



SPEECH RECOGNITION TECHNIQUE

NAAN MUDHALVAN - STUDENT IMPLEMENTATION PROGRAM

Voice enabled technologies

Submitted By,

NAME: Swapna Devi K

Register Number: 820622104079

Email ID :swapnaswapz82@gmail.com

Mobile No: +91 9500819545

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ARASU ENGINEERING COLLEGE

KUMBAKONAM-612501

Table of content

S.No.	PROJECT NAME	Page No.
1	Speech recognition for medical transcriptions	3
2	Voice assistance (E.g., Siri, Alexa)	8
3	Speech based data entry systems	17

Speech recognition for medical transcriptions

Introduction

In recent years, the integration of speech recognition technology into healthcare has transformed the way medical professionals document patient information. Speech recognition for medical transcriptions offers a faster, more efficient, and accurate alternative to traditional manual documentation methods. By converting spoken language into written text, this technology helps reduce administrative burdens, allowing healthcare providers to focus more on patient care. With the growing complexity of clinical documentation and the demand for real-time data entry, speech recognition systems have become increasingly vital. This paper explores the role, benefits, challenges, and future prospects of speech recognition in the domain of medical transcription.

Problem Statement

Medical transcription is a critical yet time-consuming task that requires high accuracy and efficiency. Traditional manual transcription methods are prone to human error, delays, and increased administrative workload, which can negatively impact the quality

of patient care. Although speech recognition technology has shown promise in automating transcription, its application in the medical field faces unique challenges such as diverse accents, medical jargon, background noise, and speaker variability. There is a pressing need for a robust, domain-specific speech recognition system that can accurately and reliably transcribe medical dictations in real-time, thereby improving documentation efficiency and supporting better clinical decision-making.

Objectives

- * To develop a speech recognition system optimized for accurately transcribing medical dictations.
- * To ensure the system can recognize and process medical terminology and jargon effectively.
- * To improve the speed and efficiency of medical transcription compared to manual methods.
- * To reduce human errors in documentation by leveraging automated transcription.
- * To enable real-time or near real-time transcription to support timely clinical decision-making.
- * To handle variations in speech such as accents, tone, and background noise in medical environments.
- * To explore integration with electronic health record (EHR) systems for seamless workflow.

Methodology

Data Collection: Gather a diverse dataset of medical speech recordings, including doctor-patient interactions and clinical dictations.

Preprocessing: Clean and preprocess audio data by removing noise, normalizing volume, and segmenting speech for better recognition accuracy.

Speech-to-Text Conversion: Use or train a speech recognition model (e.g., DeepSpeech, Google Speech API, or a custom deep learning model) to convert audio into text.

Medical Vocabulary Integration: Incorporate a domain-specific medical lexicon to improve recognition of medical terms and abbreviations.

Natural Language Processing (NLP): Apply NLP techniques to structure, format, and correct the transcribed text for readability and accuracy.

Evaluation: Test the system using real-world medical audio samples and measure performance based on accuracy, word error rate (WER), and processing time.

System Integration: Optionally integrate the transcription system with electronic health record (EHR) platforms for seamless documentation.

Program flow

Input data

```
import speech_recognition as sr

from datetime import datetime

import time

# Define 3 medical transcription prompts
medical_datasets = [

    "1. Dictate the progress note for a patient admitted for COPD exacerbation.",

    "2. Provide a discharge summary for a patient recovering from dengue fever.",

    "3. Record a new patient consultation for chronic lower back pain.",

]

def transcribe_medical_note(index, prompt):

    recognizer = sr.Recognizer()

    print(f"\n--- Dataset {index + 1} ---")
    print(f"Prompt: {prompt}")
```

```

    input("🎤 Press Enter when you're ready to begin dictation
(You will have 300 seconds)...")

with sr.Microphone() as source:
    print("🔊 Calibrating for background noise...")
    recognizer.adjust_for_ambient_noise(source, duration=3)

    print("🔴 Start speaking now...")
    try:
        # Listen for up to 5 minutes
        audio = recognizer.listen(source, timeout=5,
phrase_time_limit=300)
        print("🔴 Recording finished. Transcribing...")

        # Transcribe using Google Speech Recognition
        transcription = recognizer.recognize_google(audio)
        print("✅ Transcription successful.")
        print("📝", transcription)

        # Save to a file with timestamp
        timestamp = datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        filename = f"medical_transcription_{index +
1}_{timestamp}.txt"

        with open(filename, "w") as file:
            file.write("Medical Dictation Transcription\n")
            file.write("Date: " + datetime.now().strftime("%Y-%m-%d
%H:%M:%S") + "\n\n")
            file.write("Prompt: " + prompt + "\n\n")
            file.write("Transcription:\n" + transcription)

        print(f"📁 Transcription saved to: {filename}")

    except sr.WaitTimeoutError:
        print("❌ No speech detected. You took too long to start.")
    except sr.UnknownValueError:
        print("❌ Could not understand the audio.")
    except sr.RequestError as e:
        print(f"❌ Request to Google API failed: {e}")

# Run the transcription for each dataset

