CASPER AI: AN INTERPRETATION OF AIML WITH PYTHON

by

PROF. BIPASHA CHAKRABORTY BANIK

PROF. SAMPA DAS

TANMOY KHATUA

SUJAN DAS

GARGI MEMORIAL INSTITUTE OF TECHNOLOGY

Baruipur, Balarampur, Kolkata - 700144

Table Of Contents

ABSTRACT

As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of user. Speech recognition is the process of converting speech into text. This is commonly used in voice assistants like Alexa, Siri, etc. In Python there is an API called Speech-Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favourite IDE with the help of a single voice command. In the current scenario, advancement in technologies is such that they can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time.

Functionalities of this project include:

1. It can show locations on Map
2. It can open command prompt, your favourite IDE, notepad etc.
3. It can play music.
4. It can do Wikipedia searches for you.
5. It can open websites like Google, YouTube, etc., in a web browser.
6. It can give weather forecast.
7. It can give desktop reminders of our choice.
8. It can have some basic conversation.
9. Can do mathematical calculations also

Now the basic question arises in mind that how it is an AI? The virtual assistant that I have created is like if it is not an A.I, but it is the output of a bundle of the statement. But fundamentally, the mail purpose of A.I machines is that it can perform human tasks with the same efficiency or even more efficiently than humans. It is a fact that my virtual assistant is not a very good example of A.I., but it is an A.I.

CHAPTER-1

1.1 Introduction:

Artificial Intelligence when used with machines, it shows us the capability of thinking like humans. In this, a computer system is designed in such a way that typically requires interaction from human. As we know Python is an emerging language so it becomes easy to write a script for Voice Assistant in Python. The instructions for the assistant can be handled as per the requirement of user. Speech recognition is the Alexa, Siri, etc. In Python there is an API called Speech Recognition which allows us to convert speech into text. It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. In the current scenario, advancement in technologies are such that they can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time.

As the voice assistant is using Artificial Intelligence hence the result that it is providing are highly accurate and efficient. The assistant can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. The assistant is no less than a human assistant but we can say that this is more effective and efficient to perform any task. The libraries and packages used to make this assistant focuses on the time complexities and reduces time.

The functionalities include, it can send emails, it can read PDF, It can send text on WhatsApp, It can open command prompt, your favorite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc., in a web browser, It can give weather forecast, It can give desktop reminders of your choice. It can have some basic conversation.

Tools and technologies used are VS Code IDE for making this project, and I created all py files in VS Code. Along with this I used following modules and libraries in my project. pyttsx3, Speech- Recognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2, pyautogui, pyQt etc. I have created a live GUI for interacting with the CASPER as it gives a design and interesting look while having the conversation.

1.2 Present System:

We are familiar with many existing voice assistants like Alexa, Siri, Google Assistant, Cortana which uses concept of language processing, and voice recognition. They listen the command given by the user as per their requirements and performs that specific function in a very efficient and effective manner.

As these voice assistants are using Artificial Intelligence hence the result that they are providing are highly accurate and efficient. These assistants can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. These assistants are no less than a human assistant but we can say that they are more effective and efficient to perform any task. The algorithm used to make these assistant focuses on the time complexities and reduces time.

But for using these assistants one should have an account (like Google account for Google assistant, Microsoft account for Cortana) and can use it with internet connection only because these assistants are going to work with internet connectivity. They are integrated with many devices like, phones, laptops, and speakers etc.

1.3 Proposed System:

It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. CASPER is different from other traditional voice assistants in terms that it is specific to desktop and user does not need to make account to use this, it does not require any internet connection while getting the instructions to perform any specific task.

The IDE used in this project is VS Code. All the python files were created in VS Code and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e., pyttsx3, Speech Recognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyautogui, pyQt etc. I have created a live GUI for interacting with the CASPER as it gives a design and interesting look while having the conversation. With the advancement CASPER can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time.

Functionalities of this project include, It can send emails, It can read PDF, It can send text on WhatsApp, It can open command prompt, your favorite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc., in a web browser, It can give weather forecast, It can give desktop reminders of your choice. It can have some basic conversation.

1.4 Structural Integration

This system, which uses AIML as its primary functional component, also includes Python and gTTS as alternate enhancement compatibility mechanisms. The structure includes previous modules and the global R-R Model (Request-Response Model). constructed in Python. The speech recognition, pyttsx, pyaudio, pygame, and other python libraries are incorporated into the system to integrate with the AIML and gTTS events and be handled correctly. The gTTS engine transforms the raw data into text from the speech format and sends the parameters to the python interpreter, who then parses the text into the AIML kernel (AIML Bootstrap Kernel). The intelligence then suspects the data and generates the proper response, which is also parsed to the Python script.The text is coherently transferred to the gTTS engine, which now changes the text input to gTTS to an audio file that is temporarily retained in the system because this data is now comprising of the final output sought by the end-user.
Once more, Python implements this process and starts the audio file that contains the user reaction. The Python script tracks all command data, which is then shown on the terminal in raw-data format.

1.5 Voice

In our CASPER AI project, we have incorporated the default Windows voice as part of the text-to-speech functionality. The default Windows voice refers to the built-in voice synthesis capability provided by the Windows operating system. Here is how we have utilized the default Windows voice within our project:
Text-to-Speech Conversion: We have integrated the pyttsx3 library, which is a Python wrapper for text-to-speech engines, including the default Windows voice. pyttsx3 allows us to convert text into speech using the default voice provided by the Windows system.
Voice Feedback: Within our CASPER AI system, we generate responses or feedback in text format based

on user inputs or AI algorithms. To provide a more interactive and natural experience, we convert these text-based responses into spoken words using the default Windows voice. This enables our AI system to audibly communicate information or instructions to the user.

User Notifications: In addition to generating voice feedback for user interactions, we also utilize the default Windows voice for notifications or alerts. For example, when a task is completed or when specific conditions are met, our AI system can generate spoken notifications using the default Windows voice to inform the user.

By leveraging the default Windows voice, we make use of the builtin capabilities of the Windows operating system for text-to-speech conversion. This eliminates the need for external voice synthesis engines or APIs, simplifying the implementation and ensuring consistent voice output across Windows systems.

It's crucial to remember that depending on the Windows version and language packs installed, the default Windows voice's availability and quality may change. Users can select alternative voices or change voice parameters like pitch and speed by modifying the voice settings in the Windows Control Panel.

By utilizing the default Windows voice, we provide an accessible and platform-consistent voice-based experience within our Caspe AI project.

For web-based voice command, we have utilized Selenium for web testing and automation purposes. However, we have not specifically incorporated Selenium's voice-related features within our project. Our focus has primarily been on other aspects of AI development and functionality. While Selenium can be extended to incorporate voice interactions, such as voice commands or voice-controlled elements, it requires integration with additional libraries, APIs, or tools that provide voice recognition and text-to-speech capabilities. Depending on the requirements of CASPER AI project, we can explore integrating voice-related features using third-party libraries like SpeechRecognition for voice recognition and pyttsx3 for text-tospeech conversion.

By incorporating voice interactions, we can enable voice-based commands, speech-to-text input, or audio responses within CASPER AI system. This can enhance the user experience and provide a more natural and interactive interface.

To integrate Selenium with voice-related features, we have typically followed these steps:

Set up a voice recognition system: Integrate a voice recognition library or API, such as SpeechRecognition, to capture voice commands or user input. Process the voice input: Use the voice recognition library to convert the captured voice into text format, which can be further processed and understood by CASPER AI system. Perform actions based on voice commands: Utilize Selenium to interact with web elements and perform actions based on the recognized voice commands. For example, we can simulate clicks, input text, or navigate through web pages. Incorporate text-to-speech functionality: Integrate a text-to-speech library or API, such as pyttsx3, to convert textual responses from

CASPER AI system into voice output. This allows the system to provide spoken responses or feedback to the user.

