CHAPTER 4

PROJECT DETAILS

4.1 What is a Voice Assistant?

A voice assistant or intelligent personal assistant is a software agent that can perform tasks or services for an individual based on verbal commands i.e. by interpreting human speech and respond via synthesized voices. Users can ask their assistants' questions, open or close any shopping sites or any social media sites, media playback via voice, and manage other basic tasks such as email, open or close any application etc. with verbal commands.

Fig 4.1 Various scope of voice assistant

4.2 History of Voice Assistant

The first personal digital assistant (PDA) was released in 1984 by Pison, the Organizer. Early PDAs were devices having full keyboard and touch screen. The concept of virtual assistant was developed by Joseph Wizenbaum of MIT in the late 60s. In recent times,

Voice assistants got the major platform after Apple integrated the most astonishing Virtual Assistant — Siri which is officially a part of Apple Inc. But the timeline of greatest evolution began with the year 1962 event at the Seattle World Fair where IBM displayed a unique apparatus called Shoebox. It was the actual size of a shoebox and could perform scientific functions and can perceive 16 words and also speak them in the human recognizable voice with 0 to 9 numerical digits.

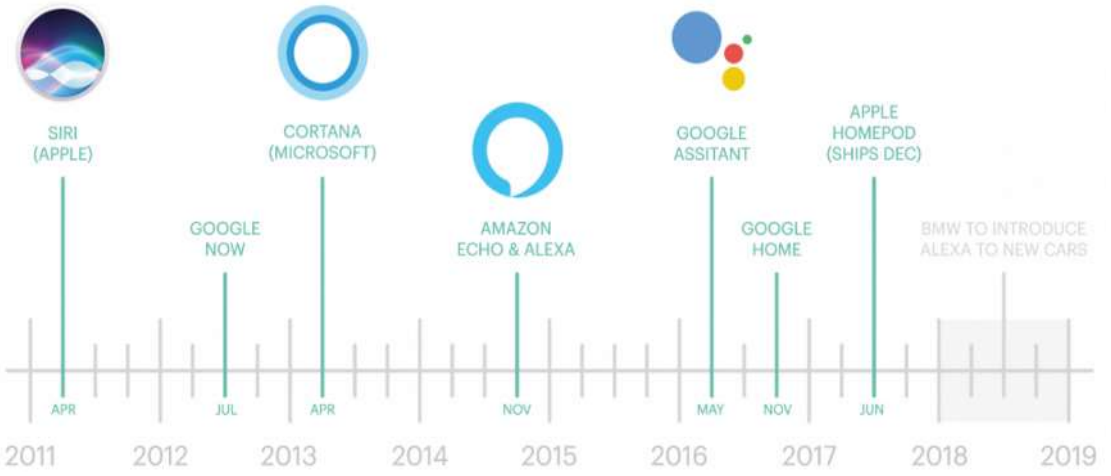

Fig 4.2 A modern history of voice assistant

4.3 Feature of the project

The main feature of this project is Voice Recognition Engine which has an ability to work without internet connection i.e. Offline Voice Recognition. But anything which needs to be searched online, will surely require active internet connection.

The technology on which it is based upon is Python and Machine Learning. It performs its various tasks with the help of different python modules being imported in it for its functionality. It also uses the sub-modules of the main python modules. It performs its voice speaking ability with the help of inbuilt Microsoft voice package **SAPI5**. The **Speech Application Programming Interface** or **SAPI** is an API developed by Microsoft to allow the use of speech recognition and speech synthesis within windows application. To date, a number of versions of the API have been released, which have shipped either as part of a speech SDK or as part of Windows OS itself. Application that use SAPI includes Microsoft Office, Microsoft Agent and Microsoft Speech Server. It can operate in both male and female voice which can be accordingly set by

their voice id. Voice id [0] is used for a male voice is which is called Microsoft Davis. Voice id [1] is used for a female voice which is called Microsoft Zira.

It has a function, named as takeCommand (), which takes microphone input from the user and convert it into string output. Subsequently, the output either is printed as a string on the output terminal or is converted into voice output with the help of pyttsx3 module and the assistant produces the speech output.

4.4 Python modules imported in the project

- **pyttsx3**:- An offline python text to speech library (TTS) which works for both python 2 and python 3. This library is very useful if we don't want any delay in the speech produced and don't want to depend only on the internet for TTS conversion. It includes TTS engine like:-
    i. Sapi5
    ii. nss
    iii. Espeak

    How to install?