```

```

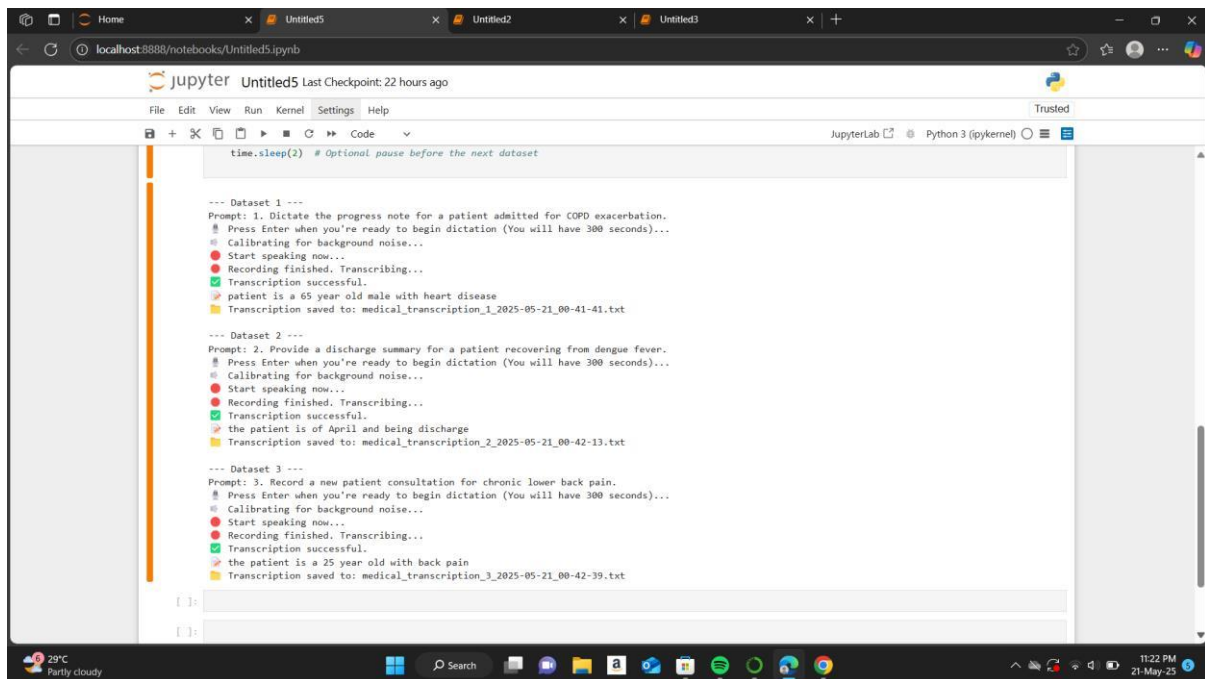
for idx, prompt in enumerate(medical_datasets):

    transcribe_medical_note(idx, prompt)

    time.sleep(2) # Optional pause before the next dataset