By following these steps, we can extend CASPERAI project to support voice interactions using Selenium and additional voicerelated libraries or APIs. However, it's essential to consider the specific requirements and constraints of project and choose the appropriate tools and technologies accordingly

1.4 Tools and Technology Used:

SOFTWARE REQUIREMENT:

As the project is developed in python, we have used Anaconda for Python 3.11.3 and VS Code.

1. Visual Studio Code: Visual Studio Code, is a popular and powerful source code editor developed by

Microsoft. It is known for its versatility, extensive feature set, and wide range of supported programming languages. Here are some key points to know about VS Code:

Features include:

- Lightweight and Cross-platform: VS Code is a lightweight editor that runs on various platforms, including Windows, macOS, and Linux. It offers a consistent experience across different operating systems, making it accessible to a wide range of developers.
- Wide Language Support: VS Code supports an extensive range of programming languages out of the box, including popular ones like JavaScript, Python, C++, and Java. It provides syntax highlighting, code completion, and other language-specific features to enhance the development experience.
- Integrated Terminal: VS Code comes with an integrated terminal, allowing developers to run command-line tools and scripts without leaving the editor. It enables seamless interaction with the development environment and simplifies common tasks like running tests or executing build commands.
- Extensions Ecosystem: VS Code has a rich ecosystem of extensions that enhance its functionality and cater to specific programming languages, frameworks, or development workflows. These extensions provide additional features such as linting, debugging, version control integration, and more, making it highly customizable and adaptable to different needs.
- IntelliSense: VS Code offers IntelliSense, an intelligent code completion feature that suggests code snippets, variable names, function signatures, and other relevant information as you type. It significantly speeds up coding by reducing the need for manual typing and minimizing errors.
- Git Integration: VS Code integrates seamlessly with Git, a popular version control system. It provides visual cues for file changes, allows for easy staging and committing of code changes, and offers features like branch management and conflict resolution, making it convenient for collaborative software development.
- Debugging Capabilities: VS Code supports debugging for a wide range of programming languages and frameworks. It allows developers to set breakpoints, step through code, inspect variables, and analyze runtime behavior, enabling efficient debugging and issue resolution.
- Customizable and Extensible: VS Code is highly customizable, allowing developers to personalize the editor's appearance, keyboard shortcuts, and other settings according to their preferences. It also provides extensive APIs that enable the development of custom extensions to extend the editor's functionality.
- Integrated Version Control: VS Code includes built-in support for version control systems like Git. It provides a seamless interface for managing repositories, viewing changes, committing code, and resolving conflicts. This integration simplifies collaboration and makes it easy to track and manage code changes.
- Active Community and Support: VS Code has a vibrant and active community of users and contributors. This community actively develops and maintains extensions, shares tips and tricks, and provides support through forums and online resources. The community-driven nature of VS Code fosters continuous improvement and ensures a wealth of available resources for users.
- Overall, VS Code is a versatile and powerful code editor that combines essential features with extensive customization options and an active community. Its wide language support, integrated terminal, debugging capabilities, and rich extension ecosystem make it a top choice for developers across different programming disciplines.

Hardware Interfaces

- Processor: Intel CORE i5 processor with minimum 2.9 GHz speed.
- RAM: Minimum 4 GB.
- Hard Disk: Minimum 500 GB

Software Interfaces

- Microsoft Word 2016
- Database Storage: Google Drive
- Operating System: Windows10

1.5 PROJECT FEATURES: -

The functionalities include, It can send emails, It can read PDF, It can send text on WhatsApp, It can open command prompt, your favorite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc., in a web browser, It can give weather forecast, It can give desktop reminders of your choice. It can have some basic conversation.

Tools and technologies used are VS Code IDE for making this project, and I created all .py files in VS Code. Along with this I used following modules and libraries in my project. pyttsx3, Speech Recognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyautogui, pyQt etc. I have created a live GUI for interacting with the CASPER as it gives a design and interesting look while having the conversation.

1.6 PYTHON: IMPLEMENTATION

Python is a general-purpose, interpreted programming language. Python's design philosophy prioritises code readability and makes heavy use of white space. It offers building blocks that make it possible to programme clearly on both small and large scales, making it the most preferred platform for any implementation of AI. Since Python has numerous properties that make it suited for artificial intelligence, various AI models have been integrated into more recent Python versions. Tensor Flow, Matplotlib, Sci-kit Learn, PyTorch, and many other dynamic and AI-worthy libraries are available in Python to incorporate high-end goals like machine learning, neural networks, artificial intelligence, etc. These modules are used:

The Integrated Modules: -The first module incorporated is "AIML" because the system is built on AIML intelligence. With the aid of this library, one can parse Python entities into an AIML interpreter.
The platform of the system is the sole purpose for which the "argparse" library is employed. We require system tools to change the Linux OS's file structure because the CASPER system is entirely based on Linux. The actual location of the requested file must be specified when a file is called from a remote directory to a location on the system; this is referred to as the "positional argument" in the Python interpreter. Additionally, the system implements common programming libraries like "os" and "time" since AIML requires system values like time and system state to explain real-time entities to the user.

API's Produced: - The CASPER system uses reconstructions of the majority of the AIML categories. To complete the action, a new path or re-directing mechanism is required for the creation of new AIML categories and structures. Therefore, Application Programming Interfaces are developed for Linux systems

employing Python scripting techniques for multitasking to ensure the least amount of latency in data processing.
The Google API, which is in charge of all the speech synthesis algorithms offered by Google, is the fundamental API that has been developed. This API handles the following tasks: text-to-speech conversion, Google Maps integration, and trigger-based browser redirection.

1.7 WINDOWS OS FOR DEVELOPMENT OF CASPER AI

Here are some reasons why Windows can be a suitable choice for certain AI projects:
Compatibility with Software: Windows has a wide range of software tools, libraries, and development environments that are compatible with AI frameworks and libraries. Many popular AI frameworks, such as TensorFlow and PyTorch, provide official support for Windows, making it easier to set up and work with these frameworks on a Windows machine.

User-Friendly Interface: Windows offers a user-friendly and familiar interface, which can be beneficial for developers who are more comfortable working in a Windows environment. The graphical user interface (GUI) provided by Windows simplifies tasks such as installing software, managing dependencies, and configuring hardware, making it accessible to a broader range of developers.
Tool and Library Support: Windows has a robust ecosystem of tools and libraries that support AI development. This includes integrated development environments (IDEs) like PyCharm and Visual Studio, as well as software packages and libraries for data processing, visualization, and model training. Many of these tools have extensive documentation and community support, making it easier to find resources and troubleshoot issues.

Deployment Flexibility: While AI models are typically trained on powerful hardware like GPUs or cloud infrastructure, the deployment environment may vary. Windows provides flexibility in terms of deploying AI applications on both local machines and cloud platforms. It supports popular cloud platforms like Microsoft Azure, allowing seamless integration and deployment of AI models.

Enterprise Integration: Windows is widely used in enterprise environments, and AI projects developed on Windows can benefit from existing enterprise infrastructure and systems. Integration with enterprise tools, such as Active Directory, enterprise security protocols, and collaboration platforms, can be smoother when using Windows as the operating system.

It's important to note that the choice of operating system ultimately depends on the specific requirements and constraints of the AI project. Developers should consider factors such as the availability of necessary tools, hardware compatibility, team expertise, and the target deployment environment when deciding on the most suitable operating system for their AI project.

