A PROJECT REPORT
ON
Voice Assistant using Python

Submitted in the partial fulfilment of award of

BACHELOR OF TECHNOLOGY

Degree In

Computer Science and Engineering

Submitted To:

Mr. SUDIPTA SAHANA

Submitted By:

ANUPAM DUTTA
BIROTTAM BISWAS
RITESH SAHA
SHUBHRANIL MAZUMDER

Voice Assistant using Python

B.Tech 4th Semester Project Report

Submitted towards the partial fulfillment of the requirements for the

B.Tech in Computer Science & Engineering

By

ANUPAM DUTTA	(123200803202)
BIROTTAM BISWAS	(123200803205)
RITESH SAHA	(123200803207)
SHUBHRANIL MAZUMDER	(123200803209)

Under the supervision

Of

SUDIPTA SAHANA

Department of Computer Science & Engineering

JIS College of Engineering

Affiliated to

All India Council for Technical

University Grants Commission

Maulana Abul Kalam Azad University of Technology

NAAC, NBA, NIRF

BLOCK A, PHASE III, KALYANI, NADIA

2020 - 2021

DECLARATION

We ANUPAM DUTTA, BIROTTAM BISWAS, RITESH SAHA, SHUBHRANIL MAZUMDAR, bearing College Roll No. 123200803202, 123200803205, 123200803207, 123200803209, Declare that the project work is an original work performed by us in the Department of Computer Science & Engineering, JIS College of Engineering, Kalyani. To complete the work, we have taken some references and are cited in the report.

Place: Signature of student:

Date:

CERTIFICATE OF APPROVAL

This is to certify that Anupam Dutta, College Roll No. 123200803202 has submitted the Diploma project entitled **Voice Assistant using Python** in partial fulfillment of the requirement for the 4th Semester B.Tech in Computer Science & Engineering of JIS COLLEGE OF ENGINEERING in the session 2020 – 2021. It is hereby approved and certified as creditable study of technological subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the B.Tech in Computer Science & Engineering for which it has been submitted.

It is understood that by the approval the undersigned does not necessarily endorse or approve any statement made. Opinion expressed or conclusion drawn therein, but approve the report only for the purpose for which has been submitted.

Name of Supervisor
Designation

ACKNOWLEDGEMENT

I wish to express my profound gratitude and indebtedness to Prof. SUDIPTA SAHANA, Department of Computer Science & Engineering, JIS College of Engineering, Kalyani for introducing the present topic and for their inspiring guidance, constructive criticism and valuable suggestion throughout the project work.

Last but not least, my sincere thanks to all our friends who have patiently extended all sorts of help for accomplishing this undertaking.

ANUPAM DUTTA (123200803202)

BIROTTAM BISWAS (123200803205)

RITESH SAHA (123200803207)

SHUBHRANIL MAZUMDER (123200803209)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JIS COLLEGE OF ENGINEERING

BLOCK A, PHASE III, KALYANI, NADIA

TABLE OF CONTENTS

INTRODUCTION

A virtual assistant is an independent contractor who provides administrative services to clients while operating outside of the client's office. A virtual assistant typically operates from a home office but can access the necessary planning documents, such as shared calendars, remotely.

