# VOICE CRAFT DESKTOP ASSISTANT

*Srinivasa Rao Bendi[1], Gannamaneni Vignesh[2], Bandaru Meghana[3], Viswanadhula Jaya Sri[4], Usman Shariff[5]*

*[1]Associate Professor, Computer Science Engineering, GITAM University, Visakhapatnam, Andhra Pradesh*

*[2,3,4,5]UG – Computer Science Engineering, GITAM University, Visakhapatnam, Andhra Pradesh*

*Emails: sbendi@gitam.edu[1], vigneshgannamaneni54321@gmail.com[2], mghna03@gmail.com[3], viswanadhulajayasri@gmail.com[4], author2@gmail.com[5]*

## Abstract

In modern human-computer interaction, natural language processing (NLP) plays a crucial role, especially in voice-controlled interfaces. This study examines key NLP components such as intent classification and entity recognition, alongside ChatGPT integration, to enhance system capabilities. Intent classification categorizes user intentions, while entity recognition identifies specific entities mentioned in user input.

Integration of ChatGPT enriches NLP systems, enabling them to intelligently respond to ambiguous queries. By leveraging advanced AI and NLP, ChatGPT enhances user experience by providing informative responses to various inquiries. This amalgamation of intent classification, entity recognition, and ChatGPT integration improves the efficiency of voice-operated systems, offering users a more intuitive interaction.

This project aims to explore the significance of these components in advancing NLP systems and revolutionizing human-computer interaction. By understanding and integrating these components effectively, voice-controlled interfaces can provide users with an innovative and efficient way to interact with technology, leveraging machine learning, NLP, and artificial intelligence.

**Keywords:** Natural Language Processing, Intent Classification, Entity Recognition, ChatGPT Integration, Voice-Controlled Interfaces, Machine Learning, User Interaction, Speech Recognition, and Text Processing

## 1. INTRODUCTION

Intelligent personal assistants, or voice assistants, have revolutionized our relationship with technology and are now an essential part of our digital environment. These software agents are skilled at following spoken instructions to complete activities and offer services to users. They can also understand human speech and answer with a synthesized voice. The reason for their broad adoption is their hands-free feature and convenience, which makes using technology easier.

Users may utilize voice commands to accomplish a wide range of operations with our AI-powered desktop assistant, from basic questions to carrying out particular actions. As an example, users can ask the assistant to look up information on Wikipedia, find out the time, or even open programs like Internet Explorer, YouTube, or Visual Studio Code. Additionally, users can instruct the assistant to perform more sophisticated tasks, such as writing and sending emails to particular recipients, demonstrating the adaptability and usefulness of these helpers in daily situations.

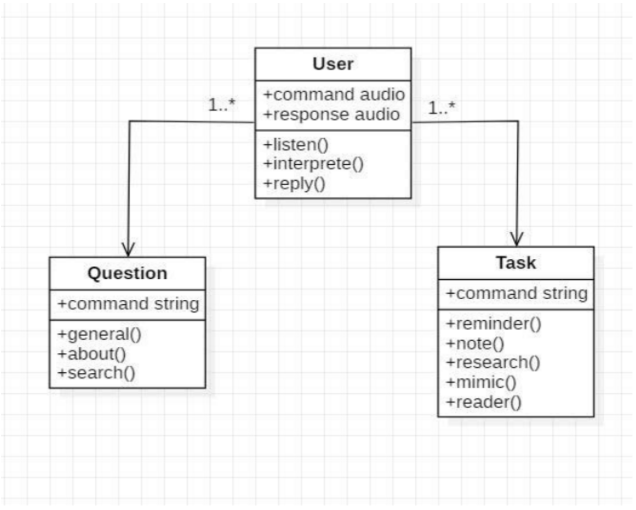
Voice assistants work by use of several complex mechanisms that allow them to understand and react to human orders with ease. Speech recognition is the first step in the process, where spoken commands are translated into text. Natural language processing algorithms then examine this material to determine its purpose and meaning. Then, using this knowledge, backend processing carries out the required operations, such as controlling linked devices or obtaining data from databases. After that, the assistant responds and relays its findings back to the user, gradually gaining new skills and knowledge.