```

Output



```

time.sleep(2) # Optional pause before the next dataset

--- Dataset 1 ---
Prompt: 1. Dictate the progress note for a patient admitted for COPD exacerbation.
  Press Enter when you're ready to begin dictation (You will have 300 seconds)...
  Calibrating for background noise...
  Start speaking now...
  Recording finished. Transcribing...
  Transcription successful.
  patient is a 65 year old male with heart disease
  Transcription saved to: medical_transcription_1_2025-05-21_00-41-41.txt

--- Dataset 2 ---
Prompt: 2. Provide a discharge summary for a patient recovering from dengue fever.
  Press Enter when you're ready to begin dictation (You will have 300 seconds)...
  Calibrating for background noise...
  Start speaking now...
  Recording finished. Transcribing...
  Transcription successful.
  the patient is of April and being discharge
  Transcription saved to: medical_transcription_2_2025-05-21_00-42-13.txt

--- Dataset 3 ---
Prompt: 3. Record a new patient consultation for chronic lower back pain.
  Press Enter when you're ready to begin dictation (You will have 300 seconds)...
  Calibrating for background noise...
  Start speaking now...
  Recording finished. Transcribing...
  Transcription successful.
  the patient is a 25 year old with back pain
  Transcription saved to: medical_transcription_3_2025-05-21_00-42-39.txt

```

Conclusion

Speech recognition technology has the potential to revolutionize medical transcription by offering a faster, more accurate, and efficient alternative to manual documentation. By leveraging advanced machine learning models and incorporating domain-specific medical vocabulary, speech recognition systems can significantly reduce the workload on healthcare professionals and improve the quality of clinical documentation. Despite existing challenges such as background noise, accent variation, and complex medical terminology, ongoing advancements in artificial intelligence and natural language processing continue to enhance the reliability and performance of these systems. As integration with electronic health records becomes more seamless, speech recognition is poised to become an essential tool in modern healthcare settings.

Voice assistance (E.g., Siri, Alexa)

Introduction

Voice assistants like Siri, Alexa, and Google Assistant have revolutionized the way we interact with technology, making tasks more convenient and accessible through voice commands. These intelligent systems use natural language processing (NLP) and artificial intelligence (AI) to understand, interpret, and respond to a wide range of spoken requests. Siri, developed by Apple, was one of the pioneers in voice assistance, followed by Amazon's Alexa, which powers a range of smart devices in homes, and Google Assistant, known for its integration with Google's vast search capabilities. These voice assistants can perform a variety of tasks, from answering questions and setting reminders to controlling smart home devices and playing music, all hands-free. As they continue to evolve, they are becoming increasingly sophisticated, adapting to user preferences, improving accuracy, and expanding their functionality to provide a more seamless and personalized experience.

Problem Statement

Despite the popularity and widespread use of voice assistants like Siri, Alexa, and Google Assistant, several challenges persist that hinder their full potential. Key issues include **accuracy in speech recognition**, especially in noisy environments or with diverse accents and dialects, leading to misinterpretation of commands. **Contextual understanding** remains a limitation, as these assistants often struggle with complex, multi-step tasks or nuanced conversations, which can lead to irrelevant or incorrect responses. **Privacy and security concerns** also arise, as these devices constantly listen to ambient sounds, raising questions about data collection, storage, and potential breaches. Additionally, these voice assistants often lack **specialized knowledge** or the ability to handle domain-specific queries effectively, limiting their utility. As voice assistants become more integrated into daily life, addressing these problems is essential for improving user trust, satisfaction, and functionality.

Objectives

The objectives of voice assistants like Siri, Alexa, and Google Assistant are:

1. **Improve Speech Recognition:** Enhance the accuracy and reliability of voice recognition across different accents, languages, and noisy environments.
2. **Enhance Contextual Understanding:** Develop the ability to interpret and respond to complex, multi-step commands, and conversations with better contextual awareness.
3. **Expand Functionality:** Increase the range of tasks that voice assistants can perform, from everyday tasks to specialized queries in various domains.
4. **Ensure Privacy and Security:** Implement robust data protection measures to ensure user privacy, secure data storage, and prevent unauthorized access.
5. **Provide Personalized Experiences:** Tailor responses and interactions based on user preferences, behavior, and previous interactions to offer a more customized experience.
6. **Improve Integration with Smart Devices:** Enhance compatibility with a broader range of smart home devices, apps, and external services to streamline user interactions.
7. **Increase Accuracy in Multi-language Support:** Improve the assistant's ability to understand and communicate in multiple languages for a diverse global user base.

