MVJ COLLEGE OF ENGINEERING, BENGALURU- 560067

(Autonomous Institution Affiliated to VTU, Belagavi)

Department of Computer Science and Engineering

Software Development Club Internship

Internship Report

On

Personal Desktop Voice Assistant Using Python

Submitted by

Abhinav Lakshmi Majeti 1MJ21CS003

Rahul Mishra 1MJ21CS170

U Giridharan 1MJ21CS233

Gokul Naga Satvik K 1MJ21CG025

Under the guidance of

Prof. Thejashwini M

Assistant Professor

Department of AI & ML

MVJ College of Engineering

In Partial fulfillment for the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

2022-23

MVJ COLLEGE OF ENGINEERING, BENGALURU- 560067

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF Computer Science and Engineering

CERTIFICATE

Certified that the internship titled Software Development Club Internship was carried out by Abhinav Lakshmi Majeti, Rahul Mishra, U Giridharan, Gokul Naga Satvik K in partial fulfillment for the award of degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2022-2023. It is certified that all corrections / suggestions indicated during the internal assessment have been incorporated in the internship report deposited in the department library. The internship report has been approved as it satisfies the academic requirements in respect of internship work prescribed by the institution for the said degree.

Signature of Guide Signature of HOD Signature of Principal

MVJ COLLEGE OF ENGINEERING, BENGALURU- 560067

(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF

Computer Science and Engineering

DECLARATION

We, Abhinav Lakshmi Majeti(1MJ21CS003), Rahul Mishra(1MJ21CS170), U Giridharan(1MJ21CS233), Gokul Naga Satvik K(1MJ21CG025), students of Second Semester B.E., Department of Computer Science and Engineering and Computer Science and Design, MVJ College of Engineering, Bengaluru - 560067, hereby declare that the Software Development Club Internship has been carried out by us and submitted in partial fulfillment for the award of the degree of Bachelor of Engineering in Computer Science and Engineering during the year 2022-2023. Further we declare that the content of the report has not been submitted previously by anybody for the award of any degree or diploma to any other University.

Place: Bengaluru

Date:

Name: Signature

Abhinav Lakshmi Majeti 1MJ21CS003

Rahul Mishra 1MJ21CS170

U Giridharan 1MJ21CS233

Gokul Naga Satvik K 1MJ21CG025

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible, success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, with gratitude we acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

We are thankful to our Principal Dr. Mahabaleshwarappa, for his encouragement and support throughout the internship.

We are thankful to our Vice-Principal Dr. M Brindha, for her encouragement and support throughout the internship.

We are thankful to our Dr. Lourdu Antony Raj, COE for his incessant encouragement & all the help during the internship.

We are also thankful to our Dr. M B Sudhan, HOD/CSE for his incessant encouragement & all the help during the internship.

We consider it a privilege and honor to express our sincere gratitude to our guide Ms.Thejashwini, Assistant Professor, Dept. of CSE for her valuable guidance throughout the tenure of this internship, and whose support and encouragement made this work possible.

It's also a great pleasure to express our deepest gratitude to all faculty members of our department for their cooperation and constructive criticism offered, which helped us a lot during our project work. We thank all the technical and non-technical staff of the Computer Science and Engineering department, MVJCE for their help

ABSTRACT

The project aims to develop a private assistant for computers. Desktop Personal Assistant draws its inspiration from virtual assistants like Google Assistant for Android, Siri for iOS. It provides a user-friendly interface for completing a spread of tasks by employing certain well-defined commands. Users can interact with the assistant through voice commands. As a private assistant, it assists the end-user with day-to-day activities like searching queries, reading the latest news, searching locations on maps, live weather, opening websites. The software uses a device's microphone to receive voice requests while the output takes place at the system's speaker. The Desktop Personal assistant is built mainly using Python.

Table of Content

List of Figures

Chapter 1
INTRODUCTION

Preamble: Voice assistants have become increasingly popular in recent years due to their ability to simplify daily tasks and provide a more convenient user experience. In this chapter, we will be introducing the concepts of voice assistants and their usage in today's world.

Our digital life is decided by innovations. Especially in recent years, more innovative technologies were developed to ease our professional lifestyle. Intelligent Personal Assistant has proved to be the most vital innovation in terms of easing our lives and providing a hands-free experience. We are building a PC Personal Assistant that works on voice commands and executes the user query. Our project, PC Personal Assistant is built mainly using python. The software uses a device's microphone to accept voice requests while the output takes place at the system's speaker. It's a mixture of varied technologies: voice recognition, voice analysis, and language processing. When a user asks an assistant to perform a task, the natural language audio signal is converted into digital data which will be analyzed by the software. Once digitized, several models are used to transcribe the audio signal to textual data.

Keywords are used to perform certain tasks and if those keywords are present within the text then the particular task is performed. For instance, the 'translate' keyword is employed for the translation of the word from one language to a different so if user audio contains that keyword then the translation function is going to be activated and translation tasks are going to be performed by an assistant. So, keywords are defined for particular tasks and if that word is present in your audio the actual tasks are going to be performed. Tasks that our Pc assistant can perform are searching the information from Wikipedia and reading the information, fetching top news from Google News and reading the news, telling the present weather of a particular city, searching locations on Google Maps, etc.

Voice assistants (VA) are a type of voice-enabled artificial intelligence (AI). AI refers to some level of intelligence displayed by digital interfaces, or the ability of algorithms to mimic intelligent human behavior. Although AI refers to "cognitive" functions that we associate with the human mind, including problem solving and learning (Syam and Sharma, 2018). VA in the form of mobile applications include Apple's Siri, Amazon's Alexa,

Google Assistant, Microsoft Cortana, and among smart speaker offerings are Amazon's Echo, Google's Home, and Apple's Home. In any form, VA are revolutionizing consumers' consumption culture and becoming a larger part of consumers' social lives. Such VA enable users to navigate, listen to music, send text messages, control smart home devices, make a phone call, order food, order an Uber ride or pizza, and so on.

Chapter 2

Literature Survey

Preamble: This chapter contains the research papers, "Humanizing voice assistant: The impact of voice assistant personality on consumers' attitudes and behaviors - Atieh Poushneh" and "AI-Based Voice Assistant Systems: Evaluating from the Interaction and Trust Perspectives - Farzaneh Nasirian & Mohsen Ahmadian" referred to help with the creation of the project.

2.1 Humanizing voice assistant: The impact of voice assistant personality on consumers' attitudes and behaviors - Atieh Poushneh

VAP refers to the attribution of cognitive, emotional, and social human characteristics to VA in order to humanize interactions with consumers. Such humane qualities prompt consumers to have more engaged interactions with VA. Previous literature developed scales to evaluate mobile interface personality (Johnson et al., 2020), website personality (Chen and Rodger, 2006), and brand personality (Aaker, 1997). Chen and Rodger demonstrated five elements of website personality: intelligence, fun, organization, candidness, and sincerity. The intelligent personality trait describes a website that is proficient, sophisticated, effective, and systematic. A website that manifests a fun personality should be engaging, exciting, and vital. The organized personality trait reflects a website that is neither confusing nor overwhelming. A website exhibiting a candid personality refers to the resource being straightforward. Lastly, the sincere personality trait refers to a website that is heartfelt and down-to-earth.

In addition, Aaker developed a scale to measure brand personality consisting of five dimensions, including sincerity, excitement, competence, sophistication, and ruggedness (Aaker, 1997). Sincerity personality traits include honesty, originality, being down-to-earth, friendliness, and sentimentality. Excitement refers to the degree of talkativeness, vitality, independence, freedom, happiness, and energy shown in a brands' personality trait (Lin, 2010). Excitement reflects the traits related to being daring, up-to-date, contemporary, cool, young, trendy, imaginative, unique, and independent. Competence describes reliability, security, intelligence, confidence, and success (Aaker, 1997). The sophistication trait refers to elements of glamour, upper financial classes, charm, the feminine, and visual appeal. Ruggedness relates to toughness, masculinity, and an affinity for nature (Aaker,

1997). Prior research has shown that purchase behavior is linked with individual personality traits that shape the buyer-seller relationship (Barrick and Mount, 1991), consumers' responses to brand choice (Aaker, 1997; Rau et al., 2017), and their purchase intentions (Poddar et al., 2009). Websites' personalities can influence consumers' attitudes, both build and improve relationships with customers, and prevent website failure (Palmer and Griffith, 1998). Further, website personalities influence consumers' attitudes and their website interactions (Rodgers and Thorson, 2000) through flow experience (e.g., Heller et al., 2015; Rau et al., 2017), which impacts consumers' purchase intentions.