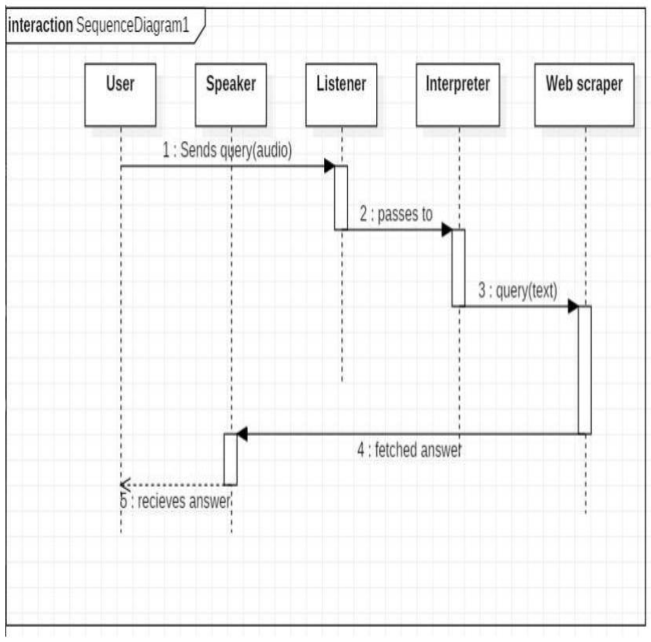
The Speech Application Programming Interface (SAPI) is a vital tool that makes it easier to design and incorporate speech detection and synthesis

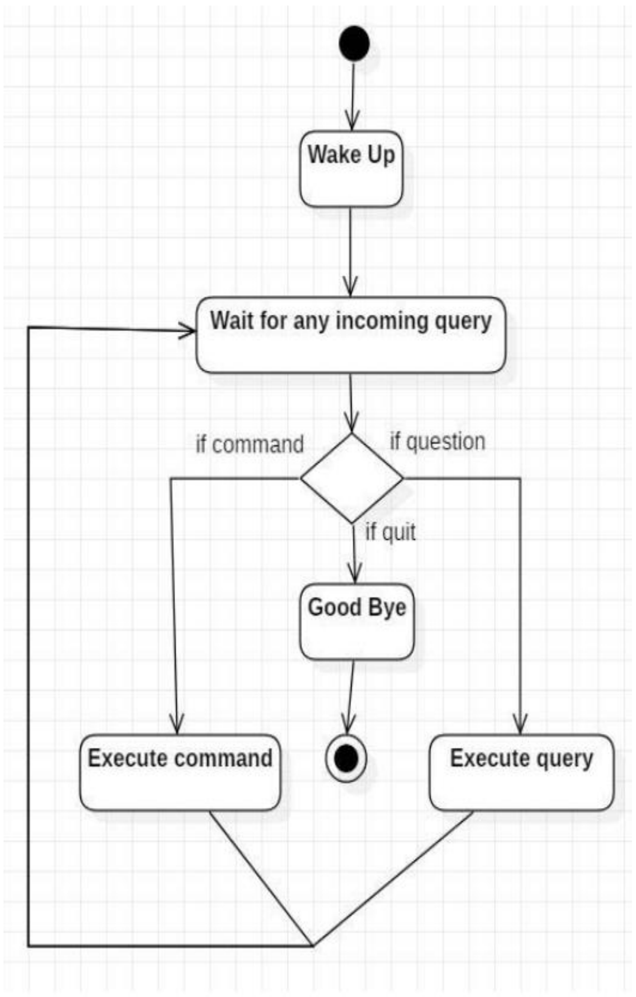
features into Windows programs. Microsoft's Speech API (SAPI) is an essential development tool for software developers that aims to simplify the incorporation of speech-related features into different applications.
SAPI has been through multiple modifications over the years; it is frequently included in Windows operating systems or supplied as a component of the Speech Software Development Kit (SDK). Its incorporation into well-known programs like Microsoft Office, Microsoft Agent, and Microsoft Speech Server attests to its adaptability and broad use.
Our project makes use of SAPI5, the most recent version of the API, to easily incorporate voice synthesis and recognition features into Windows programs. Like its predecessors, SAPI5 provides a standardized set of interfaces that are available in multiple programming languages, enabling developers to easily integrate speech-related functionality into their applications.
Additionally, the flexibility of SAPI enables external businesses to create their own Text-To-Speech and Speech Recognition engines or adapt pre-existing ones, to interface with the API. This promotes the development of a wide variety of speech-related applications that are customized to meet particular user needs and linguistic requirements.
The ability of speech-to-text, or speech recognition, is essential to voice assistant operation. It entails the process of turning spoken words into text so that users can give spoken commands that are transcribed into text. This technology makes the ability to produce written text quickly and easily possible, which does away with the necessity for manual typing.
Furthermore, as speech-to-text software facilitates simpler communication and easier access to digital content, it is especially helpful for those with disabilities that could make it difficult for them to operate a keyboard.
To sum up, voice assistants that possess the ability to recognize and synthesize speech mark a noteworthy development in the field of human-computer interaction, providing consumers with a smooth and simple way to engage with technology.
Using technologies such as SAPI makes it easier to incorporate these features into different applications, which improves user experience, productivity, and accessibility in a variety of contexts.