CHAPTER-2

2. Data Flow

DATAFLOW:

The data flow for CASPER are as follows: -

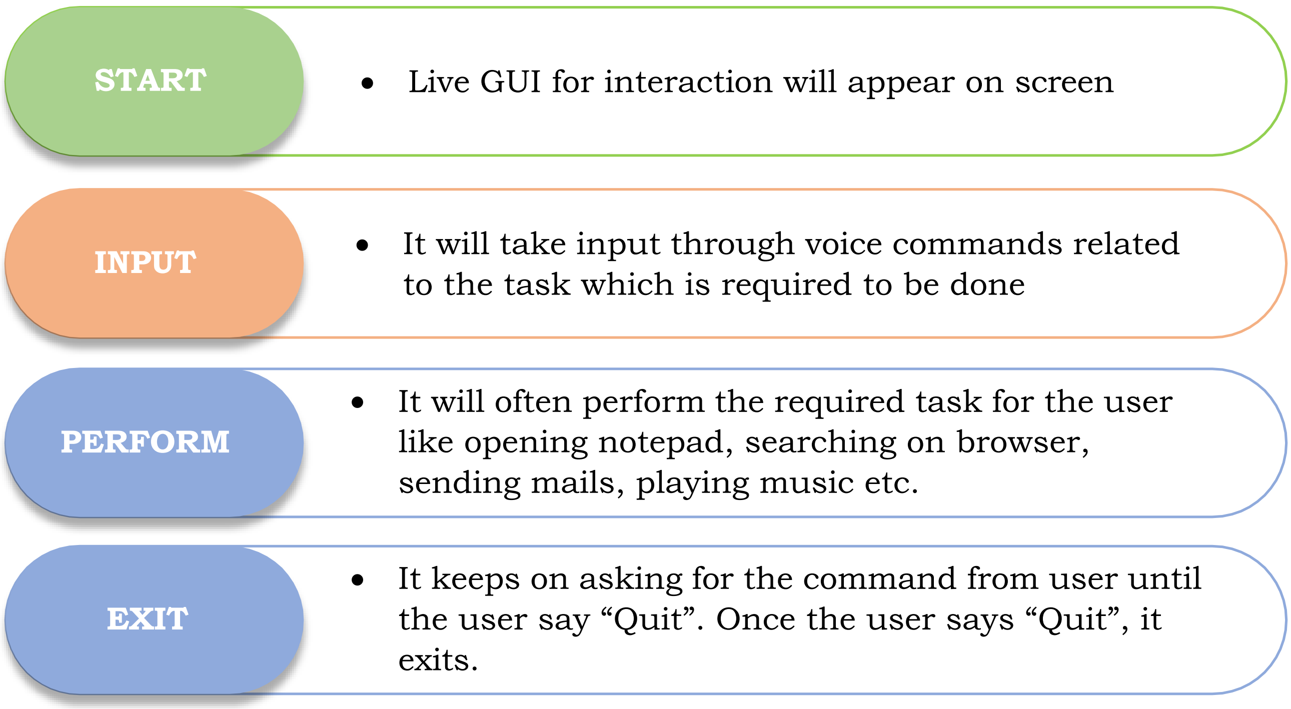

START
• Live GUI for interaction will appear on screen
INPUT
• It will take input through voice commands related to the task which is required to be done
PERFORM
• It will often perform the required task for the user like opening notepad, searching on browser, sending mails, playing music etc.
EXIT
• It keeps on asking for the command from user until the user say "Quit". Once the user says "Quit", it exits.

Figure Data flow for CASPER: -

The system is designed using the concept of Artificial Intelligence and with the help of necessary packages of Python. Python provides many libraries and packages to perform the tasks, for example pyPDF2 can be used to read PDF. The details of these packages are mentioned in Chapter 3 of this report.

The data in this project is nothing but user input, whatever the user says, the assistant performs the task accordingly. The user input is nothing specific but the list of tasks which a user wants to get performed in human language i.e., English.

2.1 **Present System:**

We are familiar with many existing voice assistants like Alexa, Siri, Google Assistant, Cortana which uses concept of language processing, and voice recognition. They listen the command given by the user as per their requirements and performs that specific function in a very efficient and effective manner.

As these voice assistants are using Artificial Intelligence hence the result that they are providing are highly accurate and efficient. These assistants can help to reduce human effort and consumes time while performing any task, they removed the concept of typing completely and behave as another individual to whom we are talking and asking to perform task. These assistants are no less than a human assistant but we can say that they are more effective and efficient to perform any task. The algorithm used to make these assistant focuses on the time complexities and reduces time.

But for using these assistants one should have an account (like Google account for Google assistant, Microsoft account for Cortana) and can use it with internet connection only because these assistants are going to work with internet connectivity. They are integrated with many devices like, phones, laptops, and speakers etc.

2.2 PROPOSED SYSTEM: -

It was an interesting task to make my own assistant. It became easier to send emails without typing any word, searching on Google without opening the browser, and performing many other daily tasks like playing music, opening your favorite IDE with the help of a single voice command. CASPER is different from other traditional voice assistants in terms that it is specific to desktop and user does not need to make account to use this, it does not require any internet connection while getting the instructions to perform any specific task.

The IDE used in this project is VS Code. All the python files were created in VS Code and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e., pyttsx3, Speech Recognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyjokes, pyPDF2, pyautogui, pyQt etc. I have created a live GUI for interacting with the CASPER as it gives a design and interesting look while having the conversation.

With the advancement CASPER can perform any task with same effectiveness or can say more effectively than us. By making this project, I realized that the concept of AI in every field is decreasing human effort and saving time. Functionalities of this project include, It can send emails, It can read PDF, It can send text on WhatsApp, It can open command prompt, your favorite IDE, notepad etc., It can play music, It can do Wikipedia searches for you, It can open websites like Google, YouTube, etc., in a web browser, It can give weather forecast, It can give desktop reminders of your choice. It can have some basic conversation.

CHAPTER-3

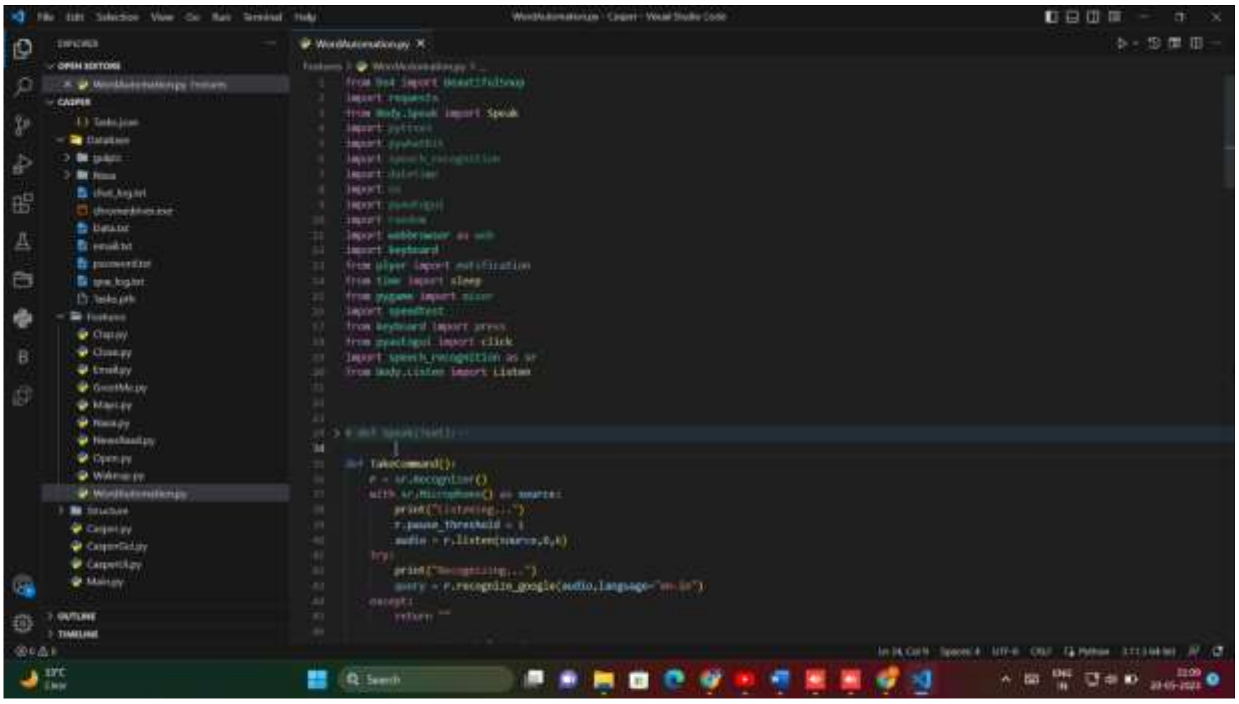
System Requirement

3.1 Introduction: -

The IDE used in this project is VS Code. All the python files were created in VS Code and all the necessary packages were easily installable in this IDE. For this project following modules and libraries were used i.e., pyttsx3, Speech Recognition, Datetime, Wikipedia, Smtplib, pywhatkit, pyautogui, pyQt etc. I have created a live GUI for interacting with the CASPER as it gives a design and interesting look while having the conversation.