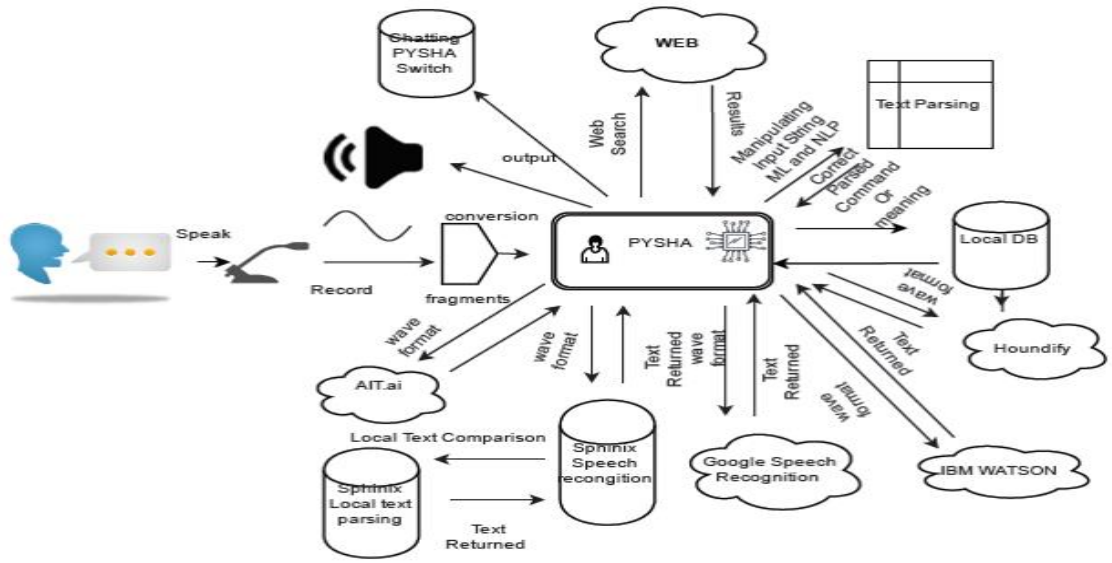
People employed as virtual assistants often have several years of experience as an administrative assistant or office manager. New opportunities are opening up for virtual assistants who are skilled in social media, content management, blog post writing, graphic design, and Internet marketing. As working from home has become more accepted for both workers and employers, the demand for skilled virtual assistants is expected to grow.

In today's era almost all tasks are digitalized. We have Smartphone in hands and it is nothing less than having world at your fingertips. These days we aren't even using fingers. We just speak of the task and it is done. There exist systems where we can say Text Friend, "I will Call you Letter." And the text is sent or any one may make call by saying "Call To Mr. Roy" That is the task of a Virtual Assistant. It also supports specialized task such as booking a flight, or finding cheapest book online from various e-commerce sites and then providing an interface to book an order are helping automate search, discovery and online order operations.

An intelligent virtual assistant (IVA) or intelligent personal assistant (IPA) is a software agent that can perform tasks or services for an individual based on commands or questions. The term "chatbot" is sometimes used to refer to virtual assistants generally or specifically accessed by online chat. In some cases, online chat programs are exclusively for entertainment purposes. Some virtual assistants are able to interpret human speech and respond via synthesized voices. Users can ask their assistants questions, control home automation devices and media playback via voice, and manage other basic tasks such as email, to-do lists, and calendars with verbal (spoken?) commands. A similar concept, however with differences, lays under the dialogue systems.

The project aims to develop a personal-assistant for Linux-based / Windows systems using Python, Jupitar Notebook, PyCharm. Voice Assistant using Python draws its inspiration from virtual assistants like Cortana for Windows, and Siri for iOS. It has been designed to provide a user-friendly interface for carrying out a variety of tasks by employing certain well-defined commands. Users can interact with the assistant either through voice commands or using sound or Voice input input. As a personal assistant, this system assists the end-user with day-to-day activities like general human conversation, searching queries in google, Bing or yahoo, searching for videos, retrieving images, live weather conditions, word meanings, searching for medicine details, health recommendations based on symptoms and reminding the user about the scheduled events and tasks. The user statements/commands are analyzed with the help of machine learning to give an optimal solution.

Basic Diagram



Technology Used

1. Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.
Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

2. Jupyter Notebook

JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.

3. PyCharm

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as data science with Anaconda.

LITERATURE SURVEY

Google Assistant is an artificial intelligence–powered virtual assistant developed by Google that is primarily available on mobile and smart home devices. Unlike the company's previous virtual assistant, Google Now, the Google Assistant can engage in two-way conversations.