2.1 AI-Based Voice Assistant Systems: Evaluating from the Interaction and Trust Perspectives - **Farzaneh Nasirian & Mohsen Ahmadian**

The aim of this work is to better understand the quality preferences of users, which affect adoption of AI based technologies. We found that interaction quality is an important factor for adoption of VAS, as an example of AI-based systems. As shown in Figure 2, the hypotheses regarding interaction quality is supported at the 0.05 significance level. Moreover, the positive relationship between both trust and personal innovativeness with intention to use is supported at the 0.05 significance level, and the explanatory power of our model is 54.9%. Also, it is interesting that the relationship between both information quality and system quality with trust is not supported whereas they were supported in previous researches (Kim et al. (2004); Zahedi and Song (2008); Chen and Cheng (2009); Teo et al. (2008); Zhou (2013)). This result for AI-based technologies could be described by Herzberg's motivation hygiene theory (Herzberg 2005).

Considering this theory and the role of satisfaction on trust studied by Garbarino and Johnson (1999), we believe in our study, both information and system quality are the factors that should be at a high level of quality to avoid loss of trust. Indeed, these two factors are hygiene factors which should be at an acceptable level that meets users' expectations. On the other hand, interaction quality in VAS technologies is the motivation factor that adds value, and creates trust.

Upon diffusion of innovation model (Rogers 1962), innovators and early adopters are two types of users who pick up a technology earlier than others. Indeed, they are curious to experience new technologies whereas there would be many uncertainties and risks. AI-

based technologies are new branches of science based on several disciplines which are in the early stage of their life cycle. Accordingly, people who intend to use them can be considered as early adopters. In line with this model, we tested the impact of personal innovativeness on users' trust and intention. Although the results of our sample do not illustrate the moderating role of personal innovativeness on the relationship between trust and intention of using AI based VAS, we believe by increasing our sample size, this hypothesis can be supported.

The results of this study provide valuable ideas to both researchers and practitioners. This research has extended the literature in three significant ways. First, we applied many different theoretical perspectives including ISSM, SET, and HCI, and brought them in adoption literature to develop a new adoption model that can better describe adoption of new emerging technologies. Multiple theoretical perspectives allow us to investigate phenomena in different aspects, and develop a more comprehensive model to better interpret them and the relationships between them.

Chapter – 3

Features and Requirements

Preamble: In this chapter, we will explore the different features of the voice assistant, its configurations, hardware requirements. We will also delve into the language used for the project, Python, the python modules used and the flowchart of the project.

3.1 Features:

It has a login system. Voice assistant is only activated after the login is successful.

The different functions of Friday are:

- Greets the user according to the time of day
- Can search anything on Wikipedia, google
- Can open Google, GitHub, Spotify, YouTube, and other websites
- Can tell time
- Can open the control panel, settings, calculator, etc.
- Can search any location on Google Maps
- Can send automated WhatsApp message to any phone number
- Can tell random jokes
- Can tell the weather of any city in the world
- Can read the current news
- It has a dedicated to-do list/memo

Programming Language Used: Python

3.2 Configuration:

- Processors: Intel Atom processor or Intel Core i3 processor
- RAM: At least 2 GB
- Disk space: 1 GB
- Operating systems: Windows* 7 or later, macOS, and Linux
- Python versions: 2.7.X, 3.6.X
- Must have a functional Microphone
- Internet Connection
- IDE: Visual Studio Code, PyCharm

3.3 Python:

Python is an OOPs (Object Oriented Programming) based, high level, interpreted programming language. It is a robust, highly useful language focused on rapid application development (RAD). Python helps in easy writing and execution of codes. Python can implement the same logic with as much as 1/5th code as compared to other OOPs languages.

Python provides a huge list of benefits to all. The usage of Python is such that it cannot be limited to only one activity. Its growing popularity has allowed it to enter into some of the most popular and complex processes like Artificial Intelligence (AI), Machine Learning (ML), natural language processing, data science etc. Python has a lot of libraries for every need of this project. For ZIRA, libraries used are speech recognition to recognize voice, Pyttsx for text to speech, selenium for web automation etc.

Python is reasonably efficient. Efficiency is usually not a problem for small examples.
Python is meant to be an easily readable language. Its formatting is visually uncluttered and often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but rarely used. It has fewer syntactic exceptions and special cases than C or Pascal.

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in

indentation signifies the end of the current block. Thus, the program's visual structure accurately represents its semantic structure.

3.4 Python Modules Used:

Pyttsx3: It is a text-to-speech conversion library in Python. Unlike alternative libraries, it works offline, and is compatible with both Python 2 and 3.

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

Wikipedia: Wikipedia API for Python

Pyaudio: It provides Python bindings for PortAudio v19, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple macOS.

Pywhatkit: It is a Python library with various helpful features. It's easy-to-use and does not require you to do any additional setup. Currently, it is one of the most popular libraries for WhatsApp and YouTube automation. New updates are released frequently with new features and bug fixes.

Selenium: Python language bindings for Selenium WebDriver. The *selenium* package is used to automate web browser interaction from Python.

Tkinter: It is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

lxml: Powerful and Pythonic XML processing library combining libxml2/libxslt with the Element Tree API.

wego: Module for weather printing format in console.

bs4: This is a dummy package managed by the developer of Beautiful Soup to prevent name squatting. The official name of PyPI's Beautiful Soup Python package is beautifulsoup4. This package ensures that if you type *pip install bs4* by mistake you will end up with Beautiful Soup.

Pyaudio: PyAudio provides Python bindings for PortAudio v19, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a variety of platforms, such as GNU/Linux, Microsoft Windows, and Apple macOS.

3.5 Choosing what python module to use:

There are multiple speech recognition modules in python like apiai, assemblyai, google-cloud-speech, pocketsphinx and SpeechRecognition.

There is one package that stands out in terms of ease-of-use: SpeechRecognition. Recognizing speech requires audio input, and SpeechRecognition makes retrieving this input really easy. Instead of having to build scripts for accessing microphones and processing audio files from scratch, SpeechRecognition will have you up and running in just a few minutes. The SpeechRecognition library acts as a wrapper for several popular speech APIs and is thus extremely flexible. One of these—the Google Web Speech API—supports a default API key that is hard-coded into the SpeechRecognition library. That means you can get off your feet without having to sign up for a service. The flexibility and ease-of-use of the SpeechRecognition package make it an excellent choice for any Python project. However, support for every feature of each API it wraps is not guaranteed. You will need to spend some time researching the available options to find out if SpeechRecognition will work in your case.

Chapter 4:

Implementation

Preamble: In this chapter, the implementation of the virtual desktop assistant is mentioned, including speech recognition, which modules to choose for the voice assistant and how to install said modules for the project. There is also a description on the different classes from the modules used, how to use them, how to work with the device microphone and how to handle unrecognizable speech.

4.1 Speech Recognition:

Speech recognition has its roots in research done at Bell Labs in the early 1950s. Early systems were limited to a single speaker and had limited vocabularies of about a dozen words. Modern speech recognition systems have come a long way since their ancient counterparts. They can recognize speech from multiple speakers and have enormous vocabularies in numerous languages. The module speechRecognition was used.

The first component of speech recognition is, of course, speech. Speech must be converted from physical sound to an electrical signal with a microphone, and then to digital data with an analog-to-digital converter. Once digitized, several models can be used to transcribe the audio to text.

Most modern speech recognition systems rely on what is known as a Hidden Markov Model (HMM). This approach works on the assumption that a speech signal, when viewed on a short enough timescale (say, ten milliseconds), can be reasonably approximated as a stationary process—that is, a process in which statistical properties do not change over time.