3.2 VS CODE: -

It is an IDE i.e., Integrated Development Environment which has many features like it supports scientific tools (like matplotlib, numpy, scipy etc) web frameworks (example Django, web2py and Flask) refactoring in Python, integrated python debugger, code completion, code and project navigation etc.

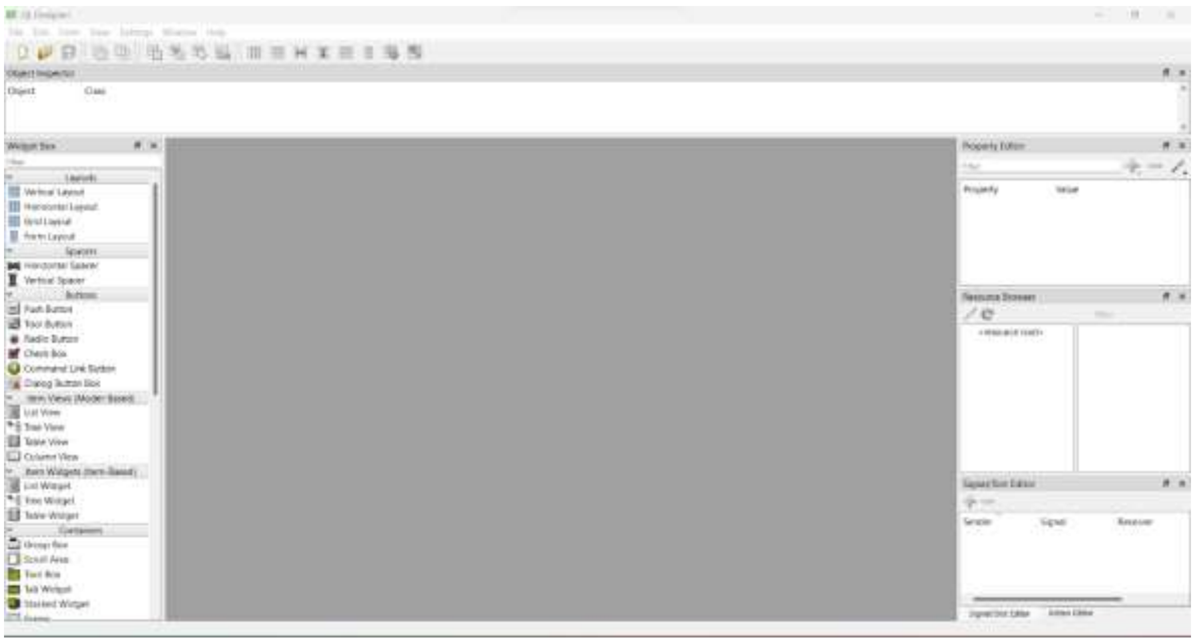


```
from bs4 import BeautifulSoup
import requests
from Body.Speak import Speak
import pyttsx3
import pywhatkit
import speech_recognition
import datetime
import os
import pyautogui
import random
import webbrowser as web
import keyboard
from plyer import notification
from time import sleep
from pygame import mixer
import speedtest
from keyboard import press
from pyautogui import click
import speech_recognition as sr
from Body.Listen import Listen



> # def Speak(text):...

def TakeCommand():
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Listening...")
        r.pause_threshold = 1
        audio = r.listen(source,0,4)
    try:
        print("Recognizing...")
        query = r.recognize_google(audio,language="en-in")
    except:
        return ""
```

3.3 PYQT5 FOR LIVE GUI: -

PyQt5 is the most important python binding. It contains set of GUI widgets. PyQt5 has some important python modules like QTWidgets, QtCore, QtGui, and QtDesigner etc.

PyQt5 is the most important python binding. It contains set of GUI widgets. PyQt5 has some important python modules like QTWidgets, QtCore, QtGui, and QtDesigner etc.



3.4 PYTHON LIBRARIES: -

In our CASPERAI project, we have utilized various Python libraries to enhance the functionality and efficiency of our system. Here are some of the key libraries we have employed:

In order to scrape and parse HTML/XML files from websites, we used BeautifulSoup4. We were able to collect pertinent data for our AI algorithms by using this library to extract specific data from web pages.

Our CASPER AI project's user-friendly command-line interface is thanks in large part to Click. By using Click, we have made it easier for users to engage with our system and enter commands and parameters.

DateTime: DateTime has been essential to our AI project's ability to manage and manipulate dates and timings. It has given us the fundamental tools we need to manage temporal data, like the ability to format dates, calculate time differences, and convert between different time zones.

Flask: We used Flask, a compact web framework, to create web applications for our Caspe AI project. It has made it easier to create RESTful APIs, enabling simple connection with our AI algorithms and seamless interaction with other systems.

Googletrans: To make use of the Google Translate API in our app, we included the Googletrans library. We have been able to translate textual data thanks to this package, which is extremely helpful for multilingual support and language processing activities.

Our preferred HTTP client library, httpx, supports asynchronous requests and has served as such. We have used it to efficiently and concurrently process HTTP queries, manage answers, and web interactions within our AI system.

nltk: The Natural Language Toolkit (nltk) has been invaluable for text processing and analysis in our CASPER AI project. We have utilized its comprehensive set of tools and algorithms for tasks such as tokenization, stemming, part-of-speech tagging, and sentiment analysis.

numpy: As a fundamental package for scientific computing, numpy has been extensively utilized in our project. It has provided support for efficient array operations, numerical computations, and linear algebra, enabling us to process and manipulate data efficiently within our AI algorithms.

openai: We have leveraged the openai library to interface with OpenAI models and services. It has allowed us to integrate state-of-the-art AI models into our system, enabling advanced capabilities such as natural language understanding, text generation, and more.

Selenium is a web testing platform that has been crucial in helping our CASPER AI project automate browser actions. We can now run automated tests, interact with web apps, and get information from dynamic web sites.