## 2. METHOD

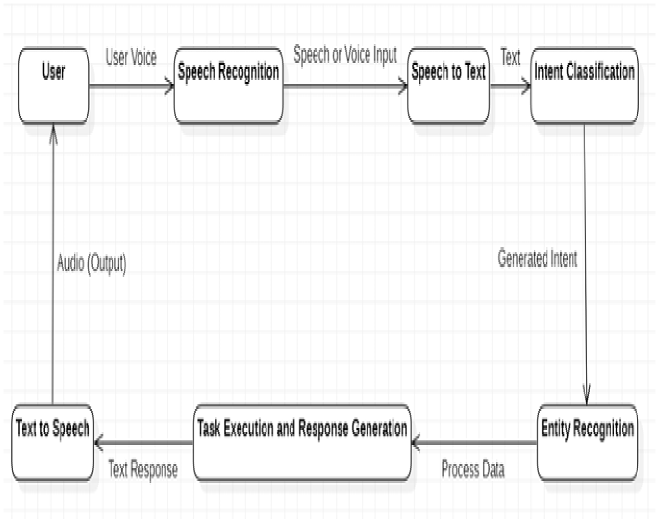
### 2.1. UML:







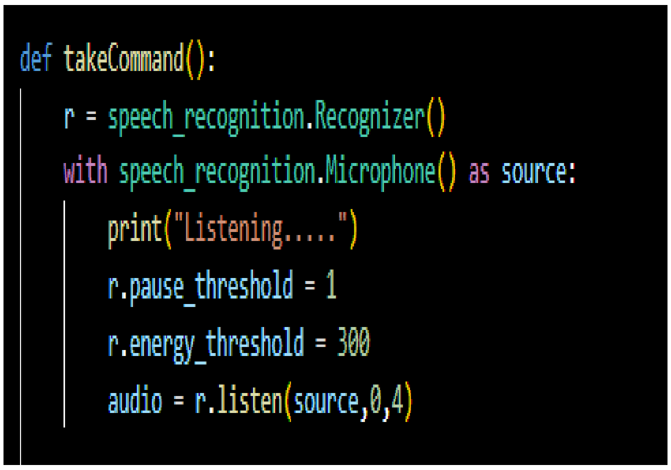
### 2.2. Working Model



### 2.3. Speech Recognition

Speech recognition facilitates hands-free device operation, automates transcription tasks, aids communication for individuals with disabilities, and enhances customer support and healthcare documentation.