In a typical HMM, the speech signal is divided into 10-millisecond fragments. The power spectrum of each fragment, which is essentially a plot of the signal's power as a function of frequency, is mapped to a vector of real numbers known as cepstral coefficients. The dimension of this vector is usually small—sometimes as low as 10, although more accurate systems may have dimension 32 or more. The final output of the HMM is a sequence of these vectors.

To decode the speech into text, groups of vectors are matched to one or more phonemes—a fundamental unit of speech. This calculation requires training, since the sound of a phoneme varies from speaker to speaker, and even varies from one utterance to another by the same speaker. A special algorithm is then applied to determine the most likely word (or words) that produce the given sequence of phonemes.

One can imagine that this whole process may be computationally expensive. In many modern speech recognition systems, neural networks are used to simplify the speech signal using techniques for feature transformation and dimensionality reduction before HMM recognition. Voice activity detectors (VADs) are also used to reduce an audio signal to only the portions that are likely to contain speech. This prevents the recognizer from wasting time analyzing unnecessary parts of the signal.

4.2 Flowchart of the project:

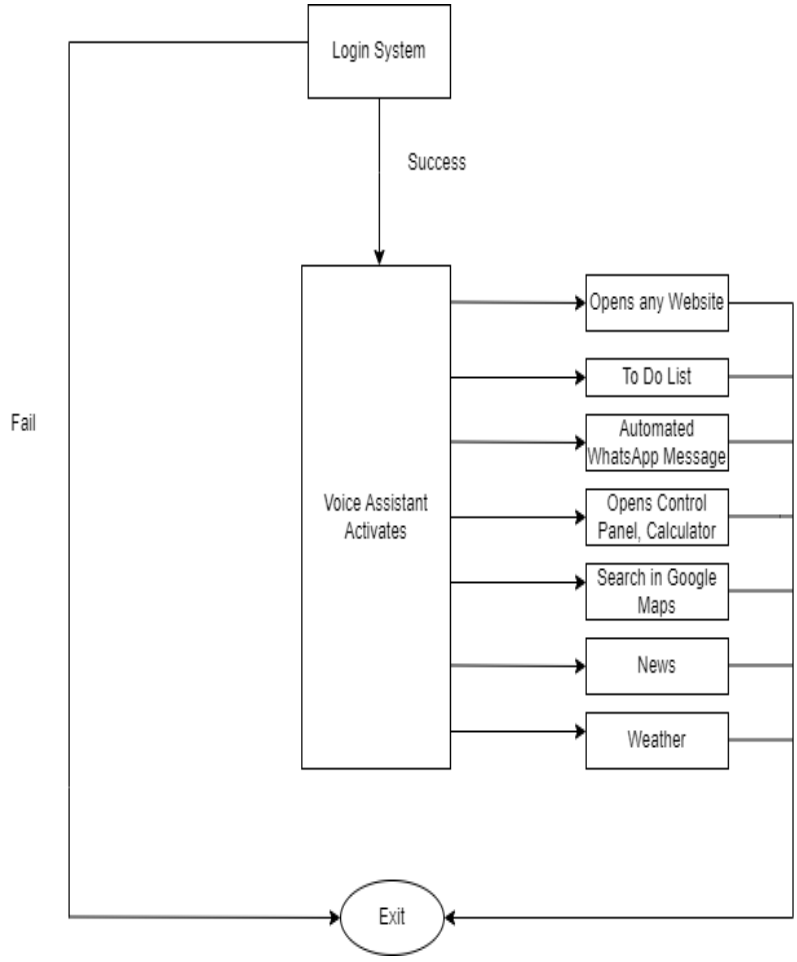

Fig 4.1: Flow of the project

4.3 Installing SpeechRecognition:

SpeechRecognition is compatible with Python 2.6, 2.7 and 3.3+, but requires some additional installation steps for Python 2. You can install SpeechRecognition from a terminal with pip:

```
Shell

pip install speechRecognition
```

SpeechRecognition will work out of the box if all you need to do is work with existing audio files. Specific use cases, however, require a few dependencies. Notably, the PyAudio package is needed for capturing microphone input.

4.4 The Recognizer Class:

All of the magic in SpeechRecognition happens with the Recognizer class.

The primary purpose of a Recognizer instance is, of course, to recognize speech. Each instance comes with a variety of settings and functionality for recognizing speech from an audio source.

Creating a Recognizer instance is easy. In your current interpreter session, just type:

Each Recognizer instance has seven methods for recognizing speech from an audio source using various APIs.

These are:

- **recognize_bing():** Microsoft Bing Speech
- **recognize_google():** Google Web Speech API
- **recognize_google_cloud():** Google Cloud Speech - requires installation of the google-cloud-speech package
- **recognize_houndify**(): Houndify by SoundHound
- **recognize_ibm**(): IBM Speech to Text
- **recognize_sphinx():** CMU Sphinx - requires installing PocketSphinx
- **recognize_wit():** Wit.ai

Of the seven, only recognize_sphinx() works offline with the CMU Sphinx engine. The other six all require an internet connection.

Since SpeechRecognition ships with a default API key for the Google Web Speech API, you can get started with it right away. For this reason, we'll use the Web Speech API in this guide. The other six APIs all require authentication with either an API key or a username/password combination. For more information, consult the SpeechRecognition docs.

```
Python >>
>>> r = sr.Recognizer()
```

Each recognize_*() method will throw a speech_recognition RequestError exception if the API is unreachable.

For recognize_sphinx(), this could happen as the result of a missing, corrupt or incompatible Sphinx installation.

For the other six methods, RequestError may be thrown if quota limits are met, the server is unavailable, or there is no internet connection.

All seven recognize_*() methods of the Recognizer class require an audio_data argument. In each case, audio_data must be an instance of SpeechRecognition's AudioData class.

4.5 Working with Microphones:

To access your microphone with SpeechRecognizer, you'll have to install the PyAudio package. Go ahead and close your current interpreter session, and let's do that.
Installing PyAudio:
The process for installing PyAudio will vary depending on your operating system.
macOS:
For macOS, first you will need to install PortAudio with Homebrew, and then install PyAudio with pip:

```
Shell
$ brew install portaudio
$ pip install pyaudio
```

Windows:

On Windows, you can install PyAudio with pip:

Shell

```
$ pip install pyaudio
```

Make sure your default microphone is on and unmuted. If the installation worked, you should see something like this:

Shell

```
A moment of silence, please...
Set minimum energy threshold to
600.4452854381937 Say something!
```

4.6 The Microphone Class

Open up another interpreter session and create an instance of the recognizer class.

Python >>

```
>>> import speech_recognition as sr
>>> r = sr.Recognizer()
```

Now, instead of using an audio file as the source, you will use the default system microphone. You can access this by creating an instance of the Microphone class.

```
>>> mic = sr.Microphone()
```

The device index of the microphone is the index of its name in the list returned by list_microphone_names(). For example, given the above output, if you want to use the microphone called “front,” which has index 3 in the list, you would create a microphone instance like this:

Using listen() to Capture Microphone Input:

Just like the AudioFile class, Microphone is a context manager. You can capture input from the microphone using the listen() method of the Recognizer class inside of the with block. This method takes an audio source as its first argument and records input from the

Python >>

```
>>> # This is just an example; do not run
>>> mic = sr.Microphone(device_index=3)
```

source until silence is detected.

Once you execute the with block, try speaking “hello” into your microphone. Wait a moment for the

```
Python >>
>>> with mic as source:
...     audio =
r.listen(source)
```

interpreter prompt to display again. Once the “>>>” prompt returns, you’re ready to recognize the speech.

```
Python >>
>>> r.recognize_google(audio)
'hello'
```

If the prompt never returns, your microphone is most likely picking up too much ambient noise.
You can interrupt the process with +ctrl+c++ to get your prompt back.

To handle ambient noise, you’ll need to use the adjust_for_ambient_noise() method of the Recognizer class, just like you did when trying to make sense of the noisy audio file. Since input from a microphone is far less predictable than input from an audio file, it is a good idea to do this anytime you listen for microphone input.
After running the above code, wait a second for adjust_for_ambient_noise() to do its thing, then try speaking “hello” into the microphone. Again, you will have to wait a moment for the interpreter prompt to return before trying to recognize the speech.