Methodology

1. **Data Collection:** Collect diverse voice data from users, including different accents, languages, and contextual scenarios, to build comprehensive datasets for training the system.
2. **Speech Recognition:** Implement Automatic Speech Recognition (ASR) algorithms to convert spoken words into text, refining the models to handle various speech patterns and background noise effectively.

3. **Natural Language Processing (NLP):** Use NLP techniques to process and interpret the text, enabling the assistant to understand commands, answer questions, and engage in more complex conversations.
4. **Contextual Understanding:** Develop models that analyze the context of interactions, allowing the assistant to recognize and respond accurately to follow-up questions or multi-step requests.
5. **Machine Learning and AI:** Apply machine learning to continually enhance the assistant's performance by learning from user feedback and interactions to offer more accurate and personalized responses.
6. **Integration with APIs and Devices:** Connect the voice assistant with third-party services, applications, and smart home devices to execute tasks such as controlling lights, playing music, or providing real-time information.
7. **Privacy and Security:** Employ strong encryption, anonymization, and authentication methods to protect user data and ensure secure interactions with the assistant.
8. **User Feedback Loop:** Collect user feedback to identify issues and areas for improvement, ensuring the assistant evolves to meet user needs and expectations.
9. **Testing and Optimization:** Conduct extensive testing in various real-world environments (e.g., noisy spaces, different accents) to fine-tune performance and improve speech recognition accuracy.

Program flow

Input Data

```
import speech_recognition as sr

import pyttsx3

import datetime

import webbrowser

import os

import wikipedia

import pywhatkit

import psutil

import random

# Initialize text-to-speech engine

engine = pyttsx3.init() voices = engine.getProperty('voices')
engine.setProperty('voice', voices[1].id) engine.setProperty('rate',
160)

def speak(text):

    print("Assistant:", text)

    engine.say(text)

    engine.runAndWait()

def wish_user():

    hour = datetime.datetime.now().hour

    if hour < 12:

        speak("Good morning!")

    elif hour < 18:

        speak("Good afternoon!")

    else:

        speak("Good evening!")
```

```

        speak("Hi, I am your assistant. What can I do for you?")

def take_command():
    r = sr.Recognizer()

    with sr.Microphone() as source:
        print("Listening...")

        r.pause_threshold = 1

        r.energy_threshold = 300 audio =

        r.listen(source)

    try:
        print("Recognizing...")

        query = r.recognize_google(audio, language="en-in")

        print("You said:", query)

    except Exception:
        speak("Sorry, could you repeat that?")

        return "None"

    return query.lower()

def tell_joke():
    jokes = [

        "Why don't scientists trust atoms? Because they make up everything!",

        "Why was the math book sad? Because it had too many problems.",

        "Why did the computer go to the doctor? Because it had a virus."

    ]

    speak(random.choice(jokes))

```

```

def battery_status():
    battery = psutil.sensors_battery()
    percent = battery.percent
    speak(f"Battery is at {percent} percent")
    if battery.power_plugged:
        speak("Your laptop is charging.")
    else:
        speak("Your laptop is not charging.")

def run_assistant():
    wish_user()
    while True:
        command = take_command()
        if 'time' in command:
            time=datetime.datetime.now().strftime('%I:%M %p')
            speak(f"The time is {time}")

        elif 'date' in command:
            today = datetime.date.today()
            speak(f"Today's date is {today.strftime('%B %d, %Y')}")

        elif 'open google' in command
            speak("Opening Google")
            webbrowser.open("https://www.google.com")

        elif 'open youtube' in command:
            speak("Opening YouTube")

```

```

        webbrowser.open("https://www.youtube.com")
elif 'open gmail' in command:

    speak("Opening Gmail")

    webbrowser.open("https://mail.google.com")

elif 'open stack overflow' in command:
    speak("Opening Stack Overflow")

    webbrowser.open("https://stackoverflow.com")

elif 'play music' in command:
    music_dir='C:\\Users\\YourName\\Music'

    songs = os.listdir(music_dir)
    if songs:
        os.startfile(os.path.join(music_dir, songs[0]))
else:

    speak("No music found in your directory")
elif 'wikipedia' in command:

    speak("Searching Wikipedia...")

    query = command.replace("wikipedia", "")

    try:

        result = wikipedia.summary(query, sentences=2)

        speak("According to Wikipedia")

        speak(result)

    except:

        speak("Sorry, I couldn't find anything on Wikipedia.")

elif 'joke' in command:

    tell_joke()

elif 'battery' in command or 'status' in command:
    battery_status()