Firstly, we will install a speech_recognition package to record audio from the microphone and perform speech recognition using Google's speech recognition service.
*speech_recognition.Recognizer():* This function recognizes the speech input given by the user from the microphone.



```
def takeCommand():
    r = speech_recognition.Recognizer()
    with speech_recognition.Microphone() as source:
        print("Listening.....")
        r.pause_threshold = 1
        r.energy_threshold = 300
        audio = r.listen(source,0,4)
```

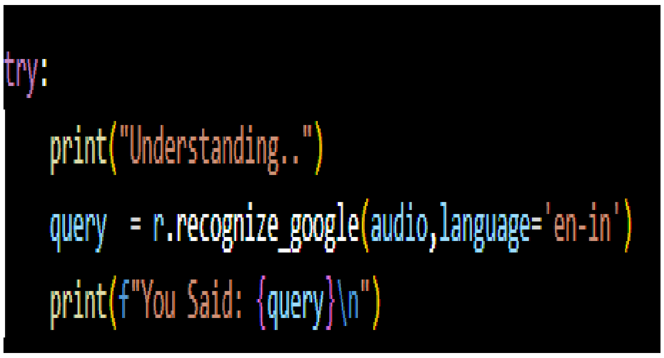
Opens the microphone as a source for audio input.
Prints "Listening...." to indicate that the program is ready to listen for audio input.
Sets the pause threshold and energy threshold for the recognizer object. These thresholds determine when the recognizer should consider speech to have ended.
Listens for audio input for up to 4 seconds, with no minimum pause between words (pause_threshold = 1 and energy_threshold = 300).

### 2.4. Speech-To-Text

Speech recognition begins by capturing the sound energy produced by the speaker using a microphone. This sound energy is then transformed into electrical signals, transitioning from analog to digital format. Once digitized, sophisticated algorithms analyze the digital signal to identify patterns and features corresponding to spoken words.

These patterns are compared against a vast database of linguistic models to interpret and transcribe the speech into text accurately.

*r.recognize_google(audio, language='en-in'):* Recognize the speech in the captured audio (audio) and convert it to text



```
try:
    print("Understanding..")
    query  = r.recognize_google(audio,language='en-in')
    print(f"You Said: {query}\n")
```

### 2.5. Intent Classification

*User Input Processing:* The user's input is captured through speech recognition and converted into text.

*Intent Identification:* Conditional statements analyze the input to identify specific intents, such as waking up Jarvis or controlling media playback.

*Execution of Intent-Specific Functionality:* Once the intent is identified, the code executes corresponding functionality, like greeting the user or controlling video playback.

**Intent Classification Example:**

*Intent:* Controlling Video Playback

*Query:* "Pause the video."

*Implementation*: Recognize commands like "pause," "play," or "mute," and execute actions to control video playback accordingly.

### 2.6. Entity Recognition

Entity recognition entails the identification and extraction of particular entities or information fragments from the user's input. In the provided code, entity recognition may not be explicitly implemented as a separate component. However, certain intents may require extracting additional information or parameters from the user's query. Here's how entity recognition could be incorporated:

**Task-specific Functions:**

Some task-specific functions may require additional parameters to perform their tasks effectively.

For example, when setting an alarm, the user may need to specify the time for the alarm to be set. In such cases, the code may extract relevant entities such as time, location, or other parameters from the user's query to provide context for the task execution.

**Entity Recognition Example:**

Alarm Setting with Specific Time:

*Intent:* The user wants to set an alarm for a specific time.

*Example Query:* "Set an alarm for 7 AM."

*Implementation:* Extract the time entity "7 AM" from the query and pass it as a parameter to the function for setting alarms.