Assistant initially debuted in May 2016 as part of Google's messaging app Allo, and its voice-activated speaker Google Home. After a period of exclusivity on the Pixel and Pixel XL smartphones, it began to be deployed on other Android devices in February 2017, including third-party smartphones and Android Wear (now Wear OS), and was released as a standalone app on the iOS operating system in May 2017. Alongside the announcement of a software development kit in April 2017, the Assistant has been further extended to support a large variety of devices, including cars and third-party smart home appliances. The functionality of the Assistant can also be enhanced by third-party developers.

Users primarily interact with the Google Assistant through natural voice, though keyboard input is also supported. In the same nature and manner as Google Now, the Assistant is able to search the Internet, schedule events and alarms, adjust hardware settings on the user's device, and show information from the user's Google account. Google has also announced that the Assistant will be able to identify objects and gather visual information through the device's camera, and support purchasing products and sending money.
Amazon Alexa, also known simply as Alexa, is a virtual assistant AI technology developed by Amazon, first used in the Amazon Echo smart speaker and the Echo Dot, Echo Studio and Amazon Tap speakers developed by Amazon Lab126. It is capable of voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, playing audiobooks, and providing weather, traffic, sports, and other real-time information, such as news. Alexa can also control several smart devices using itself as a home automation system. Users are able to extend the Alexa capabilities by installing "skills" (additional functionality developed by third-party vendors, in other settings more commonly called apps) such as weather programs and audio features.

Most devices with Alexa allow users to activate the device using a wake-word (such as Alexa or Amazon); other devices (such as the Amazon mobile app on iOS or Android and Amazon Dash Wand) require the user to click a button to activate Alexa's listening mode, although, some phones also allow a user to say a command, such as "Alexa" or "Alexa wake". Currently, interaction and communication with Alexa are available only in English, German, French, Italian, Spanish, Portuguese, Japanese, and Hindi. In Canada, Alexa is available in English and French (with the Quebec accent)

Used Modules

Speech Recognition: - Since we're building an application of voice assistant, one of the most important things in this is that your assistant recognizes your voice (means what you want to say/ ask). To install this module, type the below command in the terminal.

pip install SpeechRecognition

Web browser: - To perform Web Search. **This module comes built-in with Python.**

Ecapture: - To capture images from your Camer.To install this module type the below command in the terminal.

pip install ecapture

Pyjokes: - Pyjokes is used for collection Python Jokes over the Internet. To install this module type the below command in the terminal.

pip install pyjokes

Datetime: - Date and Time is used to showing Date and Time. This module **comes built-int with Python.**

Twilio: - Twilio is used for making call and messages. To install this module type the below command in the terminal.

pip install twilio

Requests: - Requests is used for making GET and POST requests. To install module type the below command in the terminal.

pip install requests

Beautiful Soup: - Beautiful Soup is a library that makes it easy to scrape information from web pages. To install this module type the below command in the terminal.

pip install beautifulsoup4

Building a Recommender System

We will make a **wishme()** function that will make our Voice Assistance wish or greet the user according to the time of computer or pc. To provide current or live time to A.I., we need to import a module called date time. Import this module to your program by:

import datetime

Now, let's start defining the **wishme()** function:

```
def wishme():
        hour = int(datetime.datetime.now().hour)
```

Here, we have stored the current hour or time integer value into a variable named hour. Now, we will use this hour value inside an if-else loop.

Defining Take command Function:

The next most important thing for our A.I. assistant is that it should take command with the help of the microphone of the user's system. So, now we will make a **takeCommand()** function. With the help of the **takeCommand()** function, our A.I. assistant will return a string output by taking microphone input from the user.

Before defining the **takeCommand()** function, we need to install a module called **speechRecognition**. Install this module by:

pip install speechRecognition

After successfully installing this module, import this module into the program by writing an import statement.

import speechRecognition as sr

Defining Task 5: To know the current time

```
elif 'the time' in query:
        strTime = datetime.datetime.now().strftime("%H:%M:%S")
        speak(f"Sir, the time is {strTime}")
```

In the above, code we are using the datetime() function and storing the current or live system time into a variable called strTime. After storing the time in strTime, we are passing this variable as an argument in speak function. Now, the time string will be converted into speech.