```

```

elif 'search' in command:
    speak("What do you want to search?")
    search_query = take_command()
    if search_query != "None":
        pywhatkit.search(search_query)
        speak(f"Here are the results for {search_query}")

elif 'whatsapp' in command:
    speak("What message should I send?")
    msg = take_command()
    speak("Sending message on WhatsApp in 1 minute")
    # Example: Replace with your friend's number
    pywhatkit.sendwhatmsg('+91xxxxxxxxxx', msg,
datetime.datetime.now().hour, datetime.datetime.now().minute +
1)

elif 'screenshot' in command:
    import pyautogui
    screenshot = pyautogui.screenshot()
    file_path = os.path.join(os.getcwd(), "screenshot.png")
    screenshot.save(file_path)
    speak("Screenshot taken and saved.")

elif 'weather' in command:
    speak("Weather functionality is coming soon. You can
integrate an API like OpenWeatherMap.")

elif 'hello' in command or 'hi' in command:
    speak("Hello! How are you doing?")

elif 'who are you' in command:
    speak("I am a voice assistant built in Python.")

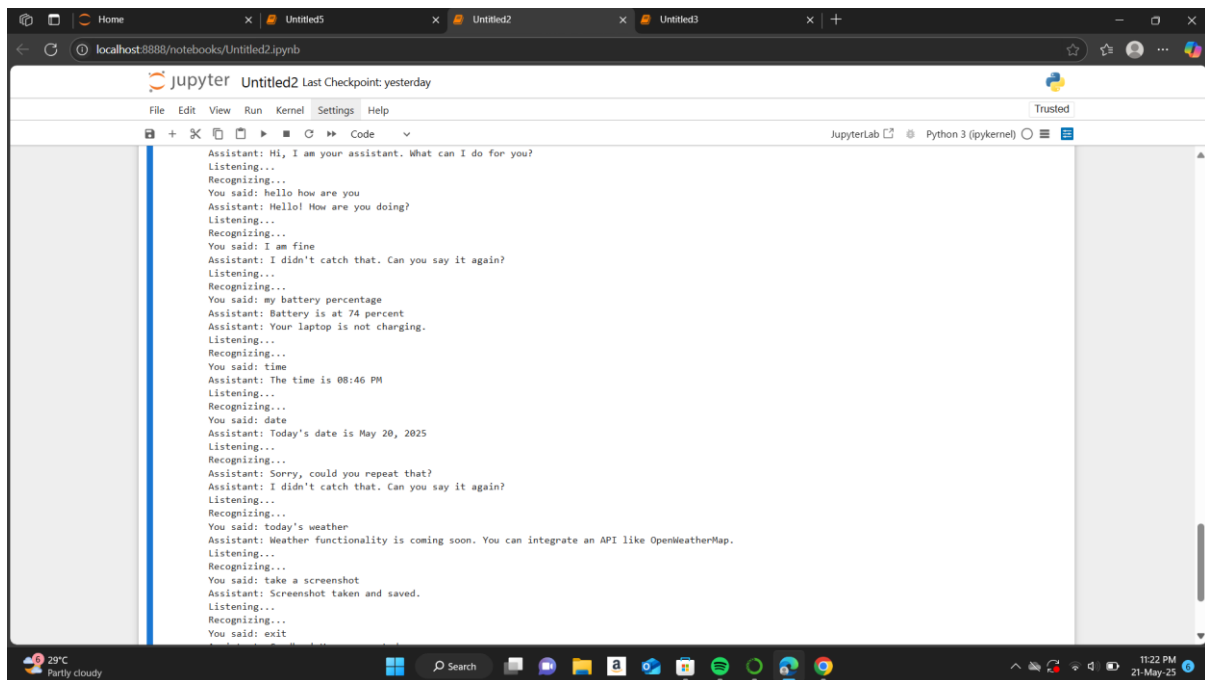
elif 'exit' in command or 'quit' in command or 'stop' in
command:
    speak("Goodbye! Have a great day.")
    break

else:
    speak("I didn't catch that. Can you say it again?")