These are but a handful of the several Python libraries that we have used in the CASPER AI project. Each library has aided in the growth

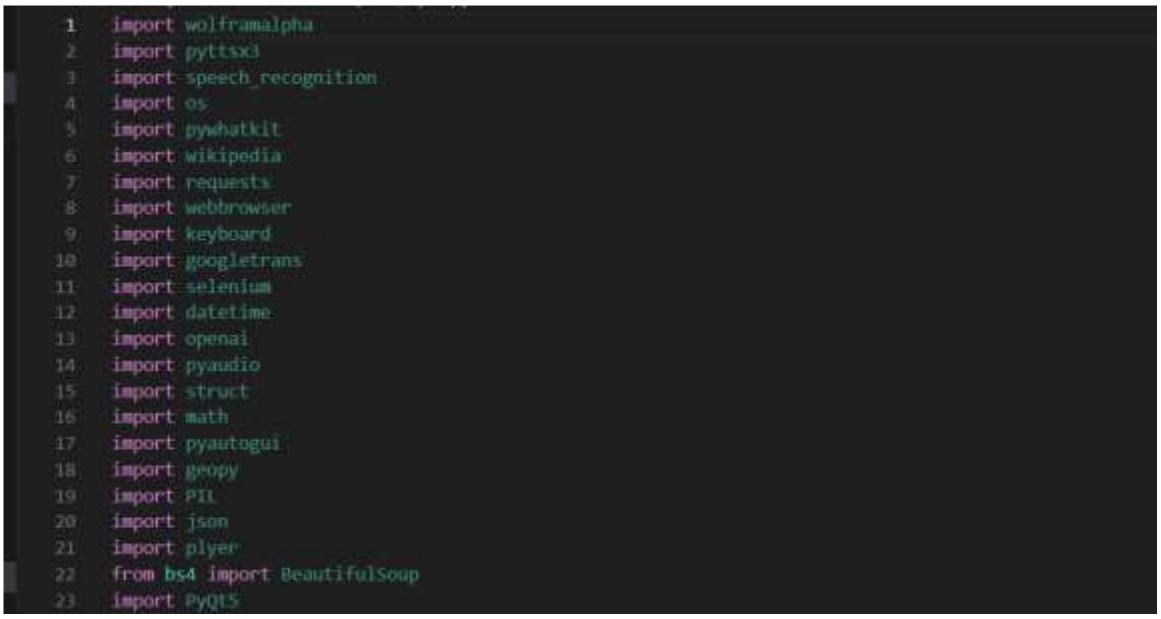
and usefulness of our system and helped us successfully and efficiently accomplish our objectives.

3.5 FUNCTIONS

Listen(): This function receives a command through the user's microphone and outputs it as a string.

GreetMe(): This function sends a good morning, good afternoon, or good evening greeting to the user based on the time.

TaskExecution(): This function provides every definition for a task's execution, including sendEmail(), news(), and numerous if conditions, such as "open google," "open notepad," "Wikipedia Search," "play music," "WeTube Search," and "open command prompt," among others.



```python
import wolframalpha
import pyttsx3
import speech_recognition
import os
import pywhatkit
import wikipedia
import requests
import webbrowser
import keyboard
import googletrans
import selenium
import datetime
import openai
import pyaudio
import struct
import math
import pyautogui
import geopy
import PIL
import json
import plyer
from bs4 import BeautifulSoup
import PyQt5
```

CHAPTER-4

4.1 Implementation/Working of Project

With the aid of a single voice command, CASPER, a desktop assistant, may carry out a variety of common desktop operations, including playing music, launching your preferred web browser, or activating installed software. CASPER differs from other conventional voice assistants in that it is designed specifically for desktop usage, users do not need to create accounts in order to use it, and it does not need an internet connection in order to receive instructions on how to carry out any given activity.

4.2 REAL LIFE APPLICATION:

Time-saving: CASPER is a desktop voice assistant that responds to voice requests. It can perform speech searches, manage voice-activated devices, and allow us to complete a number of tasks.

Conversational interaction: It makes it simpler to finish any activity because it does so automatically by utilising the key modules or libraries of Python. Because of the conversational connection between providing input and receiving the desired output in the form of a task completed, any user who gives it instructions for any task will feel like giving it to a human helper.

Desktop assistant's reactive nature: The desktop assistant responds in human-understandable English because it is very adept at understanding human language and the context that the user provides. As a result, the user reacts in a wise and educated manner.

Its capacity for multitasking is its primary application. Until the user "quits" it, it may continuously request commands one after another.

Without the requirement for a trigger phase, it asks the user for an instruction, listens to their response, and only then begins to carry out the task.

4.3 DATA IMPLEMENTATION AND PROGRAM EXECUTION

As the first step, install all the necessary packages and libraries. The command used to install the libraries is “pip install” and then import it. The necessary

4.4 LIBRARIES AND PACKAGES

beautifulsoup4 (4.11.2): Python library for web scraping and parsing HTML/XML documents.

click (8.1.3): Command-line interface creation kit for Python.

DateTime (5.1): Library for manipulating dates and times in Python.

Flask (2.2.3): Lightweight web framework for building web applications in Python.

Googletrans (3.1.0a0): Free and unlimited Python library for Google Translate API.

httpx (0.13.3): HTTP client library for Python with support for asynchronous requests.

keyboard (0.13.5): Python library for detecting and controlling keyboard events.

multidict (6.0.4): Collection of multi-valued dictionaries for Python.

nltk (3.8.1): Natural Language Toolkit for Python, providing tools for text processing and analysis.

numpy (1.24.2): Fundamental package for scientific computing with Python, providing support for arrays and numerical operations.

openai (0.27.2): Python library for interfacing with OpenAI models and services.

pip (23.0.1): Package installer for Python, used for installing and managing Python packages.

plyer (2.1.0): Cross-platform library for accessing features of the underlying operating system.

PyAudio (0.2.13): Python bindings for PortAudio, allowing audio input/output in Python.

PyAutoGUI (0.9.53): Python library for automating GUI interactions by simulating mouse and keyboard input.

Pycparser (2.21): Complete parser of the C language, written in Python.

Pygame (2.3.0): Cross-platform library for game development in Python.

PyGetWindow (0.0.9): Library for manipulating windows and getting window information in Python.

pynput (1.7.6): Library for controlling and monitoring input devices such as keyboard and mouse in Python.

pyOpenSSL (23.0.0): Python library for secure communication over SSL/TLS protocols.

pyperclip (1.8.2): Cross-platform clipboard module for Python.

pyphen (0.14.0): Library for hyphenating words in Python.

pypiwin32 (223): Python wrapper for the Win32 API, providing access to many Windows-specific functions.

PyQt5 (5.15.9): Python bindings for the Qt framework, enabling the creation of desktop applications.

pyqt5-plugins (5.15.9.2.3): Additional plugins for PyQt5, providing extended functionality.

PyQt5-Qt5 (5.15.2): Core Qt module for PyQt5.

PyQt5-sip (12.12.0): SIP bindings for PyQt5, used for generating Python bindings for C++ libraries.

pyqt5-tools (5.15.9.3.3): Set of development tools and utilities for PyQt5.

pyttsx3 (2.90): Text-to-speech conversion library in Python.

pytweening (1.0.4): Easing equations library for smooth interpolation in Python.

pywhatkit (5.4): Library for performing various tasks using WhatsApp, such as sending messages and performing searches.

pywin32 (305): Python extensions for Windows, providing access to many Windows-specific APIs.

qt5-applications (5.15.2.2.3): Collection of Qt5 applications and tools.

qt5-tools (5.15.2.1.3): Collection of development tools and utilities for Qt5.

selenium (4.1.3): Web testing framework for automating browser actions in Python.

Speech Recognition (3.10.0): Library for performing speech recognition in Python.

speedtest-cli (2.1.3): Command-line interface for testing internet bandwidth using Speedtest.net.

torch (2.0.0): PyTorch, a machine learning framework for Python.

urllib3 (1.26.15): Library for making HTTP requests in Python.

urllib3-secure-extra (0.1.0): Extra security features for urllib3, including SSL/TLS verification.

Wikipedia (1.4.0): Python wrapper for the Wikipedia API, providing access to Wikipedia content and data.

Wolframalpha (5.0.0): Python wrapper for the Wolfram Alpha API, enabling computational knowledge queries.System: It allows operating on the interpreter as it provides access to the variables and functions that usually interact strongly with the interpreter.

FUNCTIONS

Listen(): The function is used to take the command as input through microphone of user and returns the output as string.