Recall that adjust_for_ambient_noise() analyzes the audio source for one second. If this seems too long to you, feel free to adjust this with the duration keyword argument.

The SpeechRecognition documentation recommends using a duration no less than 0.5 seconds. In some cases, you may find that durations longer than the default of one second generate better results. The minimum value you need depends on the microphone’s ambient

environment. Unfortunately, this information is typically unknown during development.

4.7 Handling Unrecognizable Speech:

Try typing the previous code example into the interpreter and making some unintelligible noises into the microphone. You should get something like this in response:

```
Python

 Traceback (most recent call
last): File "<stdin>", line 1,
in <module>
     File "/home/david/real_python/speech_recognition_primer/venv/lib/python3.5/site-
  packages/speech_recognit if not isinstance(actual_result, dict) or
```

Audio that cannot be matched to text by the API raises an UnknownValueError exception. You should always wrap calls to the API with try and except blocks to handle this exception.

Chapter-5

Results

Preamble: In this chapter, multiple screenshots of the output of the virtual desktop assistant project using python. Screenshots for multiple functions like login prompts, google search, google maps search, wikipedia search, opening any given website, etc. have been provided.

Login success:

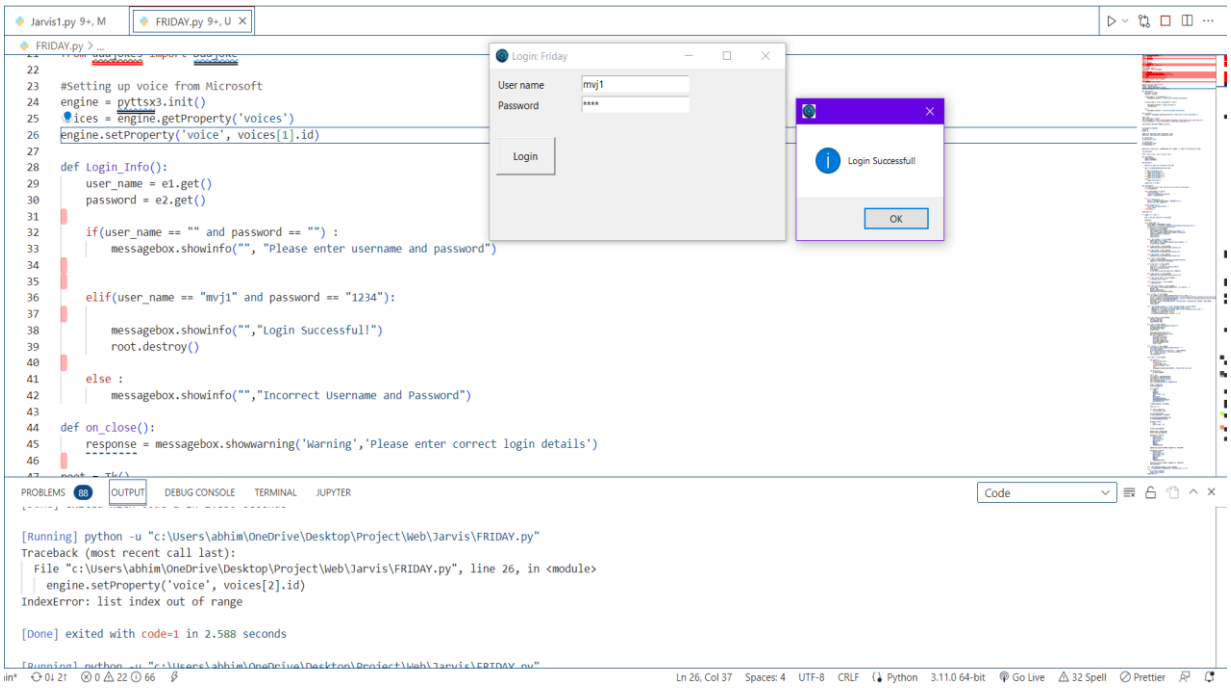

Fig 5.1: Login success prompt

- Login system is made using the Tkinter module of python. The login dialog box consists of username and password boxes and a login button. When the correct username and password is entered, login is successful and the voice assistant is activated.

Username: mvj1

Password: 1234

Login Fail:

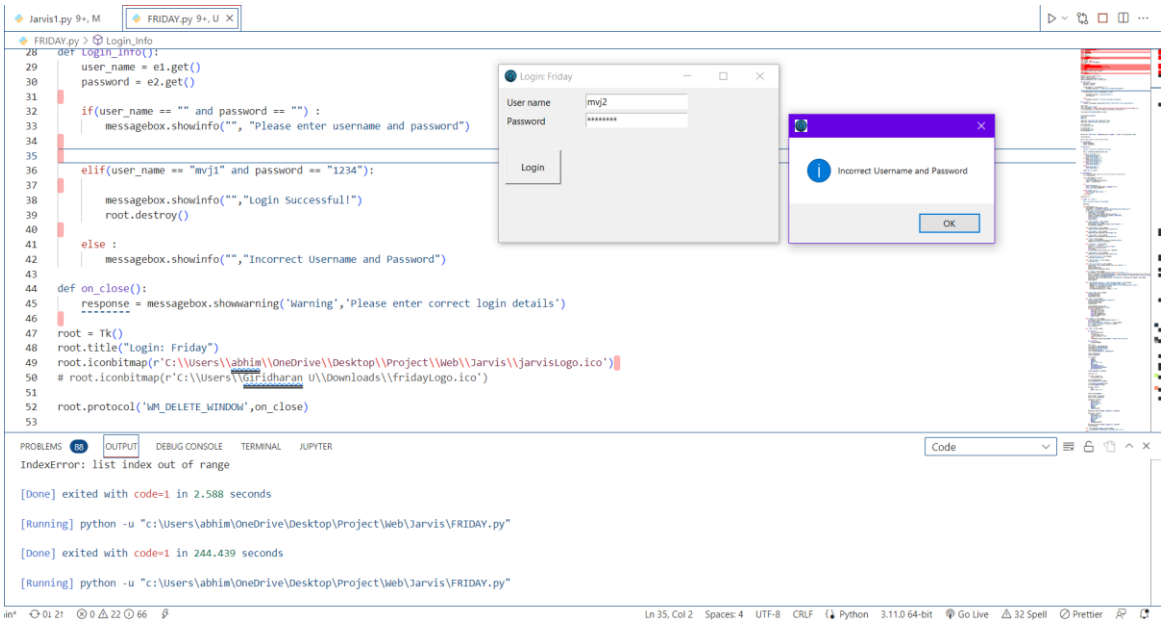

Fig 5.2: Login fail prompt

- When incorrect login credentials are entered, the login fails and the voice assistant is not activated.

Google Search:

