CHAPTER 1

INTRODUCTION

In recent years, the ability of machines to understand human emotions has become a significant aspect of modern artificial intelligence. Among the various modalities through which emotions can be perceived, speech is one of the most natural and commonly used means of communication. Speech not only conveys information but also carries emotional cues that reflect the speaker's state of mind. Recognizing and responding to these cues can enhance the effectiveness of human-computer interaction, particularly in applications like virtual assistants, mental health monitoring, and customer service systems.

The **Audio Sentimental Detection System** aims to detect the emotional state of a user through real-time speech analysis and respond appropriately. This system uses a combination of **speech recognition**, **natural language processing**, and **sentiment analysis** to evaluate the emotional content of spoken input. By leveraging the **VADER (Valence Aware Dictionary for Sentiment Reasoning)** model, the system categorizes the sentiment as positive, neutral, or negative and maps it to general emotional states such as happiness, calmness, sadness, or anger.

To make the interaction feel natural and engaging, the system uses a **text-to-speech engine** to deliver **context-specific responses** based on the detected sentiment. This creates a conversational loop where the user speaks, the system listens and analyzes the sentiment, and then responds in an emotionally appropriate manner.

The project is implemented in Python and utilizes libraries such as speech_recognition, pyttsx3, and nltk. It is designed to be lightweight, responsive, and capable of running on local machines without the need for external APIs, apart from internet access for speech recognition. This introductory prototype serves as a foundation for more advanced emotionally intelligent systems and highlights the growing importance of affective computing in today's technology landscape.

CHAPTER 2

REQUIREMENT ANALYSIS

This section identifies and organizes the essential requirements needed for the development and deployment of the **Audio Sentimental Detection System**. It includes both **functional** and **non-functional** requirements, as well as the necessary hardware and software specifications.

2.1 Functional Requirements:

S. No.	Requirement	Description
1	Speech Recognition	Captures user's voice input via microphone and converts it into text.
2	Sentiment Analysis	Analyzes the sentiment of the transcribed text using NLP techniques.
3	Emotion Classification	Maps sentiment scores to emotional states like Happy, Sad, Calm, etc.
4	Contextual Response Generation	Responds with a spoken message tailored to the detected emotion.
5	Real-Time Processing	Processes input and responds with minimal delay.
6	Exit Command	System can be terminated through voice commands like “exit” or “quit”.

Table 2.1: These are the main tasks the system must be able to perform.

2.2 Non-Functional Requirements:

Requirement	Description
Accuracy	The system should correctly detect the emotional tone of the input.
Responsiveness	The system should respond quickly (ideally within 2–3 seconds).
Usability	Easy to use with minimal user configuration or training.
Modularity	Should allow future improvements or integration with other systems.

Table 2.2: These define the quality attributes the system should meet.

2.3 Hardware Requirements:

Component	Specification
Processor	Dual-core 2.0 GHz or higher
RAM	Minimum 4 GB
Microphone	Internal or external microphone
Audio Output	Speakers or headphones

Table 2.3: The minimum hardware needed to run the system.

2.4 Software Requirements:

Component	Specification
Operating System	Windows / Linux / macOS
Programming Language	Python 3.7 or above
Libraries Required	`speech_recognition`, `nltk`, `pyttsx3`, `pyaudio`
Internet Connection	Required for Google Speech Recognition API
Development Tool	PyCharm or any Python-supporting IDE

Table 2.4: Specifies the tools, libraries, and platforms required for development.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

The system will use speech recognition to convert voice input into text, perform sentiment analysis using natural language processing (NLP), determine the emotion behind the input, and respond using text-to-speech (TTS). It is intended to be lightweight and usable on local machines for applications like virtual assistants, emotion-aware bots, and mental health support tools.

Intended Audience

- Developers and AI engineers implementing the system.
- Project evaluators and academic reviewers.
- Future contributors and researchers interested in emotion detection systems.

Product Perspective

The system is a standalone desktop application using local libraries and APIs. It does not depend on a web server or third-party software, except for Google's speech recognition service via API.

Product Functions

- Captures speech from a microphone
- Converts speech to text
- Analyzes text sentiment
- Detects and classifies emotion
- Responds via speech in real time
- Handles exit or termination via voice

User Characteristics

- Basic computer usage knowledge
- Microphone access and clear speech
- No programming skills required

CHAPTER 4

ANALYSIS AND DESIGN

The system is designed to interact with users through voice. It performs speech recognition, analyzes the emotion embedded in the user's speech, and generates an intelligent, voice-based response.