GreetMe(): This function greets the user according to the time like Good Morning, Good Afternoon and Good Evening.

taskExecution(): This is the function which contains all the necessary task execution definition like sendEmail(), news() and many conditions in if condition like “open google”, “open notepad”, “Wikipedia Search” ,”play music”, “Youtube Search” and “open command prompt” etc.

CHAPTER-5

5.1 RESULT OR OUTPUT

Casper AI is a tool made with python that lets people do different things by just speaking to it. It talks to different apps and websites using speech recognition, understanding natural language, and scraping information from the internet. Casper AI can do certain tasks.

You can use Casper AI to find a place on Google Maps. Just tell Casper AI where you want to go, like "show me the Eiffel Tower on Google Maps". Casper AI will open Google Maps in your web browser and show you the Eiffel Tower on the map.

Casper AI can do math calculations by listening to what the user says, like "what's the square root of 25". It uses a special tool in the Python programming language to figure out the answer. Then, it talks back to the user using a system that turns text into spoken words.

Casper AI is a program that can open YouTube channels and music apps. It can understand voice commands from the user, like "play jazz music on Spotify". It uses a module in Python called web browser to open Spotify website or app and search for jazz music. Casper AI is able to open YouTube channels when given a voice command like "show me videos of pandas on YouTube".

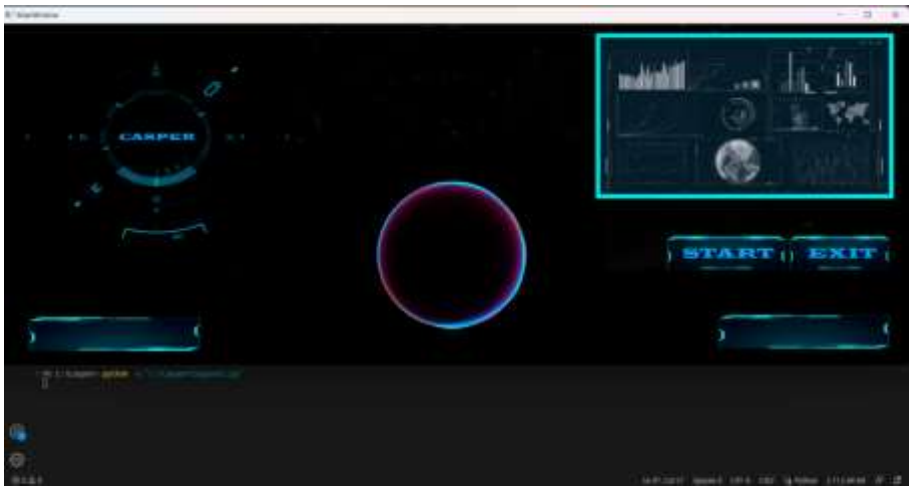
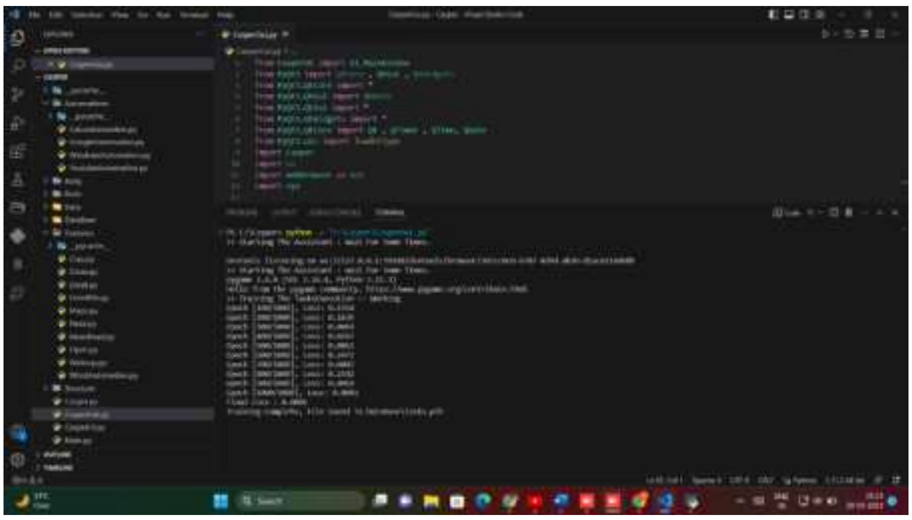
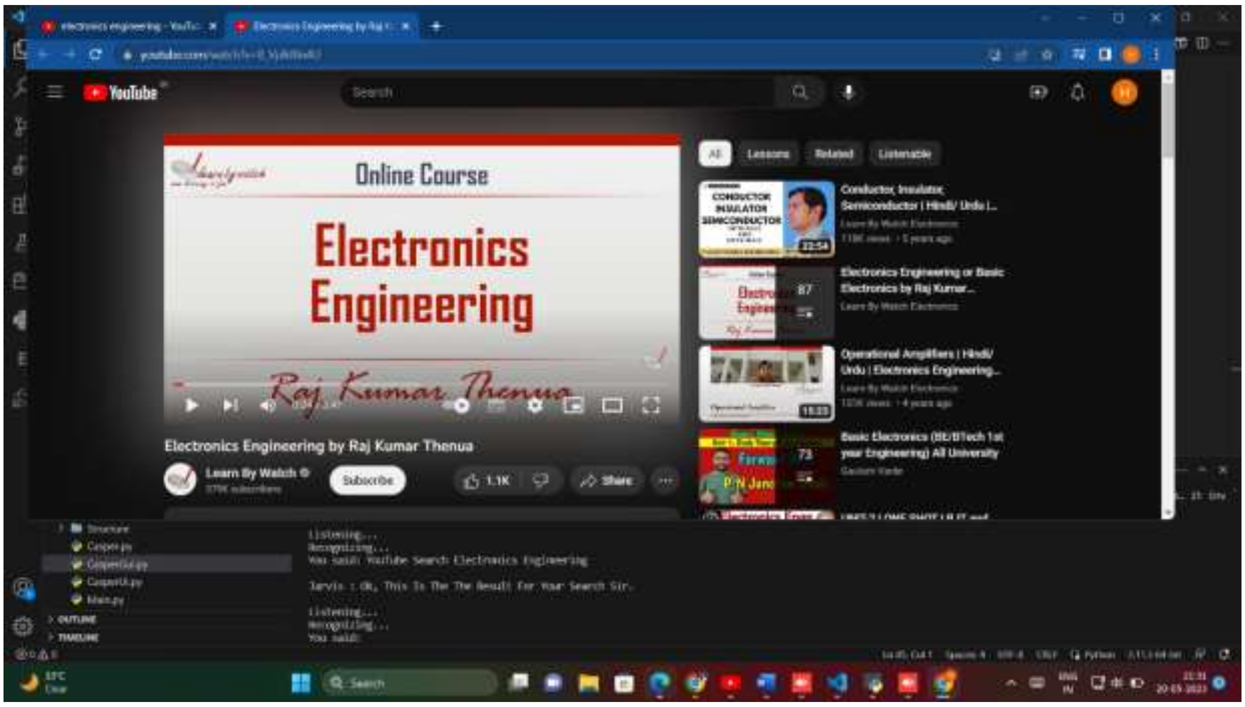