Defining Task 6: To open the VS Code Program
elif 'open code' in query:

codePath = "C:\\Users\\Haris\\AppData\\Local\\Programs\\Microsoft VS Code\\Code.exe"
os.startfile(codePath)
To open the VS Code or any other application, we need the code path of the application.

Starting VS Cod**e:**
I am going to use the VS Code IDE in this video. Feel free to use any other IDE you are comfortable d with. Start a new project and make a file called jarvis.py.

Defining Speak Function:
The first and foremost thing for an A.I. assistant is that it should be able to speak. To make our J.A.R.V.I.S. talk, we will make a function called speak(). This function will take audio as an argument, and then it will pronounce it.

```
def speak(audio):
    pass    #For now, we will write the conditions later.
```
Now, the next thing we need is audio. We must supply audio so that we can pronounce it using the speak() function we made. We are going to install a module called pyttsx3.

What is pyttsx3?
A python library that will help us to convert text to speech. In short, it is a text-to-speech library.
It works offline, and it is compatible with Python 2 as well as Python 3.
Installation:

pip install pyttsx3
In case you receive such errors:

No module named win32com.client
No module named win32
No module named win32api
Then, install pypiwin32 by typing the below command in the terminal :

pip install pypiwin32.
After successfully installing pyttsx3, import this module into your program.

Usage:

```
import pyttsx3
engine = pyttsx3.init('sapi5')
```

voices= engine.getProperty('voices') #getting details of current voice

engine.setProperty('voice', voice[0].id)
What is sapi5?
Microsoft developed speech API.
Helps in synthesis and recognition of voice.
What Is VoiceId?
Voice id helps us to select different voices.
voice[0].id = Male voice
voice[1].id = Female voice

Writing Our speak() Function :
We made a function called **speak()** at the starting of this tutorial. Now, we will write our **speak()** function to convert our text to speech.

def speak(audio):

engine.say(audio)

engine.runAndWait() #Without this command, speech will not be audible to us.
Creating Our main() function:
We will create a **main()** function, and inside this main**()** Function, we will call our speak function.

Code:

if __name__=="__main__" :

speak("JIS College Of Engineering")

Whatever you will write inside this speak() function will be converted into speech. Congratulations! With this, our Voice Assistance . has its own voice, and it is ready to speak.

Let's start coding **the takeCommand()** function :

```
def takeCommand():
    #It takes microphone input from the user and returns string output

    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)
```

Successfully created our **takeCommand()** function. Now we are going to add a try and except block to our program to handle errors effectively.

```
  try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in') #Using google for
voice recognition.
        print(f"User said: {query}\n")  #User query will be printed.

    except Exception as e:
        # print(e)
        print("Say that again please...")   #Say that again will be printed in case
of improper voice
        return "None" #None string will be returned
    return query
```

Coding logic of Voice Assistance:-

Now, we will develop logic for different commands such as Wikipedia searches, playing music, etc.

Defining Task 1: To search something on Wikipedia
To do Wikipedia searches, we need to install and import the Wikipedia module into our program. Type the below command to install the Wikipedia module :

```
pip install wikipedia
```

After successfully installing the Wikipedia module, import it into the program by writing an import statement.

```
if __name__ == "__main__":
    wishMe()
    while True:
    # if 1:
        query = takeCommand().lower() #Converting user query into lower case

        # Logic for executing tasks based on query
        if 'wikipedia' in query:  #if wikipedia found in the query then this block will
be executed
            speak('Searching Wikipedia...')
            query = query.replace("wikipedia", "")
            results = wikipedia.summary(query, sentences=2)
            speak("According to Wikipedia")
            print(results)
            speak(results)
```

In the above code, we have used an if statement to check whether Wikipedia is in the user's search query or not. If Wikipedia is found in the user's search query, then two sentences from the summary of the Wikipedia page will be

converted to speech with the speak function's help.

Defining Task 2: To open YouTube site in a web-browser
To open any website, we need to import a module called webbrowser. It is an in-built module, and we do not need to install it with a pip statement; we can directly import it into our program by writing an import statement.