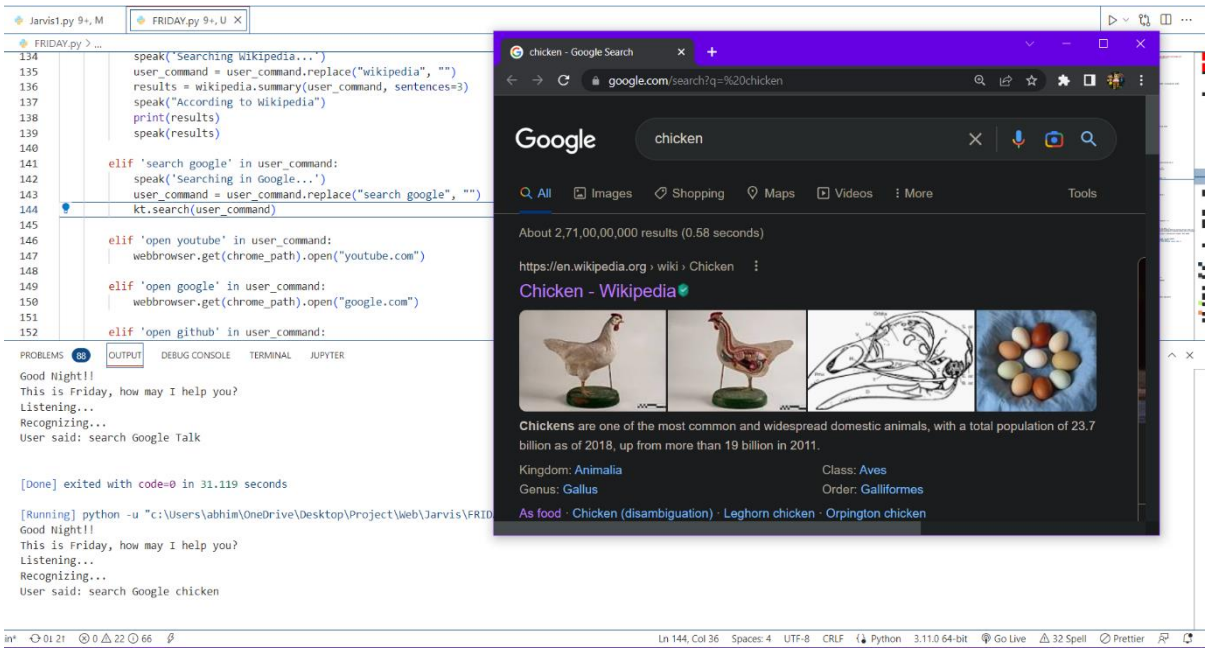

Fig 5.3: Google search result

- The voice assistant searches anything the user says in Google. This is done with the help of the pywhatkit module. The search function in pywhatkit, searches anything on Google..

Opens any website:

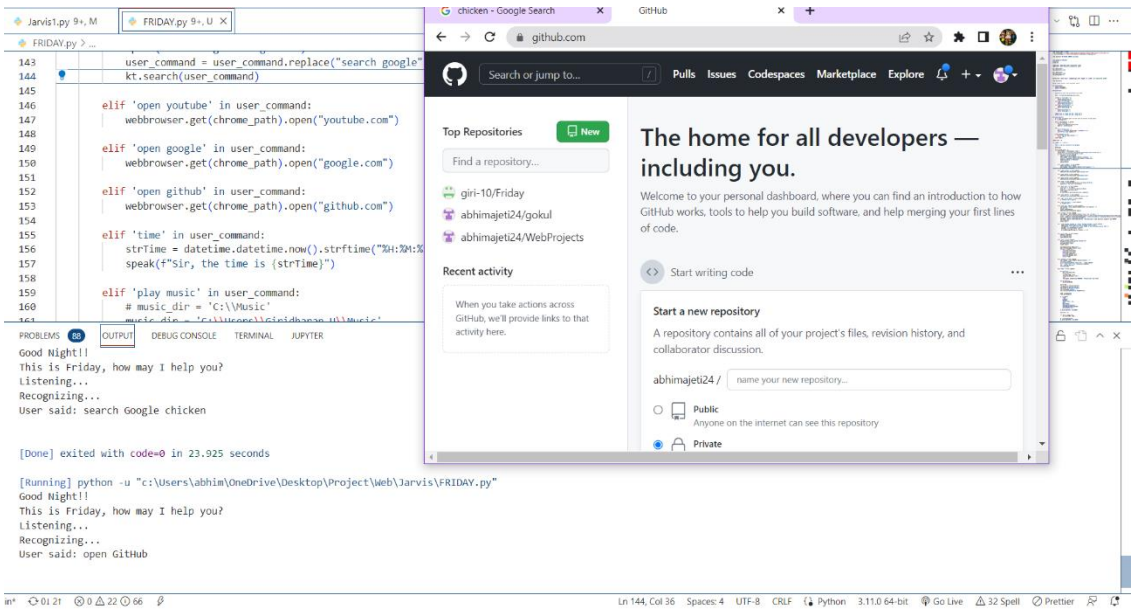

Fig 5.4: Opening GitHub website function

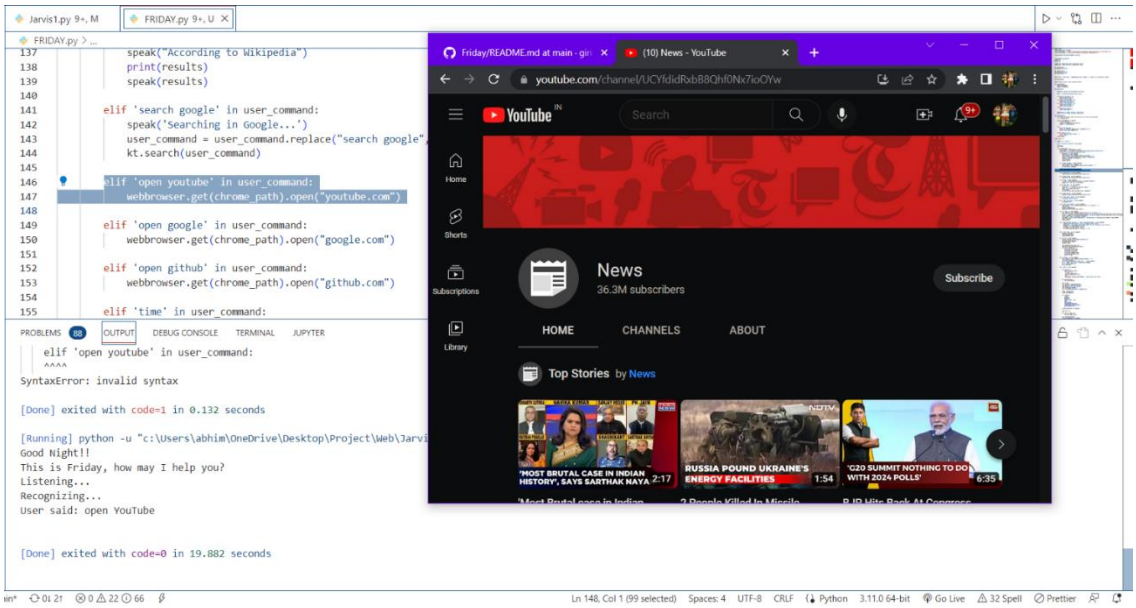

Fig 5.5: Opening YouTube website function

- This function opens any website the user asks.

Searches locations in Google Maps:

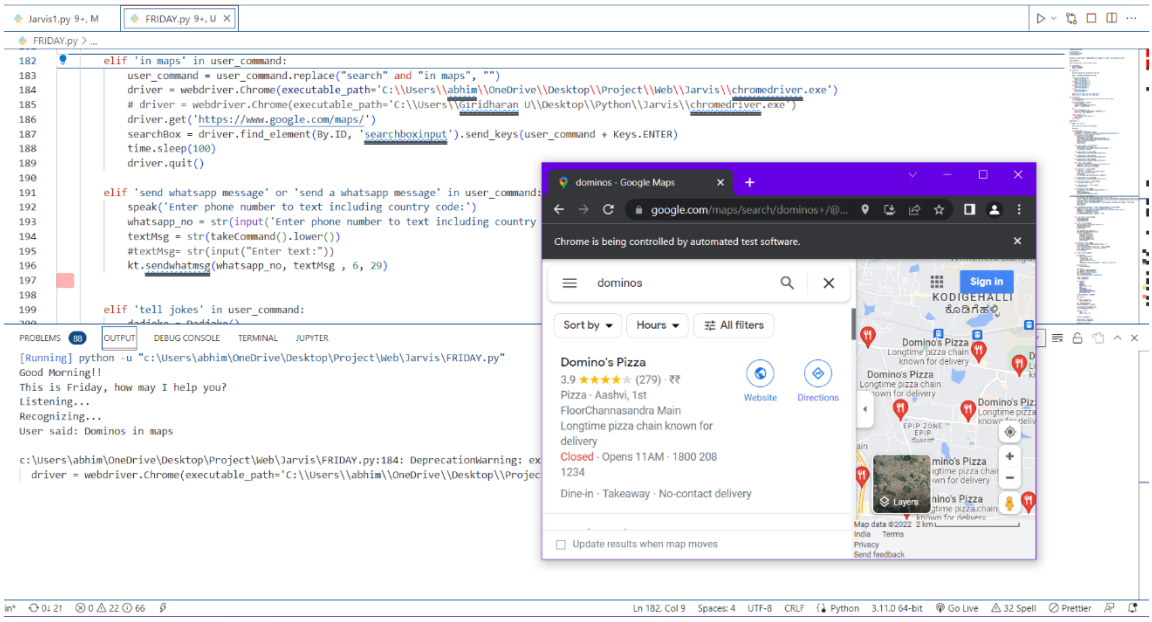

Fig 5.6: Location search in Google Maps

- Searches any location on maps. It uses selenium to automate the browser.