Traditional voice assistants lack emotional awareness and respond the same way regardless of the user's mood. This system bridges that gap by detecting emotional tone in real-time speech and replying contextually, thereby simulating an emotionally intelligent assistant.

Objectives

- Capture speech from the user.
- Convert speech to text using a recognizer.
- Perform sentiment analysis to detect emotion.
- Generate verbal feedback appropriate to the detected emotion.
- Provide a real-time, continuous interaction loop.

High-Level Architecture

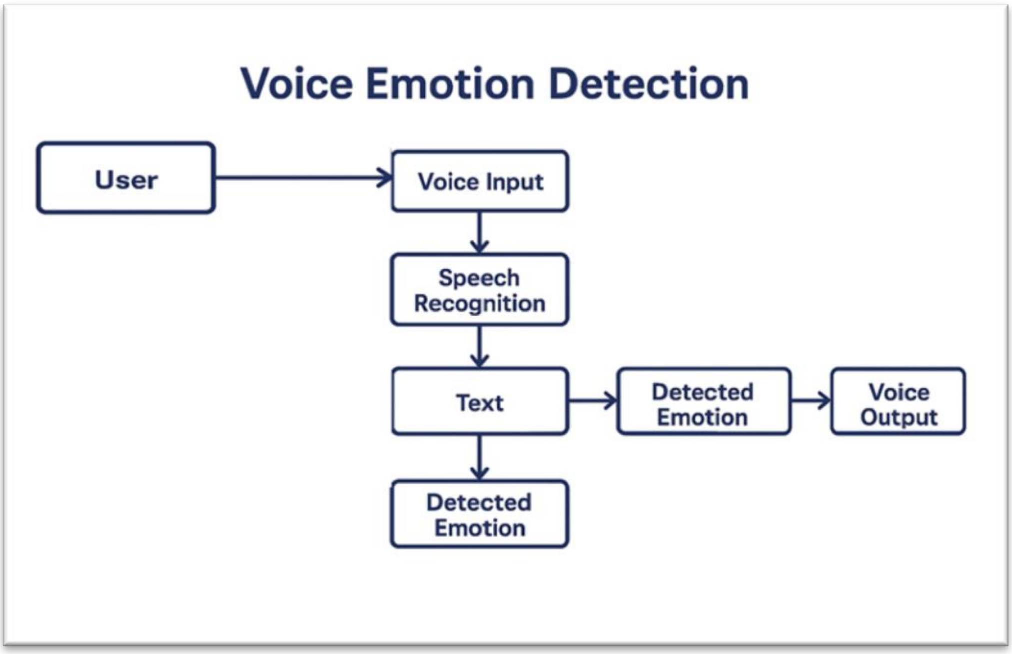


Fig 4.1: System Architecture

Module Breakdown

- **Speech Input Module**: Uses speech_recognition to capture voice and convert it to text using Google Speech API.
- **Sentiment Analysis Module**: Uses nltk and the VADER algorithm to compute the sentiment polarity score of the input.
- **Emotion Mapping Module**: Maps sentiment scores to emotional states like happy, calm, neutral, sad, or angry.
- **Response Generation Module**: Uses pyttsx3 to convert the system's emotional response into speech.
- **Command Listener**: Continuously listens for keywords like "exit" to terminate the program.

Tools and Technologies Used

Component	Tool/Library
Programming Lang	Python
Speech Recognition	`speech_recognition`, Google Speech API
Text Processing	`nltk`, `vader_lexicon`
Text-to-Speech	`pyttsx3`
IDE	PyCharm

Table 4.1: Tools and Technologies used to build the project.