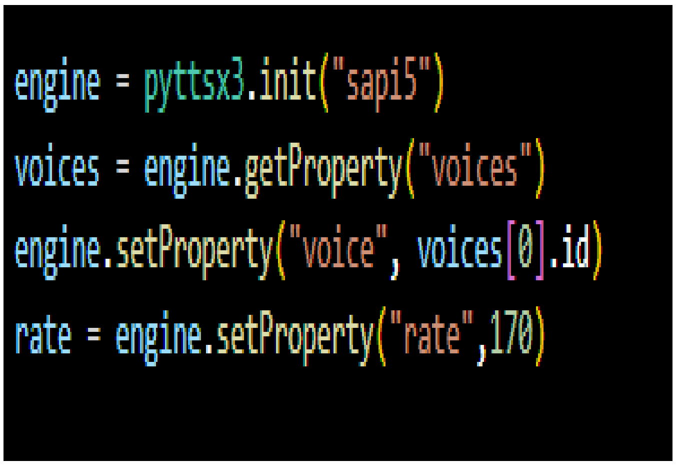
### 2.7. Task Execution and Response Generation

*Task Execution*: Upon identifying and validating the user's intent and entities, the system proceeds to execute the relevant task or action. This process entails invoking specific functions, modules, or APIs to carry out various tasks, such as retrieving data from online sources, managing devices, scheduling alarms, or delivering requested information.

*Response Generation:* Following task execution, the system generates a response to provide feedback to the user. This response is crafted based on the outcome of the task and can take various forms, including spoken output using text-to-speech synthesis, visual feedback presented on a user interface, or other communication methods tailored to the application's interface design. The response aims to inform the user of the action taken and ensure clarity and satisfaction with the system's performance.

### 2.8. Text-To-Speech

Text-to-speech (TTS) technology revolutionizes digital communication by enabling computers to transform written text into spoken words. With a wide array of applications, TTS enhances accessibility for visually impaired individuals, providing them with auditory access to digital content. Moreover, it facilitates the development of interactive voice-based systems, enriching user experiences across various platforms.
TTS operates by analyzing textual input and synthesizing it into natural-sounding audio output, mimicking human speech patterns and intonations.

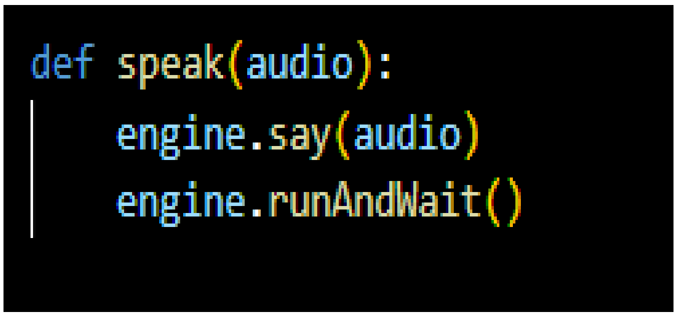


```
engine = pyttsx3.init("sapi5")
voices = engine.getProperty("voices")
engine.setProperty("voice", voices[0].id)
rate = engine.setProperty("rate",170)
```

*Pyttsx3:* takes text as input and converts it to speech using the initialized text-to-speech engine.

*pyttsx3.init("sapi5"):* Initializes the text-to-speech engine using Microsoft's Speech API (sapi5).