Opens Calculator, Control Panel:

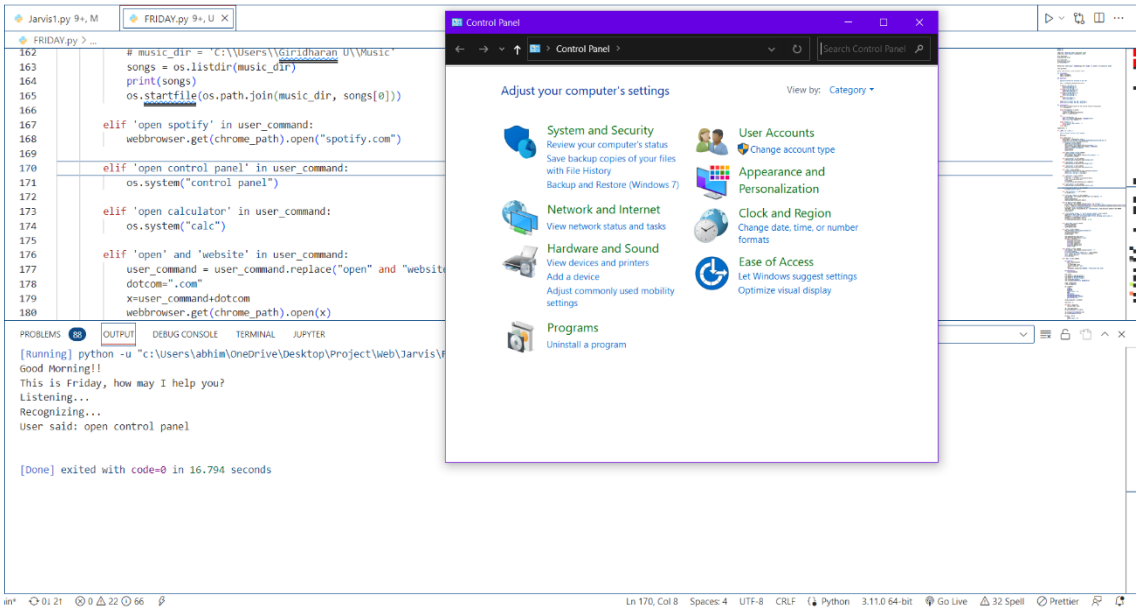

Fig 5.7: Opening control panel function

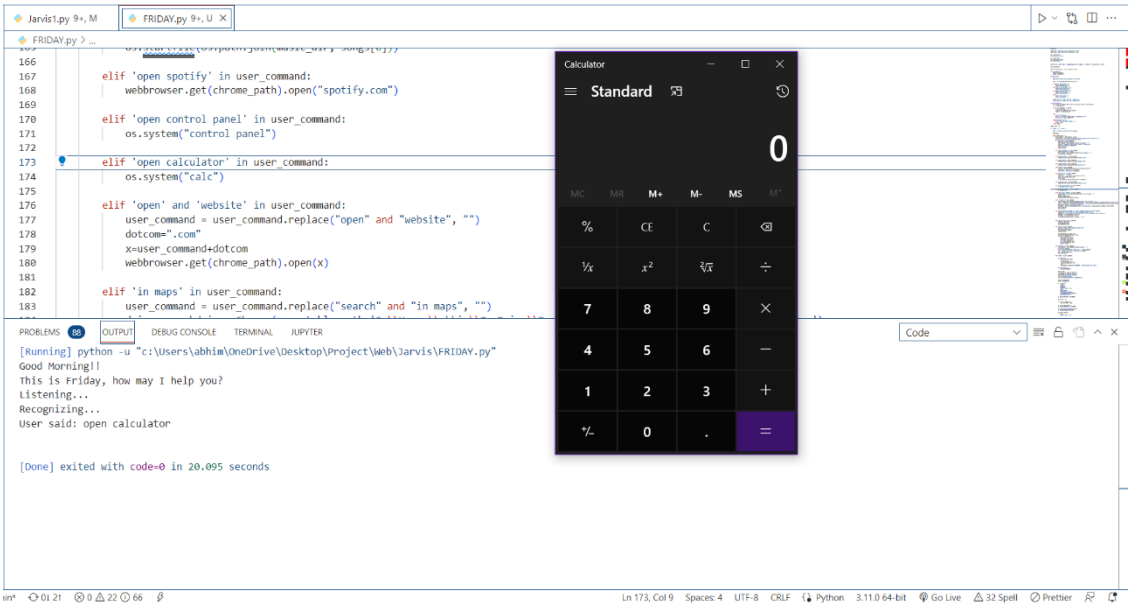

Fig 5.8: Opening calculator function

- Using this function the program can open the control panel or calculator.

Jokes:

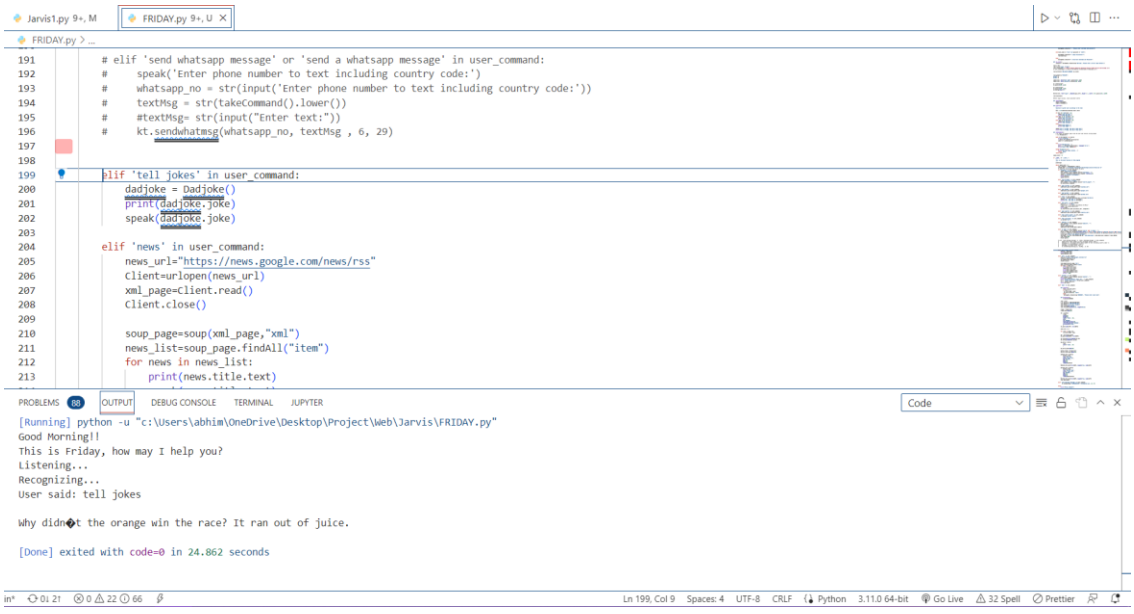

Fig 5.9: Jokes function

- Tell jokes using the pyjokes module**.**

Current News:

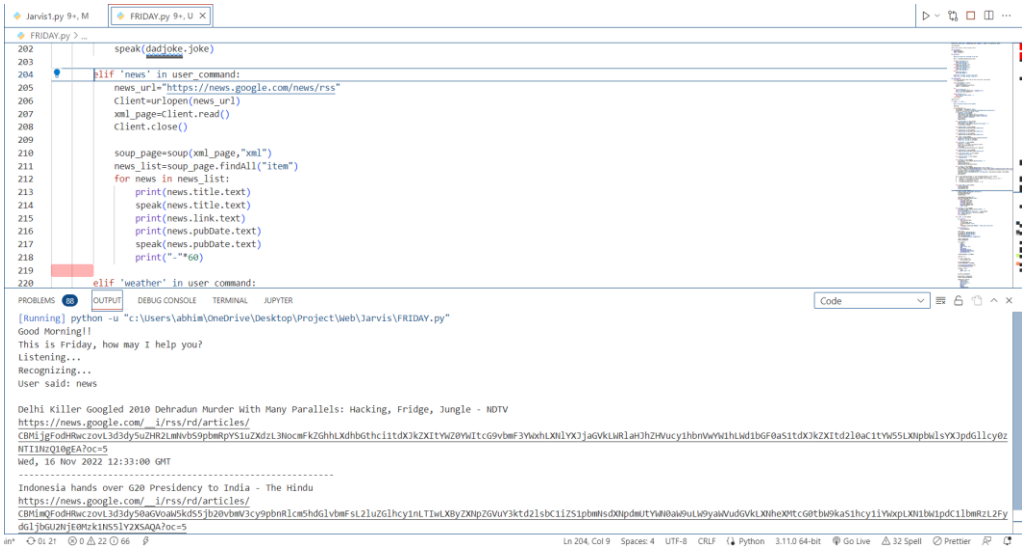

Fig 5.10: Read out current news function

- Reads the current headline from google news.