Figure 5.1.1 Live GUI of CASPER



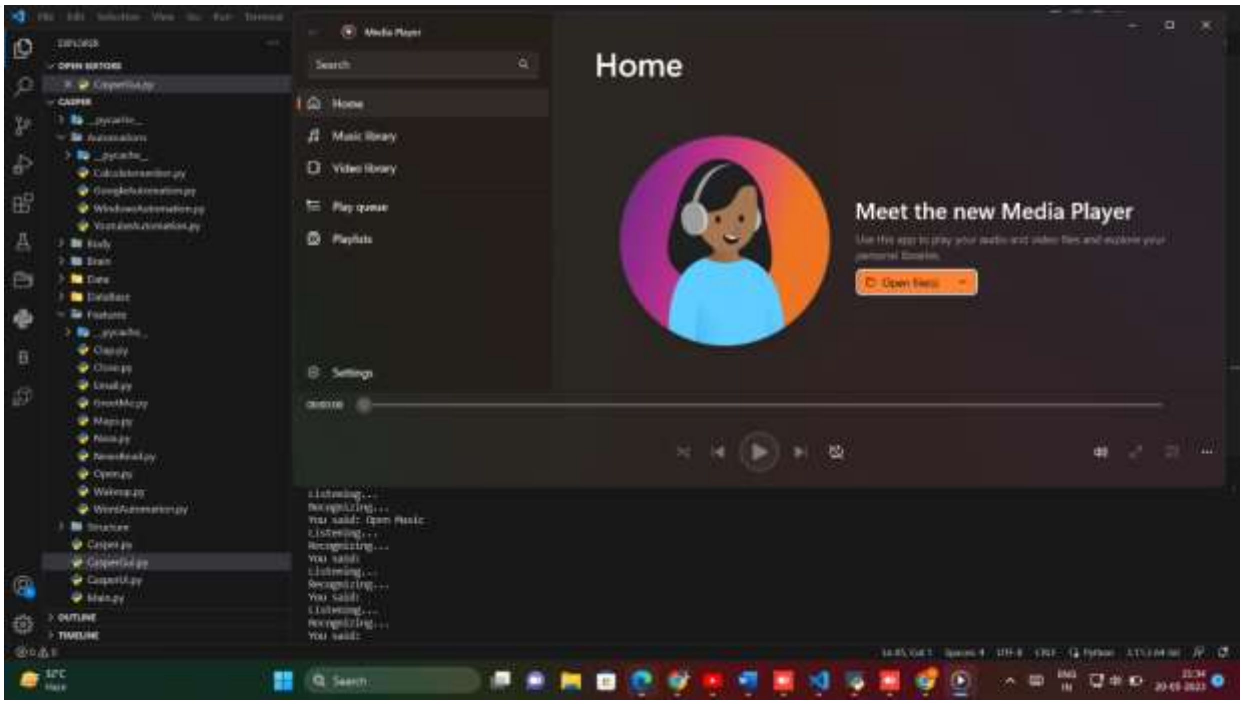
```
[illegible]
```

Figure 5.1.2 Live GUI Training



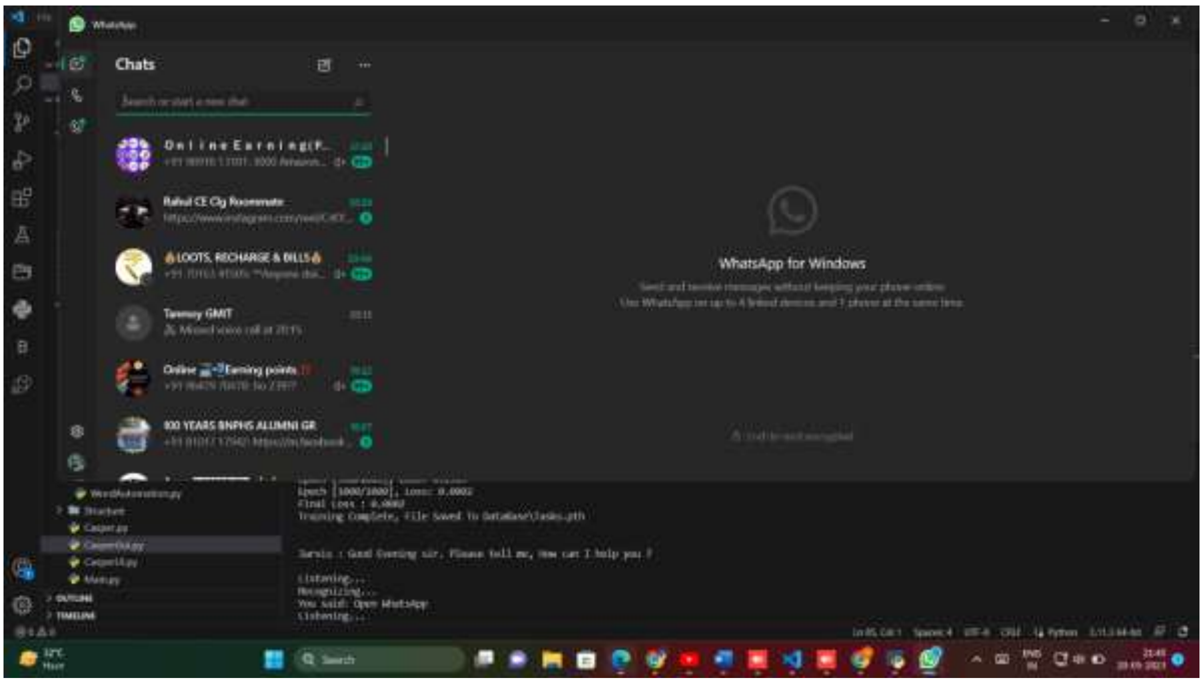
```
Listening...
Recognizing...
You said: YouTube Search Electronics Engineering

Jarvis : Ok, This Is The The Result For Your Search Sir.

Listening...
Recognizing...
You said:
```

Figure 5.1.3 Output for YouTube search



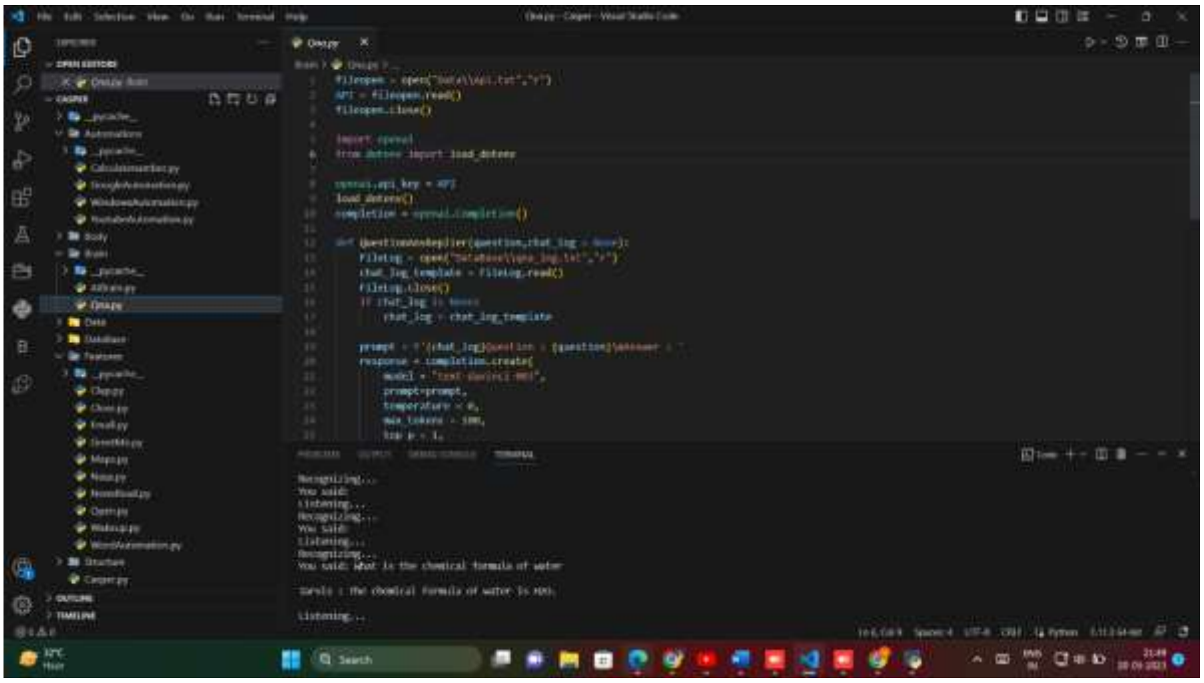
```
Listening...
Recognizing...
You said: Open Music
Listening...
Recognizing...
You said:
Listening...
Recognizing...
You said:
Listening...
Recognizing...
You said:
```