Code:

```
elif 'open youtube' in query:
    webbrowser.open("youtube.com")
```

Here, we are using an elif loop to check whether Youtube is in the user's query. Let' suppose the user gives a command as "J.A.R.V.I.S., open youtube." So, open youtube will be in the user's query, and the elif condition will be true.

Defining Task 3: To open Google site in a web-browser
elif 'open google' in query:

```
    webbrowser.open("google.com")
```

We are opening Google in a web-browser by applying the same logic that we used to open YouTube.

Defining Task 4: To play music
To play music, we need to import a module called os. Import this module directly with an import statement.

```
elif 'play music' in query:
    music_dir = 'D:\\Non Critical\\songs\\Favorite Songs2'
    songs = os.listdir(music_dir)
    print(songs)
    os.startfile(os.path.join(music_dir, songs[0]))
```

In the above code, we first opened our music directory and then listed all the songs present in the directory with the os module's help. With the help of os.startfile, you can play any song of your choice. I am playing the first song in the directory. However, you can also play a random song with the help of a random module. Every time you command to play music, A.I Assistance will play any random song from the song directory.

Steps to get the code path of the application:

Step 1: Open the file location.

Step 2: Right-click on the application and click on properties.

Step 3: Copy the target from the target section.

After copying the target of the application, save the target into a variable. Here, I am saving the target into a variable called codePath, and then we are

using the os module to open the application.

Defining Task 7: To send Email
To send an email, we need to import a module called smtplib.

What is smtplib?

Simple Mail Transfer Protocol (SMTP) is a protocol that allows us to send emails and route emails between mail servers. An instance method called sendmail is present in the SMTP module. This instance method allows us to send an email. It takes 3 parameters:
The sender: Email address of the sender.
The receiver: T Email of the receiver.
The message: A string message which needs to be sent to one or more than one recipient.

Defining Send email function :

We will create a sendEmail() function, which will help us send emails to one or more than one recipient.

```
def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login('youremail@gmail.com', 'your-password')
    server.sendmail('youremail@gmail.com', to, content)
    server.close()
```

In the above code, we are using the SMTP module, which we have already discussed above.

Note: Not forget to 'enable the less secure apps' feature in your Gmail account. Otherwise, the sendEmail function will not work properly.

Calling sendEmail() function inside the main() function:

```
 elif 'email to harry' in query:
            try:
                speak("What should I say?")
                content = takeCommand()
                to = "harryyourEmail@gmail.com"
                sendEmail(to, content)
                speak("Email has been sent!")
            except Exception as e:
                print(e)
                speak("Sorry my friend harry bhai. I am not able to send this email")
```

We are using the try and except block to handle any possible error while sending emails.

First of all, we have created a wishme() function that gives the greeting functionality according to our A.I system time.

After wishme() function, we have created a takeCommand() function, which helps our A.I to take command from the user. This function is also responsible for returning the user's query in a string format.

We developed the code logic for opening different websites like google, youtube, and stack overflow.
Developed code logic for opening VS Code or any other application.
At last, we added functionality to send emails.