Weather Report of any city:

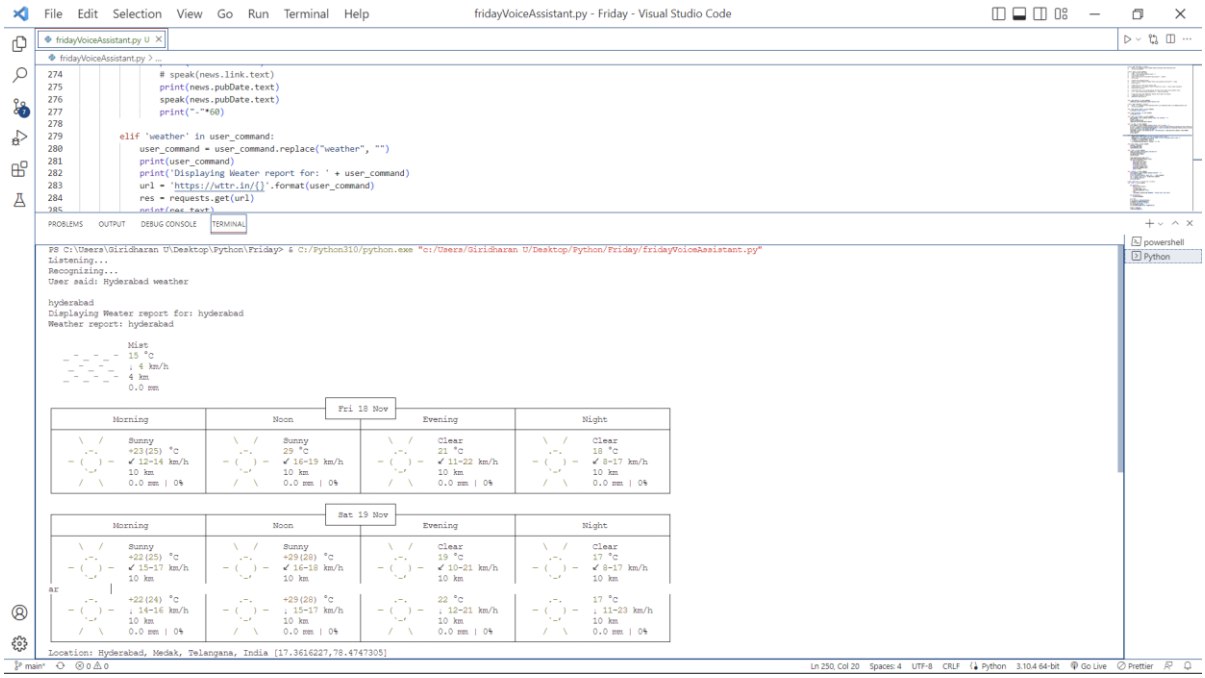

Fig 5.11: Weather report display of any city

- Shows the weather report of any city.

To do list:

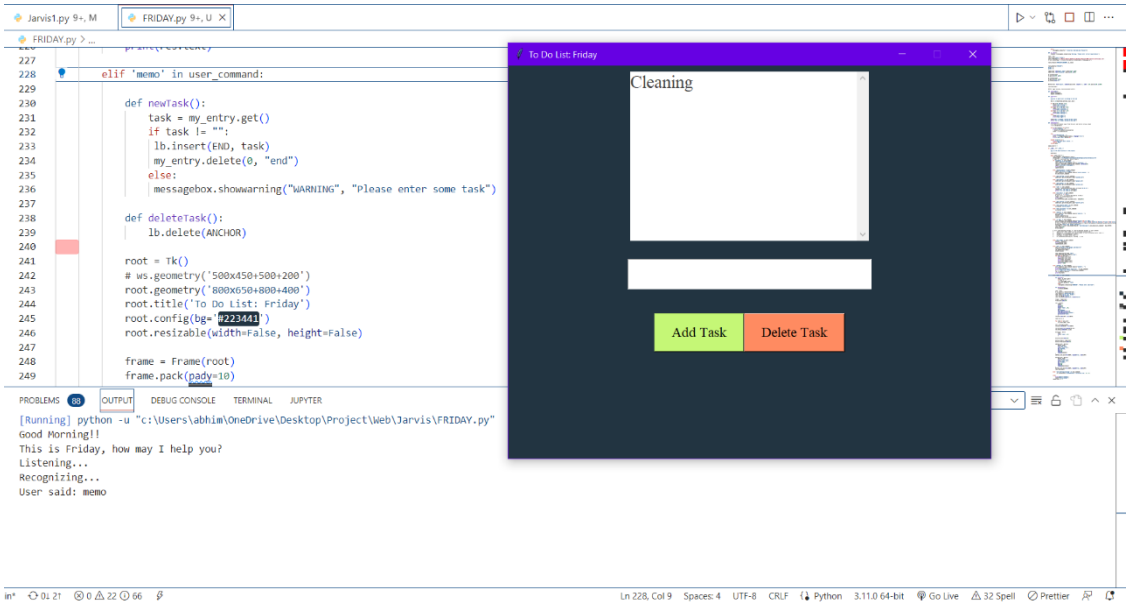

Fig 5.12: To-Do list function

- A to do list made using tkinter.

Play Music:

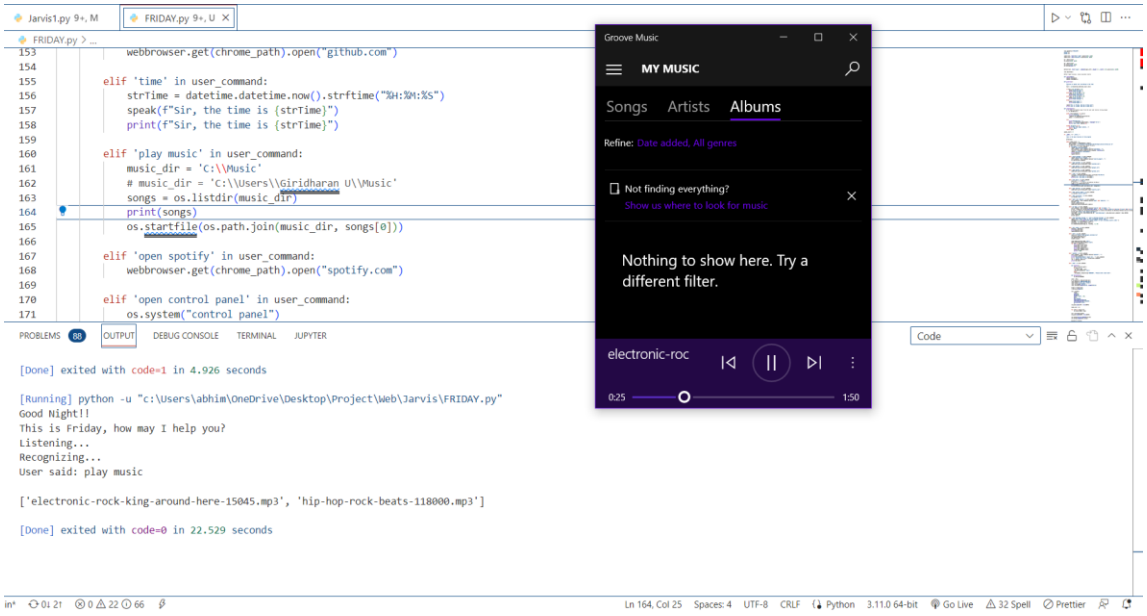

Fig 5.13: Play music function

- Plays locally stored music files.