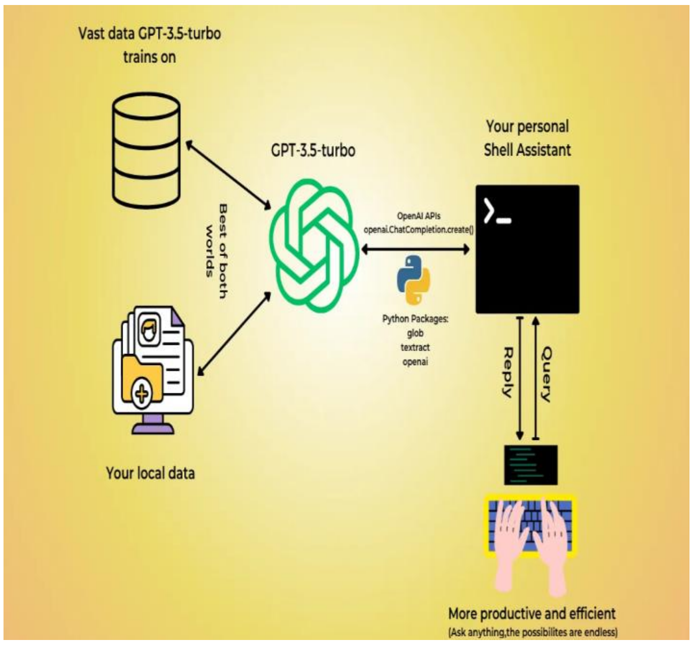
*engine.setProperty("rate", 170):* Sets the speaking rate of the text-to-speech engine to 170 words per minute.



```
def speak(audio):
    engine.say(audio)
    engine.runAndWait()
```

we provide the text we want to convert to speech using the say () method, and finally, we run the text-to-speech conversion using runAndWait()

## Chat GPT Integration

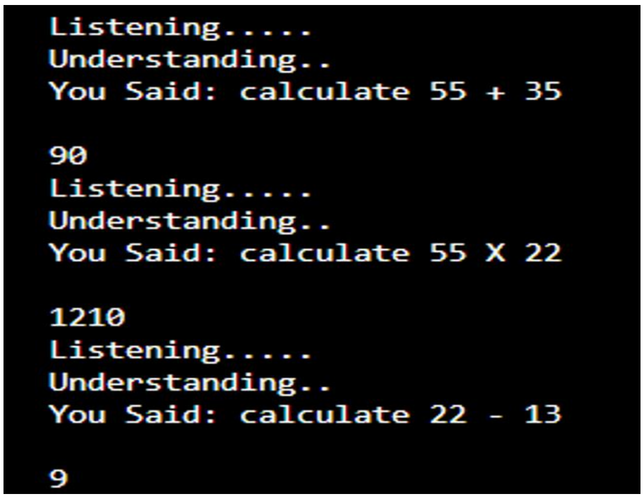

Vast data GPT-3.5-turbo trains on
GPT-3.5-turbo
Your personal Shell Assistant
Best of both worlds
OpenAI APIs
openai.ChatCompletion.create()
Python Packages:
glob
textract
openai
Reply
Query
Your local data
More productive and efficient
(Ask anything,the possibilites are endless)

The voice assistant seamlessly integrates OpenAI's GPT-3.5 Turbo model, a powerhouse in natural language processing. GPT-3.5 Turbo is adept at understanding user queries and generating contextually relevant responses, providing a conversational interaction experience akin to human conversation. Before querying GPT-3.5 Turbo, the system constructs prompt that furnish the necessary context, aiding the model in generating more accurate responses. These prompts, prefixed with "Chando:" to denote user input and followed by "\n Jarvis:" to signify the assistant's response, facilitate effective communication between the user and the assistant. This approach enhances the assistant's ability to comprehend nuanced queries and deliver appropriate responses, contributing to a more seamless user experience.
Once prompts are constructed, GPT-3.5 Turbo processes them and generates responses based on its extensive training on vast text data. The generated responses are tailored to address the user's query or command in a conversational tone, emulating human-like interaction.