Project Code:

```
import pyttsx3 #pip install pyttsx3
import speech_recognition as sr #pip install speechRecognition
import datetime
import wikipedia #pip install wikipedia
import webbrowser
import os
import smtplib

engine = pyttsx3.init('sapi5')
voices = engine.getProperty('voices')
# print(voices[1].id)
engine.setProperty('voice', voices[0].id)


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


def wishMe():
    hour = int(datetime.datetime.now().hour)
    if hour>=0 and hour<12:
        speak("Good Morning!")

    elif hour>=12 and hour<18:
        speak("Good Afternoon!")

    else:
        speak("Good Evening!")

    speak("I am Jarvis Sir. Please tell me how may I help you")
```

```
def takeCommand():
    #It takes microphone input from the user and returns string output

    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source)

    try:
        print("Recognizing...")
        query = r.recognize_google(audio, language='en-in')
        print(f"User said: {query}\n")

    except Exception as e:
        # print(e)
        print("Say that again please...")
        return "None"
    return query

def sendEmail(to, content):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.ehlo()
    server.starttls()
    server.login('youremail@gmail.com', 'your-password')
    server.sendmail('youremail@gmail.com', to, content)
    server.close()

if __name__ == "__main__":
    wishMe()
    while True:
    # if 1:
        query = takeCommand().lower()

        # Logic for executing tasks based on query
        if 'wikipedia' in query:
            speak('Searching Wikipedia...')
            query = query.replace("wikipedia", "")
            results = wikipedia.summary(query, sentences=2)
            speak("According to Wikipedia")
            print(results)
            speak(results)

        elif 'open youtube' in query:
            webbrowser.open("youtube.com")

        elif 'open google' in query:
            webbrowser.open("google.com")

        elif 'open stackoverflow' in query:
```

```
            webbrowser.open("stackoverflow.com")


        elif 'play music' in query:
            music_dir = 'D:\\Non Critical\\songs\\Favorite Songs2'
            songs = os.listdir(music_dir)
            print(songs)
            os.startfile(os.path.join(music_dir, songs[0]))

        elif 'the time' in query:
            strTime = datetime.datetime.now().strftime("%H:%M:%S")
            speak(f"Sir, the time is {strTime}")

        elif 'open code' in query:
            codePath = "C:\\Users\\Haris\\AppData\\Local\\Programs\\Microsoft
VS Code\\Code.exe"
            os.startfile(codePath)

        elif 'email to harry' in query:
            try:
                speak("What should I say?")
                content = takeCommand()
                to = "harryyourEmail@gmail.com"
                sendEmail(to, content)
                speak("Email has been sent!")
            except Exception as e:
                print(e)
                speak("Sorry my friend harry bhai. I am not able to send this email")
```

APPROACH

Many voice assistants are far from perfect. Some fail to carry out basic requests, others struggle to handle spontaneous conversation, and most seem detached from the friendly brands they're meant to represent.

The truth is, it's tough to transform a purely functional bot into a smooth and delightful voice experience that your brand can be proud of. There are just so many angles to consider to get it right—from choosing the right tone to composing the least frustrating error responses.

With over 14 years of experience in the voice tech space and with dozens of notable brands under their belt, including Mercedes-Benz, Kia, Honda, and Pandora, it's safe to say that the SoundHound team has cracked the formula for building effective voice assistants.
To help brands hit home with their own voice assistants, Anshu Chakraborty — Sound Hound's Director of Product Management— appeared on the Inside VOICE podcast to share six key practices for building better voice assistants.
Here are the main takeaways from her insightful episode.

1. Extend your brand through voice

According to Anshu, the first step to building a voice assistant is identifying the main use-cases you want to focus on. Then, narrow those down into the ones that truly help people accomplish things more easily with voice.

"For a lot of brands, especially big brands, you already have a brand experience your customers recognize. So it's really extending that same experience to another venue, through voice." —Heidi Culbertson, Founder & CEO of Marvee (source).
Anshu gives the example of a brand that helps people find hotels online. It's much faster to ask a voice assistant, "show me five-star hotels with a pool," than to click on a website, choose the filters and search. So in this scenario, the use-case for voice-enabled hotel searching would be at the top of the list.

Anshu also underlines the importance of formulating use-cases that consider the context in which they would occur. For instance, the Mercedes-Benz MBUX voice system takes into account that their users will be driving so their voice assistant's answers need to be brief and to the point. It must also avoid redirecting the user's attention to a screen to complete an action.

2. Give the assistant a personality

Once you've defined your main use-cases, you can begin to map the conversational flow for each one and define the right tone. Generally, you want the tone to fit the context (e.g. a calm and professional tone when handling banking information), or

counteract any negative emotions the user may be experiencing (e.g. a reassuring tone after an error response).