    It can be installed with the command- pip install pyttsx3

    Usage:

    First we need to import the library and then initialise it using init() function. This function may take 2 arguments.

    init(driverName string, debug bool)

    i. drivername: [Name of available driver] ***sapi5*** on Windows | ***nsss*** on MacOS
    ii. debug: to enable or disable debug output

    After initialisation, we will make the program speak the text using say() function. This method may also take 2 arguments.

    say(text unicode, name string)

    i. **text :** Any text you wish to hear.
    ii. **name :** To set a name for this speech. (optional)

    Finally, to run the speech we use runAndWait(). All the say() texts won't be said unless the interpreter encounters runAndWait().

Why pyttsx?

It works offline, unlike other text-to-speech libraries. Rather than saving the text as audio file, *pyttsx* actually speaks it there. This makes it more reliable to use for voice-based projects.

Functions under pyttsx3:

pyttsx3.init():- It is used get a reference to a pyttsx3.Engine instance. During construction, the engine initializes a pyttsx3.driver.DriverProxy object responsible for loading a speech engine driver implementation from the pyttsx3.drivers module. After construction, an application uses the engine object to register and unregister event callbacks; produce and stop speech; get and set speech engine properties; and start and stop event loops. If the requested driver is already in use by another engine instance, that engine is returned. Otherwise, a new engine is created.

engine.getProperty():- Gets the current value of an engine property.

The following property names are valid for all drivers.

rate- Integer speech rate in words per minute. Defaults to 200 word per minute.

voice- String identifier of the active voice.

voices- List of pyttsx.voice.Voice descriptor objects.

volume- Floating point volume in the range of 0.0 to 1.0 inclusive. Defaults to 1.0.

engine.setProperty():- None

Queues a command to set an engine property. The new property value affects all utterances queued after this command.

Parameters:

name- name of the property to change

value- value to set

The following property names are valid for all drivers.

rate- Integer speech rate in words per minute

voice- String identifier of the active voice

volume- Floating point volume in the range of 0.0 to 1.0 inclusive.

say():- None

Queues a command to speak an utterance. The speech is output according to the properties set before this command in the queue.

Parameters:

text- text to speak

name- name to associate with the utterance. Included in notifications about the utterance.

runAndWait()- None

Blocks while processing all currently queued commands. Invokes callbacks for engine notifications appropriately. Returns when all commands queued before this call are emptied from the queue.

- **speech_recognition:-** Library for performing speech recognition, with support for several engines and APIs, online and offline.

  How to install?

  This can be installed with the command:- pip install SpeechRecognition
- **datetime:-** The datetime module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation. For related functionality, see also the time and calendar modules.
- **wikipedia:-** Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia.

  How to install?

  It can be installed with the command- pip install wikipedia

- **os:-** The OS module in python provides functions for interacting with the operating system. OS, comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality. The *os* and *os.path* modules include many functions to interact with the file system.
  In this project, I have used os.startfile() module to start any application being installed in the system by providing its path.

  os.startfile()- The os.startfile() method allows us to start a file with its associated program. In other words, we can open a file with its associated program, just like when we double click a pdf and it opens in Adobe Reader if it has a path like >>> os.startfile(r'C:\User's\mike\Documents\labels.pdf'). In this example, I passed a fully qualified path to os.startfile() that tells it to open a file called labels.pdf.

- **smtplib:-** Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers. Python provides **smtplib** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

  Here is a simple syntax to create one SMTP object, which can be used to send an e-mail –

  ```
  import smtplib

  smtpObj = smtplib.SMTP([host [, port [, local_hostname]]])
  ```

  Functions under smtplib:-

  server.ehlo:- Extended HELO (EHLO) is an Extended Simple Mail Transfer Protocol (ESMTP) command sent by an email server to identify itself when connecting to another email server to start the process of sending an email. It is followed with the sending email server's domain name. The EHLO command tells the receiving server it supports extensions compatible with ESMTP.

  server.starttls([keyfile, certfile]):- Put the SMTP connection in TLS (Transport Layer Security) mode. All SMTP commands that follow will be encrypted. You should then call ehlo() again.

If *keyfile* and *certfile* are provided, these are passed to the socket module's ssl() function.

server.login(username, password):- Log in on an SMTP server that requires authentication. The arguments are the username and the password to authenticate with. If there has been no previous EHLO or HELO command this session, this method tries ESMTP EHLO first. This method will return normally if the authentication was successful.

server.sendmail(sender's address, recipient address, content):-

It takes three arguments:-

i. Sender's address- Mail address of the sender is passed here.

ii. Recipient address- Mail address of the recipient is passed here.

iii. Content- Content of the mail is passed here. In this project, we are passing the content via voice input.

server.close():-Terminate the SMTP session and close the connection. Return the result of the SMTP close command.