```

```
if name == "main":  
  
    run_assistant()
```

Output



The screenshot shows a JupyterLab window with a single notebook titled 'Untitled2'. The notebook contains a Python script that simulates a voice assistant. The output of the script is displayed in the right-hand pane, showing a series of interactions between a user and an assistant. The assistant responds to various commands, including greetings, status checks, and requests for information. The interface includes a menu bar with options like File, Edit, View, Run, Kernel, Settings, and Help. The bottom status bar shows the system temperature as 23°C and the time as 11:22 PM on May 21, 2025.

```
Assistant: Hi, I am your assistant. What can I do for you?  
Listening...  
Recognizing...  
You said: hello how are you  
Assistant: Hello! How are you doing?  
Listening...  
Recognizing...  
You said: I am fine  
Assistant: I didn't catch that. Can you say it again?  
Listening...  
Recognizing...  
You said: my battery percentage  
Assistant: Battery is at 74 percent  
Assistant: Your laptop is not charging.  
Listening...  
Recognizing...  
You said: time  
Assistant: The time is 08:46 PM  
Listening...  
Recognizing...  
You said: date  
Assistant: Today's date is May 20, 2025  
Listening...  
Recognizing...  
Assistant: Sorry, could you repeat that?  
Assistant: I didn't catch that. Can you say it again?  
Listening...  
Recognizing...  
You said: today's weather  
Assistant: Weather functionality is coming soon. You can integrate an API like OpenWeatherMap.  
Listening...  
Recognizing...  
You said: take a screenshot  
Assistant: Screenshot taken and saved.  
Listening...  
Recognizing...  
You said: exit
```

Conclusion

In conclusion, voice assistants like Siri, Alexa, and Google Assistant have significantly transformed the way we interact with technology, providing convenience, efficiency, and hands-free functionality. While they have made substantial progress in speech recognition, contextual understanding, and smart device integration, challenges such as privacy concerns, limitations in contextual awareness, and accuracy in diverse environments still remain. As technology continues to evolve, these voice assistants are expected to become even more accurate, secure, and capable of handling complex tasks. By addressing current limitations and enhancing user personalization, voice assistants will continue to play a crucial role in simplifying daily tasks and improving the overall user experience.

Speech-based data entry systems

Introduction

Speech-based data entry systems are innovative technologies that enable users to input data into a computer or device using voice commands instead of traditional keyboard or manual methods. These systems leverage speech recognition and natural language processing (NLP) technologies to convert spoken words into text, making data entry faster, more efficient, and accessible. Speech-based systems are particularly beneficial in environments where hands-free operation is required, such as in healthcare, customer service, and for individuals with disabilities. By allowing users to dictate information verbally, these systems can improve productivity, reduce errors in data entry, and provide a more inclusive approach to interacting with digital platforms. As speech recognition technologies continue to evolve, speech-based data entry is becoming an increasingly important tool in various industries.

Problem Statement

Despite the growing popularity and potential of speech-based data entry systems, several challenges hinder their widespread adoption and effectiveness. Key issues include **accuracy limitations**, especially in noisy environments or with diverse accents and dialects, leading to misinterpretation of voice commands or data. **Speech recognition errors** may result in incorrect data entry, requiring time-consuming corrections. Additionally, **language and context comprehension** remain limitations, as these systems often struggle to handle complex commands, technical terminology, or domain-specific inputs. **Privacy and security concerns** also arise, as sensitive data is processed through voice, raising questions about data protection and unauthorized access. Furthermore, **user adaptation** can be a challenge, as some users may be uncomfortable or less efficient when relying on voice commands instead of traditional input methods. Addressing these issues is crucial to improving the reliability, accuracy, and security of speech-based data entry systems for broad-scale implementation.