Although making your voice assistant sound as human and natural as possible is one of the hardest parts. To help, Anshu recommends reading the script aloud (or even blindfolded) to pick up on confusing or jarring dialogue. “A content strategist or dialogue writer is really essential to make this work,” she added.

As for what your voice assistant should *sound* like, Sonic brand composer Jeanna Isham once explained that it’s not as simple as picking the voice that the CEO likes best from an online library. You need to have a foundational understanding of what persona your brand wants to represent (e.g. the friendly helper, the formal academic), so you can choose the voice and tone that conveys your brand’s character to your users.

3. Prompt natural interactions

Interacting with a voice assistant shouldn’t feel like placing an order at a drive-through. Brands need to think of these interactions as organic conversations that recognize follow-ups and can maintain the pace of a real-life chat.

At SoundHound, the team covers their bases by brainstorming every possible way that a user could say the same request. For example: “What’s the weather tomorrow?” as opposed to, “Will it rain tomorrow?”

Granted, it’s impossible to capture every single variation that users could think of. But this is why continuous user testing is essential—before *and* after launching your voice assistant. Anshu strongly suggests watching how actual users ask for something to help you gradually compose the most natural-sounding conversations.

4. Handle errors gracefully

Anyone who has asked an off-hand question to their voice assistant and immediately confused them knows that this tech is far from perfect. What marks the difference between a good voice assistant and a *better* one, however, is how it handles those shortcomings and what it does next.
Take Google, for example. If your internet connection fails in the middle of a search they don’t just drop you onto a blank page with a 404 error. Instead, they entertain you with a little dinosaur game until Google can get back online to retrieve your search results.

So, when designing error responses for your voice assistant, always give the user options rather than just throwing them an error message. For instance, if the assistant didn’t capture what the user was saying, offer them the chance to repeat themselves or select an option from a menu. This way, as Anshu so nicely put it: “At least if we don't have the answer, we can get you closer to it.”

5. Personalize the user experience

“Personalization is all about data,” Anshu said with absolute certainty. She also emphasized the importance of understanding the user’s needs before you can even begin to devise ways to personalize your voice assistant.

You can do this in small ways, like “remembering” the user’s location when they ask where the nearest bar is. Or what their most played songs are when they ask for music. It can be as simple as adding quick shortcuts for things that they use the most.

“Ultimately, users just want to make their life easier. So make something that makes using your voice assistant worth it. It doesn't need to be complicated or in-depth; it could be as simple as giving a short, relevant response that gives the user exactly what they need.” Anshu concluded.

RESULTS

Provides 24/7 Customer Support

Consumers request for round the clock support. Sometimes there are instances when they require assistance at odd hours, and when help isn't available, it becomes a frustrating experience. To avoid such situations, voice assistants come in handy. A digital talking assistant does not require any off days or sick leaves that interrupt customer care and experiences.

Take, for instance, if you run a hotel using voice assistant technology, if a guest gets too cold or warm at night, rather than call the front desk or fiddle with instruments, the guest can make use of the in-room smart speaker. Therefore, you eliminate the need for night support staff because you are covered with personal assistant technology, and your 24/7 customer support is guaranteed.

Eradicates Language Barriers

While travelling abroad or even while interacting with content online, most people have to deal with language barriers. So what's the solution? Including personal assistant technology integrated with automatic translation to help ease the language barrier.

For example, Google's assistant is compatible with 27 different languages and working on adding more languages. Thus, when consumers are able to communicate with you in their native language, it can unequivocally lead to better customer experience and more business for you.

Helps Streamline Operations

Another excellent business benefit of personal voice assistants is that it streamlines operations that come with integrating digital assistants into your business. Even with emerging innovations and deep learning, these talking assistants never stop working. They are continually accessing reports, analyzing data, and ensuring critical systems are updated.

You can make use of Alexa Skills and Google Actions to facilitate specific actions for your customers. Artificial intelligence assistant technology helps your business to streamline day-to-day operations that are always being supervised. Things such as remembering important dates, deadlines, booking appointments, scheduling, etc. can be all triggered using specific voice commands.