- **googlesearch():-** It is a python library used for scraping google search results.
- **webbrowser():**-In Python, web browser module provides a high-level interface which allows displaying Web-based documents to users.

  webbrowser.open(url, new=0, autoraise=True):- Display *url* using the default browser. If *new* is 0, the *url* is opened in the same browser window if possible. If *new* is 1, a new browser window is opened if possible. If *new* is 2, a new browser page ("tab") is opened if possible. If *autoraise* is True, the window is raised if possible.

- **BeautifulSoup():**-Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favourite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

- **urllib.request():**-urllib.request is a Python module for fetching URLs (Uniform Resource Locators). It offers a very simple interface, in the form of

the *urlopen* function. This is capable of fetching URLs using a variety of different protocols. It also offers a slightly more complex interface for handling common situations - like basic authentication, cookies, proxies and so on. These are provided by objects called handlers and openers.

4.5 Segments in the program:-

The program contains various segments which gets triggered when it gets a voice command input from the user. After getting a command, it recognizes what the user has said and then it matches the user's query with the keyword specified in its conditional statements.

When a keyword is matched with any conditional statement then it executes the next command written in the program within that particular sub-section. It performs all its task with the above listed python libraries.

The whole project can be divided into subparts depending on their purpose which is listed below:-

- Greetings
- Web Searching
- Online Shopping sites
- Social Media sites
- Entertainment Features
- Tells the current time
- Launch any app
- Email Feature
- Shows NEWS
- Terminate Feature

4.5.1 Greetings Features:-

How it works: When it is turned on then depending on the time of the day, the assistant will greet the user. If the time is more than 5am and less than 12 pm noon, it will respond "Hello. Good Morning". If the time is more than 12 noon, it will respond "Hello. Good afternoon", likewise if the time is more than 6 pm, it will respond "Hello. Good evening". And when we give command as quit or

bye or terminate then, exit() will be called to terminate the program and it gives its final greeting to the user by responding “Thank you so much for your time. Good Bye”.

4.5.2 Online Web Searching, Online Shopping sites, Social media site:-

We can open any website by just our voice input. We only have to say the name of the website which we want to open. For example:- To search anything on Wikipedia, we only need to speak what do we want to search about and we have to add “Wikipedia” in our query.

How it works: When we give any voice input to the assistant, then it recognizes it and then it will search for website name in the user command using re.search(). Next, it will append the website name to https://www. and using **webbrowser** module the complete URL gets opened in the browser.

4.5.3 Entertainment features:-

How it works: This feature allows your voice assistant to plays song and videos in default media player. The user will say “play song” or “play videos”, the assistant will say, “Alright, playing music/videos for you” and then it plays it according to user’s voice input.

4.5.4 Tells the current time:-

How it works: “Can you tell me the current time?” or “what is the time now?” and assistant will tell you the current time of your timezone.

4.5.5 Launch any app:-

Say “launch notepad” or “can you please launch notepad” or “Garp launch notepad” etc. and the assistant will launch that system application for you.

How it works: If we have said the word **notepad** in our command then it will search for application name(if it is present in our system) then it will start this with the help of python library **“os”** and **os.startfile().**

4.5.6 Email feature:-

We can also ask your desktop assistant to send the email.

How it works: If we have said the word **email** in our command then the assistant will ask for typing the recipient address manually, and after this it will ask for the content of the email via voice input. The **smtplib module** defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon. Sending mail is done with Python's smtplib using an SMTP server. First it will initaite gmail SMTP using **smtplib.SMTP()**, then identify the server using **ehlo()** function, then encypting the session **starttls(),** then login to your mailbox using **login()**, then sending the message using **sendmail().**

4.5.7 Shows NEWS:-

Garp can also tell you the latest news update. The user just has to say "news" or "tell me the news for today" or any query which contains the word "news".

How it works: If we have said the phrase **news for today** in our command then it will scrape data using Beautiful Soup from Google News RSS() and read it for us. For convenience I have set number of news limit to 10, which can be increased or decreased as per the need. And then we can manually click on any link and the whole related article will be opened in a new tab in our default browser.

4.5.8 Terminate Feature:-

It can terminate by itself just by giving command which must contain the keyword, **"quit"**, **"bye"**, or **"terminate".**

How it works: When we give command as quit or bye or terminate then, exit() will be called to terminate the program and it gives its final greeting to the user by responding "Thank you so much for your time. Good Bye".