user queries and provide relevant answers based on the meeting transcript or summary.

CODE:

```
from transformers import pipeline

# Load the question answering pipeline
qa_pipeline = pipeline("question-answering")

# Function to handle user queries
def chatbot_response(user_input, text):
    if "summary" in user_input.lower():
        # Summarize the text file
        summary = generate_summary(text)
        return f"Here is the summary: {summary}"
    else:
        # Use question-answering model to answer user's question
        answer = qa_pipeline({
            "question": user_input,
```

```
            "context": text  # Provide the meeting content as context
        })
        return answer['answer']

# Read meeting content
file_path = "final_All_data.txt"
meeting_content = read_meeting_content(file_path)

# Start the conversation loop
print("Welcome to the Meeting Chatbot!")
print("You can ask questions about the meeting content. Type 'exit' to
end the conversation.")

while True:
    user_input = input("You: ")
    if user_input.lower() == "exit":
        print("Goodbye!")
        break

    response = chatbot_response(user_input, meeting_content)
    print("Chatbot:", response)
```

OUTPUT:

```
No model was supplied, defaulted to distilbert/distilbert-base-cased-distilled-squad and revision 626af31 (https://huggingface.co/distilbert/distilbert-base-cased-distilled-squad).
Using a pipeline without specifying a model name and revision in production is not recommended.
Welcome to the Meeting Chatbot!
You can ask questions about the meeting content. Type 'exit' to end the conversation.
You: hi
Chatbot: generative pre-trained models
You: summary
Chatbot: Here is the summary: These models often based on neural networks, they take the input text and generated summary in a sequence sequence manner, they have an encoder and de
 ]
[' In this project, our goal is to use NLP to create a smart chatbot that can answer questions and the chatbot will use advanced NLP methods to understand what users ask find the r
 Then pre-trained models']
[" In summary, the Spacey approach offers a lightweight and first solution for NLP task making it suitable for real-time application and it provides linguistic features like part o
You: spacey approach
Chatbot: lightweight
You: chatbot
Chatbot: POS tagging
You: our goal
Chatbot: use NLP to create a smart chatbot
You: exit
Goodbye!
```

INFERENCE:

1. **End-to-End Solution:** The system provides an end-to-end solution for meeting summarization, starting from audio-to-text conversion to generating summaries, titles, and responding to user queries.
2. **Efficiency and Accessibility:** By automating the transcription process and summarizing meeting content, the system enhances efficiency and accessibility, allowing stakeholders to quickly extract key insights without manually reviewing lengthy transcripts.
3. **User-Friendly Interface:** The integration of a chatbot and user interface facilitates seamless interaction with the system, enabling users to upload audio files, view summaries, titles, and obtain information through natural language queries.
4. **Robustness and Scalability:** The system's robustness is demonstrated through its ability to handle various audio formats, transcribe speech accurately, and generate coherent summaries. Additionally, its modular design enables scalability and flexibility for future enhancements and integrations.
5. **Potential Impact:** The meeting summarization system has the potential to significantly impact organizational decision-making, collaboration, and productivity by providing timely and actionable insights from meeting discussions.

ADVANTAGES:

1. **Time-Saving:** Enables stakeholders to quickly grasp key insights from meetings without the need to review lengthy transcripts, saving time and increasing productivity.
2. **Enhanced Accessibility:** Democratizes access to meeting content by converting audio recordings into text and providing summarized insights, making information more accessible to a wider audience.
3. **Improved Decision-making:** Facilitates informed decision-making by distilling complex meeting discussions into concise summaries, enabling stakeholders to focus on actionable insights.
4. **Efficient Collaboration:** Promotes efficient collaboration among team members by providing a shared understanding of meeting outcomes and action items, fostering alignment and coordination.
5. **Scalability:** The modular design allows for scalability and flexibility, enabling the system to adapt to varying organizational needs and accommodate future enhancements.

USE CASES

1. Corporate Meetings: Provides executives, managers, and team members with summarized insights from board meetings, project updates, and strategy sessions, facilitating strategic decision-making.
2. Academic Settings: Assists students, researchers, and educators in summarizing lectures, seminars, and academic discussions, helping them review key concepts and findings efficiently.

3. Legal Proceedings: Aids legal professionals in summarizing depositions, hearings, and courtroom proceedings, enabling them to extract pertinent information for case preparation and analysis.
4. Medical Consultations: Supports healthcare professionals in summarizing patient consultations, medical conferences, and research discussions, facilitating knowledge sharing and collaboration within the medical community.
5. Media Monitoring: Assists media professionals in summarizing interviews, press conferences, and panel discussions, enabling them to extract newsworthy information and trends for reporting and analysis.

CONCLUSION:

The meeting summarization system offers a powerful solution for extracting actionable insights from audio recordings of meetings, enhancing decision-making, collaboration, and productivity across various domains. By leveraging natural language processing and audio analysis techniques, the system streamlines the process of summarizing meeting content, making it more accessible and actionable for stakeholders. With its user-friendly interface, scalability, and potential impact on organizational efficiency, the meeting summarization system represents a valuable tool for extracting knowledge and fostering informed decision-making in diverse settings. As organizations continue to embrace digital transformation and seek innovative solutions for managing information overload, the meeting summarization system stands out as a key enabler of efficiency, collaboration, and strategic alignment.