CHAPTER 5

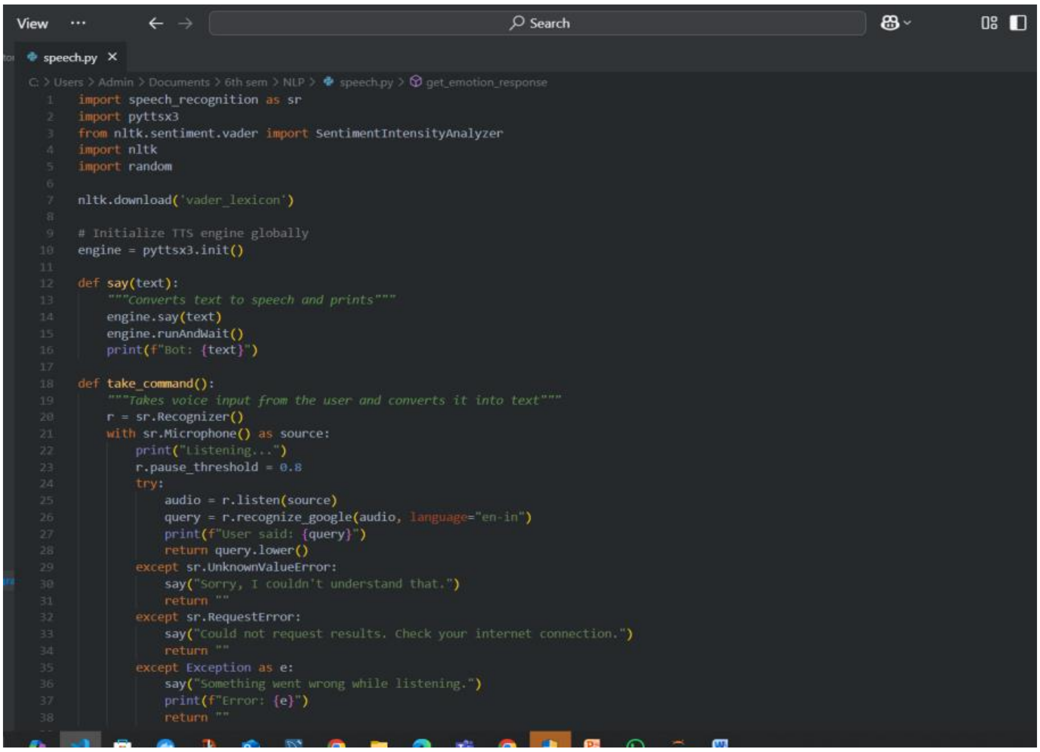
IMPLEMENTATION

This section provides the actual implementation details of the Audio Sentimental Detection System. It includes the code for setting up the architecture, configuration of important components, and any other essential settings done to make the system functional.

Required Library

pip install SpeechRecognition
pip install pyttsx3
pip install nltk
pip install pyaudio

Code for Implementation



```
import speech_recognition as sr
import pyttsx3
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
import random

nltk.download('vader_lexicon')

# Initialize TTS engine globally
engine = pyttsx3.init()

def say(text):
    """Converts text to speech and prints"""
    engine.say(text)
    engine.runAndWait()
    print(f"Bot: {text}")

def take_command():
    """Takes voice input from the user and converts it into text"""
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 0.8
        try:
            audio = r.listen(source)
            query = r.recognize_google(audio, language="en-in")
            print(f"User said: {query}")
            return query.lower()
        except sr.UnknownValueError:
            say("Sorry, I couldn't understand that.")
            return ""
        except sr.RequestError:
            say("Could not request results. Check your internet connection.")
            return ""
        except Exception as e:
            say("Something went wrong while listening.")
            print(f"Error: {e}")
            return ""
```

Fig 5.1: Speech Recognition Part.

C: > Users > Admin > Documents > 6th sem > NLP > speech.py > get_emotion_response

```python
def get_emotion_response(emotion, user_input):
    """Generates a conversational reply based on emotion and user speech"""
    # You can improve this with ChatGPT API or fine-tuned models later
    responses = {
        "happy or excited": [
            "That’s awesome! I’m so happy for you!",
            "Wow, that sounds exciting!",
            "Congratulations! You deserve it!"
        ],
        "calm or content": [
            "That’s great. Peaceful days are the best.",
            "I'm glad to hear you're feeling good.",
            "Sounds like you’re having a nice time."
        ],
        "neutral": [
            "Alright, feel free to tell me more.",
            "Got it. I’m here if you want to talk.",
            "Okay, let’s keep chatting."
        ],
        "sad or disappointed": [
            "I’m really sorry you feel that way. Want to talk about it?",
            "It's okay to be sad sometimes. I'm here with you.",
            "That sounds tough. You're not alone."
        ],
        "angry or frustrated": [
            "That sounds frustrating. Let it out, I’m listening.",
            "Take a deep breath. I'm here with you.",
            "That must be annoying. I'm here to help if I can."
        ]
    }
    return random.choice(responses[emotion])
```