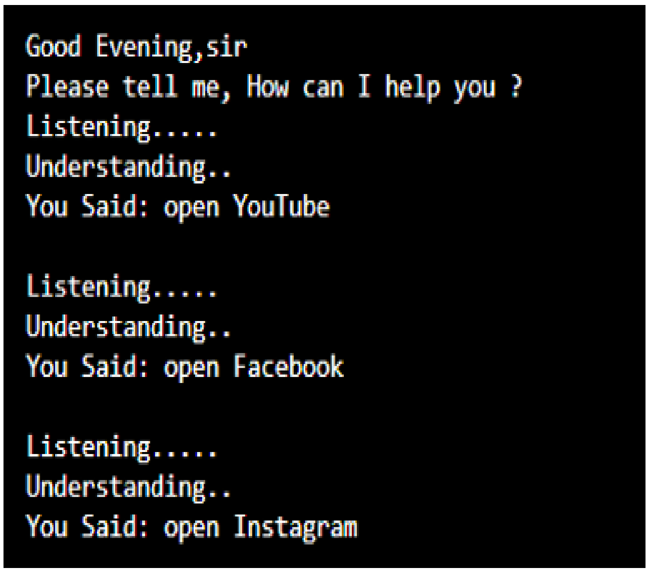
Subsequently, these responses are converted into speech format using the pyttsx3 library, enabling the assistant to deliver them audibly to the user. This text-to-speech conversion ensures that users can interact with the assistant effortlessly, without the need for textual reading.

Overall, this integrated approach, encompassing advanced natural language processing, context-aware prompt construction, and seamless speech synthesis, culminates in an engaging and user-friendly interaction experience with the voice assistant.

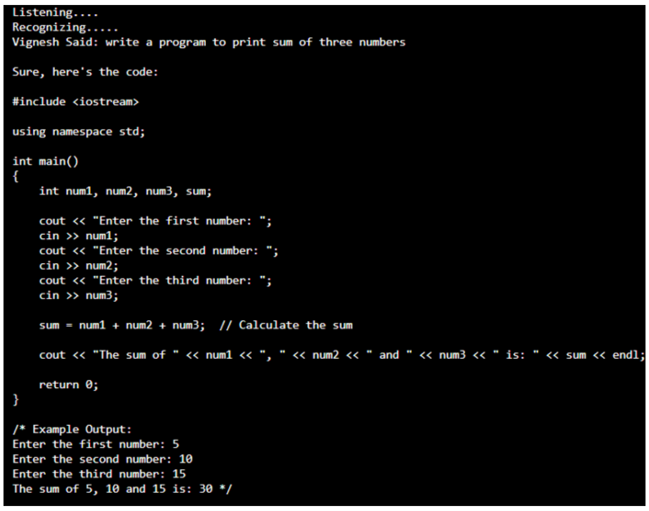
## 3. RESULTS



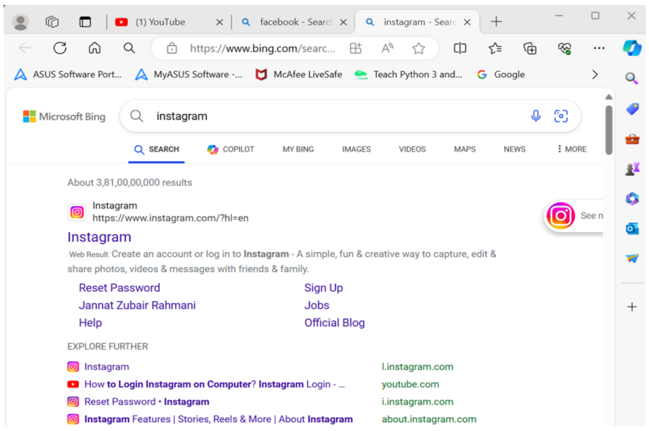
```
Listening.....
Understanding..
You Said: calculate 55 + 35

90
Listening.....
Understanding..
You Said: calculate 55 X 22

1210
Listening.....
Understanding..
You Said: calculate 22 - 13

9
```



```
Good Evening,sir
Please tell me, How can I help you ?
Listening.....
Understanding..
You Said: open YouTube

Listening.....
Understanding..
You Said: open Facebook

Listening.....
Understanding..
You Said: open Instagram
```



```
Listening....
Recognizing.....
Vignesh Said: write a program to print sum of three numbers

Sure, here's the code:

#include <iostream>

using namespace std;

int main()
{
    int num1, num2, num3, sum;

    cout << "Enter the first number: ";
    cin >> num1;
    cout << "Enter the second number: ";
    cin >> num2;
    cout << "Enter the third number: ";
    cin >> num3;

    sum = num1 + num2 + num3;  // Calculate the sum

    cout << "The sum of " << num1 << ", " << num2 << " and " << num3 << " is: " << sum << endl;

    return 0;
}

/* Example Output:
Enter the first number: 5
Enter the second number: 10
Enter the third number: 15
The sum of 5, 10 and 15 is: 30 */
```