Objectives

The objectives of speech-based data entry systems are:

1. **Enhance Accuracy:** Improve speech recognition accuracy, particularly in noisy environments and with diverse accents, dialects, and languages.
2. **Increase Efficiency:** Enable faster data entry by allowing users to speak rather than type, reducing time and effort required for manual input.
3. **Improve Contextual Understanding:** Develop systems that can handle complex commands and domain-specific terminology with higher accuracy and relevance.
4. **Ensure Data Privacy and Security:** Implement robust security protocols to protect sensitive data and maintain user privacy during voice processing.
5. **Provide Accessibility:** Make data entry more accessible for individuals with disabilities or those unable to use traditional input devices like keyboards or touchscreens.
6. **Seamless Integration:** Enable smooth integration with existing software and applications to support various use cases in different industries, including healthcare, customer service, and business.
7. **User Adaptation and Comfort:** Design intuitive and user-friendly systems that allow for easy adaptation and comfort when using voice-based input methods.
8. **Scalability and Flexibility:** Develop systems that can scale to handle large volumes of data and accommodate various industry-specific needs.

Methodology

1. **Data Collection:** Gather diverse voice datasets, including different accents, dialects, and languages, to train the speech recognition models effectively. This data may include various environmental noise levels to simulate real-world conditions.
2. **Speech Recognition:** Implement advanced Automatic Speech Recognition (ASR) systems, using deep learning models such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs) to convert spoken language into text with high accuracy.

3. **Natural Language Processing (NLP):** Integrate NLP techniques to process and understand the context of the speech input. This allows the system to handle complex commands and distinguish between similar-sounding words or phrases.
4. **Noise Reduction and Signal Processing:** Apply noise filtering and signal enhancement techniques to improve the accuracy of speech recognition in noisy environments, ensuring reliable performance across different settings.
5. **Voice Command Mapping:** Develop a system for mapping spoken commands to specific actions or data entry fields. This includes defining how to recognize various commands, handle corrections, and manage domain-specific terminology.
6. **Security and Privacy Measures:** Implement encryption and anonymization techniques to protect user data during transmission and storage, ensuring privacy and compliance with relevant regulations (e.g., GDPR).
7. **User Interface Design:** Create an intuitive user interface that integrates with the speech recognition system, offering feedback and correction options for users during data entry.
8. **Machine Learning Optimization:** Continuously improve the system by training it with new data, adapting to users' speech patterns, preferences, and industry-specific needs to enhance performance over time.
9. **Integration with Existing Systems:** Ensure seamless integration with existing software and databases to allow easy data entry and storage across different platforms, such as CRM systems, healthcare management software, or inventory systems.

10. **Testing and Evaluation:** conduct rigorous testing in various real-world environments to identify areas for improvement, validate the system's accuracy and ensure it performs reliably across different accents, speech styles and noise conditions.

Program flow

Input data

```
import speech_recognition as sr

import pyttsx3

import csv

import os

# Initialize text-to-speech engine

engine = pyttsx3.init()

engine.setProperty('rate', 160)

def speak(text):

    print("Assistant:", text)

    engine.say(text)

    engine.runAndWait()

def listen():

    recognizer = sr.Recognizer()

    with sr.Microphone() as source:

        print("🎤 Listening...")

        recognizer.adjust_for_ambient_noise(source)

        audio = recognizer.listen(source)

    try:

        print("🧠 Recognizing speech...")
```

```

        query = recognizer.recognize_google(audio)
        print("You said:", query)
        return query.lower()
    except sr.UnknownValueError:
        speak("Sorry, I didn't catch that.")
        return ""
    except sr.RequestError:
        speak("Network error.")
        return ""

# Define data fields you want to collect

fields = ["name", "age", "gender", "email", "symptoms", "diagnosis"]
data_entry = {field: "" for field in fields}

def fill_field_from_speech():

    speak("Start speaking the details. Say one field at a time,
    like 'Name is John' or 'Age is 45'.")

    while True:
        user_input = listen()
        if not user_input:
            continue

        # Exit condition
        if "done" in user_input or "stop" in user_input or "exit"
in user_input:
            break

        # Fill recognized fields
        for field in fields:
            if field in user_input:
                try:
                    value =
user_input.split(field)[1].strip(" is:") # Get value after 'field'
                    data_entry[field] = value
                    speak(f"{field.capitalize()} recorded as
{value}")

                except IndexError:
                    speak(f"Could not extract value for
{field}")