Enables Smart Offices

Voice assistants also allow the creation of smart and connected offices. If a voice-activated personal assistant knows that a particular part of the office space will not be in use, it can connect with smart office solutions to turn off the lights and other utilities until the area is needed. You can do this by simply setting up a smart thermostat to your voice assistant and tell it when you're leaving the office so that your heating and lights get turned off. Also, office and business resources can be ordered using simple voice commands and likewise set to give alerts when supplies are running low.

Aids Hand-free Operation

Voice talking gives consumers hands-free access to many functions because you only need the voice to activate them. So it makes it easier and faster to do certain things. The research from PwC shows that consumers often make use of personal voice assistant while doing other tasks such as cooking, watching TV, driving, etc.

The research also shows that several demographic groups find it easy to use voice assistance. Hence the high adoption rate of voice technology poses great potential for companies that can utilise the technology to their advantage.

System improvements

Our project has several fetchers' but it does not have AI Neural networking and data mining so it became a static application by integrating many other concepts like AI neural networking and data mining we can get new cut of the edge system

Conclusion:

This examination speaks to an initial phase in investigating The potential job of remote helpers in programming Improvement Ventures finished by Virtual groups. This Project will be help for visually impaired and physically Challenge people. Instead, we will see a fragmented Marketplace emerges. It will be a market where you are Might into using default AI providers depending on the Hardware Purchase. This will lead to consumer friction and third-party solutions to remove incumbent solutions. This is how simple it is to build your own voice assistant. You can add many more features such as play your favorite songs, give weather details, open email application, compose emails, Restart your system, etc. You can integrate this application into your phone or tablet as well. Have fun exploring and developing
Your own Alexa/Siri/Cortana. Throughout the history of computing, user Interfaces have become progressively natural to use. The screen and Keyboards were one step in this direction. The mouse and graphical user Interfaces were another. Touch screens are the most recent development. The next step will most likely consist of a mix of augmented reality, Gestures and Voice Commands. After all, it is often easier to ask a Question or have a conversation than it is to type something or enter Multiple Details in an online form. The more a person interacts with voice-activated devices, The more trends and patterns the system identifies based on the information it receives. Then, this data can be Utilized to determine user preferences and tastes, which Is a long-term selling point for making a home smarter?
Google and Amazon are looking to integrate voice-enabled Artificial intelligence capable of analyzing and responding To Human Emotion.

References:

1. Abhay Dekate, Chaitanya Kulkarni, Rohan Killedar, “Study of Voice Controlled Personal Assistant Device”, International Journal of Computer Trends And Technology (IJCTT) – Volume 42 Number 1 – December 2016.

2. Deny Nancy, Sumithra Praveen, AnushriaSai, M.Ganga, R.S. Abisree, “Voice Assistant Application For a college Website”, international Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-7, Issue-6S5, April 2019.

3. Deepak Shende, Ria Umahiya, Monika Raghorte, Aishwarya Bhisikar, Anup Bhange, “AI Based Voice Assistant Using Python”, Journal of Emerging Technologies and Innovative Research (JETIR), February 2019, Volume 6, Issue 2,

4. Dr.Kshama V. Kulhalli, Dr.KotrappaSirbi, Mr.Abhijit J. Patankar, “Personal Assistant with Voice Recognition Intelligence”, International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 10, Number 1 (2017).

5. Isha S. Dubey, Jyotsna S. Verma, Ms.Arundhati Mehendale, “An Assistive System for Visually impaired using Raspberry Pi”, International Journal of Engineering Research & Technology (IJERT), Volume 8 Issue 05, May-2019.

6. Kishore Kumar R, Ms. J. Jayalakshmi, KarthikPrasanna, “A Python based Virtual Assistant using Raspberry Pi for Home Automation”, International Journal of Electronics

7. Agatha repository and Code Demo: https://github.com/Dipeshpal/Jarvis-Assisant

8. GitHub Pip repository to contribute: https://github.com/Dipeshpal/Jarvis_AI

9. JarvisAI Library: https://pypi.org/project/JarvisAI/