Time:

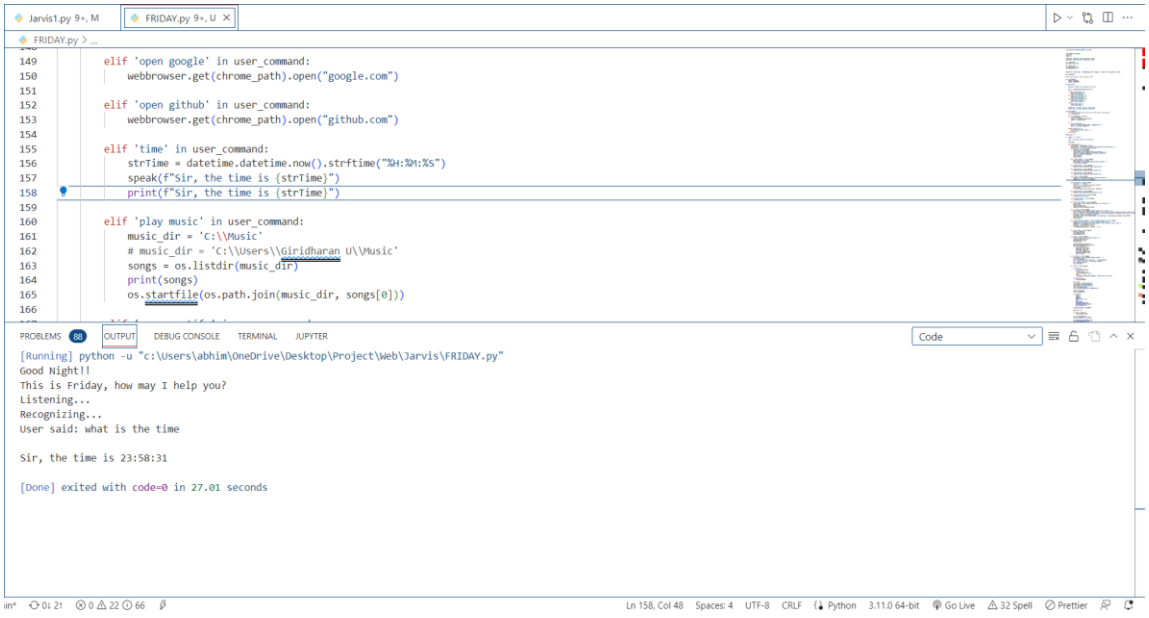

Fig 5.14: Current time display

- Tells time

Wikipedia Search:

Fig 5.15: Wikipedia search function

- Searches anything on Wikipedia and reads it out.

Wrong Command:

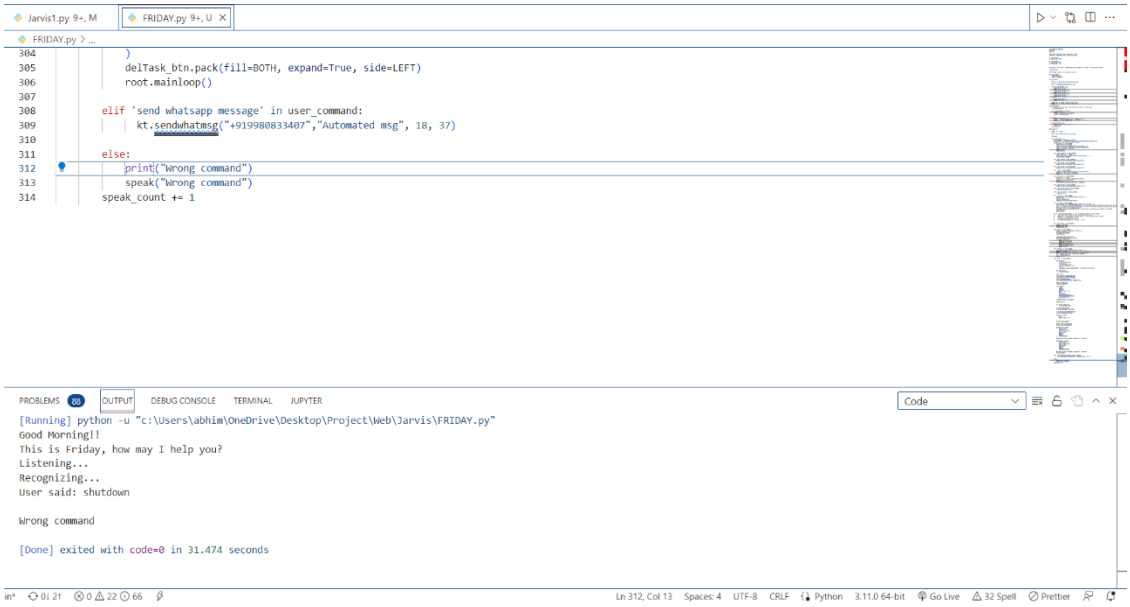

Fig 5.16: Wrong command error

- This is the message that is displayed when the wrong command is said.

Fig Chapter-6

Future Scope and Conclusion

Preamble: In this chapter, the future scope of desktop voice assistant using python on potential areas like improved natural language processing, personalization and multilingual support and other applications have been mentioned, also the conclusion of the project is described in this chapter.

6.1 Future Scope

The future scope of personal desktop voice assistants using Python is promising as the use of voice assistants continues to grow rapidly. With the advancements in natural language processing and machine learning, personal desktop voice assistants can become even more powerful and intelligent.

Potential areas for the future scope of personal desktop voice assistants are:

Improved Natural Language Processing (NLP) - As NLP technologies improve, personal voice assistants will be able to understand more complex commands and context, making them even more useful in everyday life.

Integration with Other Devices - Personal voice assistants could integrate with other devices in the home, such as smart home devices, to provide a seamless and convenient experience for users.

Personalization - With machine learning algorithms, voice assistants can learn user preferences and adapt to their behavior, providing more personalized and tailored recommendations.

Business Applications - Voice assistants can be used in a variety of industries, such as healthcare, customer service, and education, to provide more efficient and effective communication.

6.2 Conclusion

By using Python, developers can leverage a powerful and versatile programming language to create a personal voice assistant that can understand and respond to user queries, automate tasks, provide personalized recommendations, and integrate with other devices and services.

However, developing a personal desktop voice assistant requires a deep understanding of natural language processing, machine learning, and software engineering, as well as knowledge of APIs and SDKs for integrating with various services and devices.

Overall, a personal desktop voice assistant project using Python can be a valuable learning experience and a stepping stone towards building more complex and sophisticated voice-enabled applications in the future.

References:

- *Hoy, Matthew B. (2018). "Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants". Medical Reference Services Quarterly. **37** (1): 81–88. doi:10.1080/02763869.2018.1404391. PMID 29327988. S2CID 30809087.*
- *^ Klüwer, Tina. "From chatbots to dialog systems." Conversational agents and natural language interaction: Techniques and Effective Practices. IGI Global, 2011. 1–22.*
- *^ Daniel B. Kline (30 January 2017). "Alexa, How Big Is Amazon's Echo?". The Motley Fool.*
- *^ Markowitz, Judith. "Toys That Have a Voice". SpeechTechMag.*
- *^ Moskvitch, Katia. "The machines that learned to listen". BBC.* Retrieved *5 May* 2020.
- *^ Epstein, J; Klinkenberg, W. D (1 May 2001). "From Eliza to Internet: a brief history of computerized assessment". Computers in Human Behavior. **17** (3): 295–314. doi:10.1016/S0747-5632(01)00004-8. ISSN 0747-5632.*
- *^ Weizenbaum, Joseph (1976). Computer power and human reason : from judgment to calculation. Oliver Wendell Holmes Library Phillips Academy. San Francisco : W. H. Freeman.*
- *https://docs.python.org/3/*
- *https://pypi.org/*