```

```

speak("Finished recording data.")

def save_to_csv(data_dict, filename="voice_data.csv"): file_exists =
os.path.isfile(filename)

    with open(filename, mode='a', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=fields)
        if not file_exists:
            writer.writeheader()
        writer.writerow(data_dict)

    speak("Data saved successfully.")

# Main function

def main():

    speak("Welcome to the speech-based data entry system.")
    fill_field_from_speech()

    print("\nCollected Data:")
    for key, value in data_entry.items():
        print(f"{key.capitalize()}: {value}")

    save_to_csv(data_entry)
    speak("You may close the application now.")

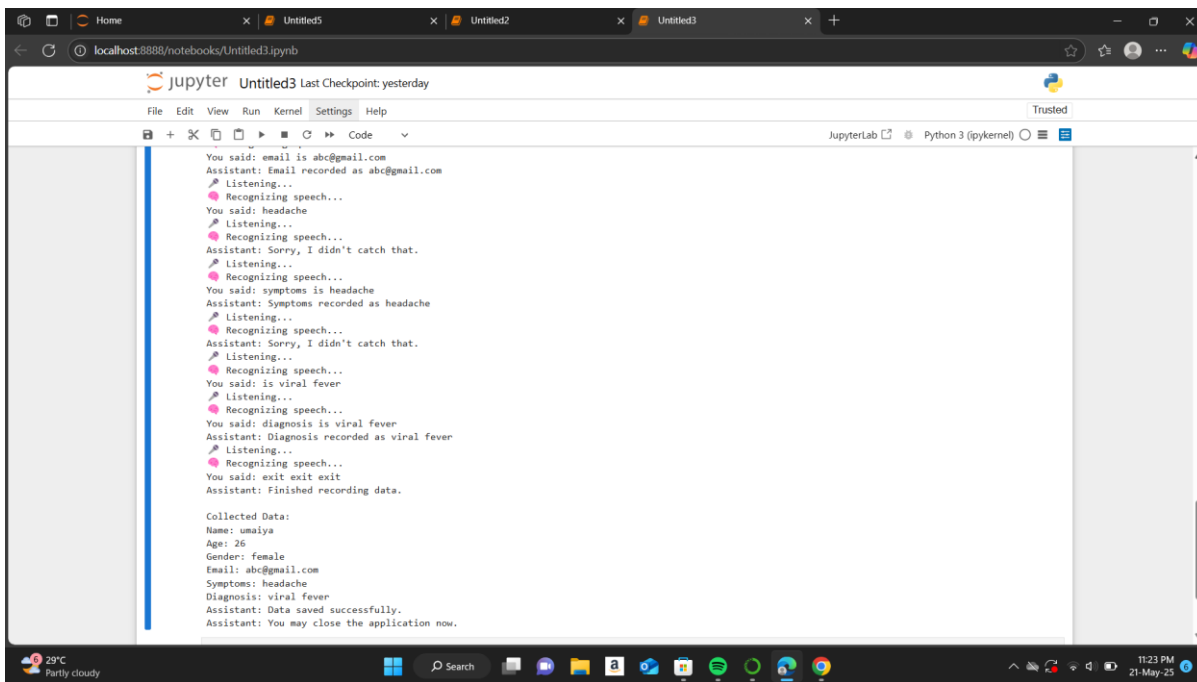
# Run

If__ name__ == "main":

    main()

```

Output



Conclusion

Speech-based data entry systems represent a significant advancement in how we interact with technology, offering a more efficient, accessible, and hands-free approach to data input. These systems hold the potential to greatly improve productivity, especially for individuals with disabilities or those working in environments where traditional typing is impractical. However, challenges such as improving accuracy, handling complex commands, ensuring privacy, and minimizing errors in noisy environments remain. As speech recognition technology continues to evolve, addressing these issues will be key to realizing the full potential of speech-based data entry systems across various industries. With ongoing advancements in artificial intelligence and machine learning, these systems are poised to become an integral part of modern workflows.