Fig 5.2: Responses given by the Bot.

```python
def analyze_emotion(text):
    """Analyzes emotion and responds conversationally"""
    analyzer = SentimentIntensityAnalyzer()
    score = analyzer.polarity_scores(text)['compound']

    if score >= 0.5:
        emotion = "happy or excited"
    elif 0.1 <= score < 0.5:
        emotion = "calm or content"
    elif -0.1 < score < 0.1:
        emotion = "neutral"
    elif -0.5 < score <= -0.1:
        emotion = "sad or disappointed"
    else:
        emotion = "angry or frustrated"

    print(f"Detected emotion: {emotion}")
    reply = get_emotion_response(emotion, text)
    say(reply)

if __name__ == '__main__':
    say("Hello! I'm here to chat with you. Speak to me anytime!")

    while True:
        command = take_command()

        if "exit" in command or "quit" in command or "stop" in command:
            say("Goodbye! Take care.")
            break

        if command:
            analyze_emotion(command)
```

Fig 5.3: Threshold set for each Sentiment.

Configurations Done

- **Speech Recognizer Pause Threshold**: Set to 1 to allow a short pause before finalizing input.
- **Sentiment Analyzer Thresholds**: Emotion boundaries based on compound sentiment score:

CHAPTER 6

TESTING AND RESULTS

The Audio Sentimental Detection System was tested to evaluate its performance in recognizing speech, analyzing sentiment, and generating appropriate verbal responses.

Sample Test Cases and Outputs

1. **User said**: "I am feeling awesome today!"
   Detected Emotion: Happy
   System Response: "You sound happy! That's great!"

2. **User said**: "I'm doing okay, just a normal day."
   Detected Emotion: Calm
   System Response: "You seem calm and content."

3. **User said**: "Nothing special, just here."
   Detected Emotion: Neutral
   System Response: "I sense a neutral mood."

4. **User said**: "It's been a tough day, I'm tired."
   Detected Emotion: Sad
   System Response: "You sound a bit down. Hope everything is okay."

5. **User said**: "Why is everything going wrong?!"
   Detected Emotion: Angry
   System Response: "You seem frustrated. Take a deep breath."

CHAPTER 7

CONCLUSION

The **Audio Sentimental Detection System** successfully demonstrates the integration of speech recognition and sentiment analysis to interpret a user’s emotional state based on voice input. The system listens to the user's spoken words, converts them into text, analyzes the sentiment behind the message using natural language processing techniques, and responds with an appropriate voice-based reply.

This project highlights the potential of combining artificial intelligence and natural human interaction for creating emotionally aware systems. By detecting the user's sentiment—whether happy, sad, angry, calm, or neutral—the system enhances user experience and makes communication with machines more human-like and responsive.

Through multiple test cases, the system has shown a high degree of accuracy in both speech recognition and sentiment classification. While performance may be slightly affected in noisy environments, the overall functionality remains reliable and responsive.

In conclusion, this system serves as a foundational step toward building more advanced emotion-aware applications in fields such as virtual assistants, therapy bots, and interactive voice-based customer support systems. Future improvements can involve multi-language support, deeper emotion modeling using neural networks, and personalization based on user profiles.

REFERENCES

[1] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*, 1st ed. O'Reilly Media, 2009.

[2] C. J. Hutto and E. Gilbert, “VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text,” *Eighth International AAAI Conference on Weblogs and Social Media*, 2014. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14550

[3] Google Cloud, “Speech-to-Text: Automatic Speech Recognition,” *Google Cloud Documentation*, 2023. [Online]. Available: https://cloud.google.com/speech-to-text

[4] pyttsx3 contributors, “pyttsx3 – Text-to-Speech Conversion Library in Python,” *PyPI*, 2023. [Online]. Available: https://pypi.org/project/pyttsx3/

[5] M. Gupta, “Building Real-Time Speech Recognition System with Python,” *Towards Data Science*, 2021. [Online]. Available: https://towardsdatascience.com/building-real-time-speech-recognition-system-in-python-7a74a4a06f1a

[6] Python Software Foundation, “SpeechRecognition 3.10.0,” *Python Package Index (PyPI)*, 2023. [Online]. Available: https://pypi.org/project/SpeechRecognition/

[7] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd ed., Pearson, 2023.

[8] A. Rosebrock, “Real-time speech recognition with Python and Google Speech API,” *PyImageSearch*, 2019. [Online]. Available: https://pyimagesearch.com

[9] J. Lin, “Introduction to Sentiment Analysis: Methods and Applications,” *IEEE Computational Intelligence Magazine*, vol. 12, no. 2, pp. 26–34, May 2017.

[10] B. Liu, *Sentiment Analysis and Opinion Mining*, Morgan & Claypool Publishers, 2012.