About 3,81,00,00,000 results

Instagram
https://www.instagram.com/?hl=en

Instagram

Web Result Create an account or log in to **Instagram** - A simple, fun & creative way to capture, edit & share photos, videos & messages with friends & family.

Reset Password
Sign Up
Jannat Zubair Rahmani
Jobs
Help
Official Blog

EXPLORE FURTHER

| | |
|---|---|
| Instagram | l.instagram.com |
| How **to Login Instagram on Computer**? **Instagram** Login - ... | youtube.com |
| Reset Password • **Instagram** | i.instagram.com |
| **Instagram** Features \| Stories, Reels & More \| About **Instagram** | about.instagram.com |

## 4. CONCLUSION

Our desktop assistant project represents a significant milestone in human-computer interaction, leveraging cutting-edge technologies to create an intuitive and efficient interface. Through robust voice command recognition, natural language understanding, and seamless integration with external services, our assistant streamlines user interactions, boosting productivity. A primary achievement lies in its natural interaction capabilities, powered by advanced natural language processing. This enables meaningful conversations, enhancing user experience with a sense of adaptability.

Moreover, our assistant extends its functionality through integration with external APIs, providing real-time, personalized information such as weather forecasts, news updates, and multimedia playback.

This versatility makes it an indispensable tool for daily use.
Our desktop assistant empowers users to accomplish tasks more efficiently, reducing cognitive load and promoting digital inclusivity through its intuitive interface and hands-free operation. Automating routine tasks and streamlining workflows, significantly enhances productivity across various domains.
Furthermore, it highlights the transformative potential of voice-controlled interfaces, paving the way for future innovation in human-computer interaction. As technology evolves, there's immense potential for further refinement in voice recognition, natural language understanding, and contextual computing.
In conclusion, our desktop assistant project not only enhances user productivity but also sets a precedent for ethical and responsible AI development, marking a significant step forward in the digital age.

## 5. FUTURE SCOPE

As AI continues to advance and voice technology becomes increasingly pervasive, voice-controlled digital assistants are poised to seamlessly integrate into daily devices, fundamentally altering how we interact with technology. These assistants will evolve to offer interactions that closely mimic human dialogues, enabling more complex and fluid task flows. This shift reflects a growing reliance on voice technology, with users embracing it for a multitude of tasks and interactions.
Looking ahead, deeper integration of voice technology into devices will make voice-based searches and commands even more effortless. For example, the introduction of products like Amazon's wall clock with built-in Alexa functionality underscores the expanding capabilities of voice integration beyond traditional personal assistants. These innovations foreshadow a future where vocal commands will serve as the primary mode of interaction with our devices, seamlessly woven into the fabric of daily life.
Moreover, as voice technology becomes more deeply ingrained in devices, interactions will become increasingly natural and intuitive. While current interactions may feel fragmented, ongoing advancements in digital processing and user familiarity with voice assistants will lead to smoother, more seamless conversations. Users will no longer encounter interruptions or delays, fostering a sense of comfort and familiarity in their interactions with technology. This evolution promises to redefine the user experience, creating a world where devices respond effortlessly to our spoken commands, enhancing convenience and efficiency in our daily lives.

## 6. REFERENCES

**Journal reference style:**

- https://towardsdatascience.com/how-to-build-your-own-ai-personal-assistant-using-pythonf57247b4494b
- https://www.analyticsvidhya.com/blog/2020/11/build-your-own-desktop-voice-assistant-inpython/