Figure 5.1.4 Output to play music



```
Epoch [1000/1000], Loss: 0.0002
Final Loss : 0.0002
Training Complete, File Saved To DataBase\Tasks.pth

Jarvis : Good Evening sir, Please tell me, How can I help you ?

Listening...
Recognizing...
You said: Open WhatsApp
Listening...
```

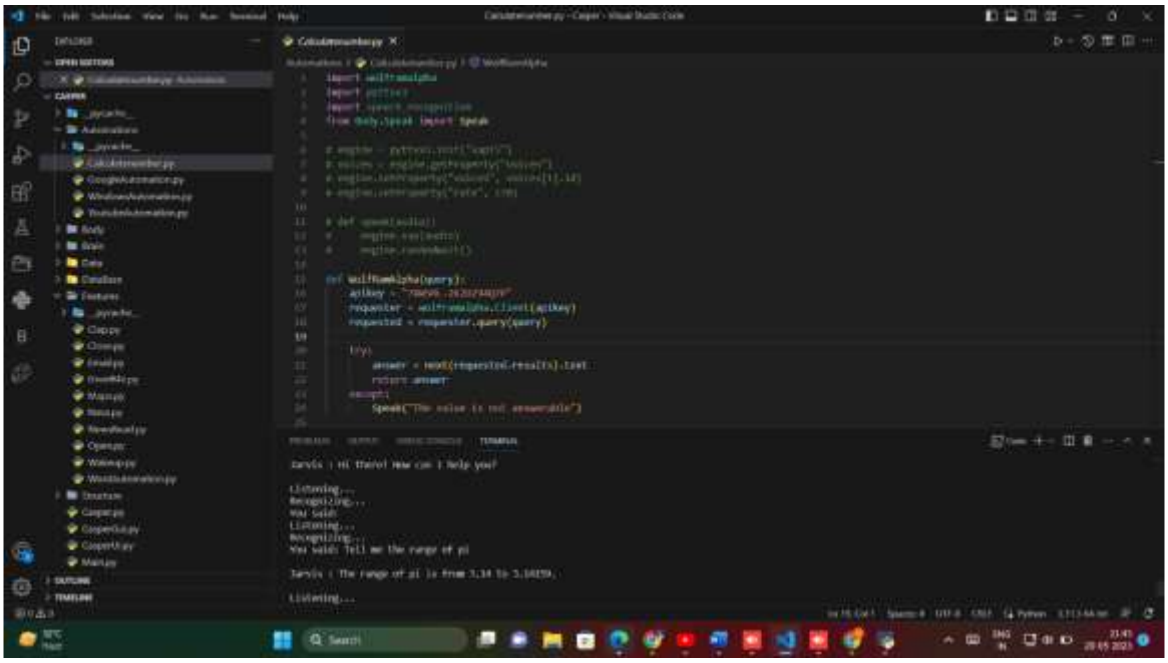
Figure 5.1.5 Output to Whataspp



```python
fileopen = open("Data\\Api.txt","r")
API = fileopen.read()
fileopen.close()

import openai
from dotenv import load_dotenv

openai.api_key = API
load_dotenv()
completion = openai.Completion()

def QuestionsAnswerReplier(question,chat_log = None):
    FileLog = open("DataBase\\qna_log.txt","r")
    chat_log_template = FileLog.read()
    FileLog.close()
    if chat_log is None:
        chat_log = chat_log_template

    prompt = f'{chat_log}Question : {question}\nAnswer : '
    response = completion.create(
        model = "text-davinci-003",
        prompt=prompt,
        temperature = 0,
        max_tokens = 100,
        top_p = 1,
```

```
Recognizing...
You said:
Listening...
Recognizing...
You said:
Listening...
Recognizing...
You said: What is the chemical formula of water

Jarvis : The chemical formula of water is H2O.

Listening...
```

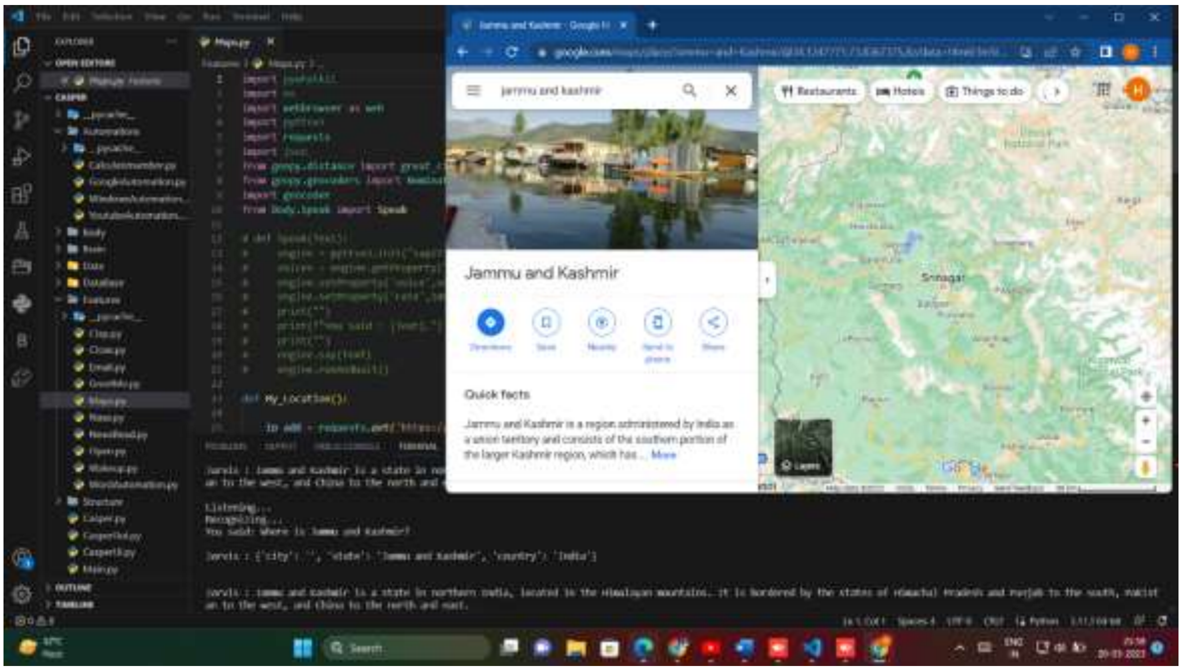
Figure 5.1.6 Q & A



```python
import wolframalpha
import pyttsx3
import speech_recognition
from Body.Speak import Speak

# engine = pyttsx3.init("sapi5")
# voices = engine.getProperty("voices")
# engine.setProperty("voices", voices[1].id)
# engine.setProperty("rate", 170)

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

def WolfRamAlpha(query):
    apikey = "[illegible]"
    requester = wolframalpha.Client(apikey)
    requested = requester.query(query)

    try:
        answer = next(requested.results).text
        return answer
    except:
        Speak("The value is not answerable")
```

```
Jarvis : Hi there! How can I help you?

Listening...
Recognizing...
You said:
Listening...
Recognizing...
You said: Tell me the range of pi

Jarvis : The range of pi is from 3.14 to 3.14159.

Listening...
```

Figure 5.1.7 Calculations



```python
import pywhatkit
import os
import webbrowser as web
import pyttsx3
import requests
import json
from geopy.distance import great_c
from geopy.geocoders import Nominati
import geocoder
from Body.Speak import Speak

# def Speak(Text):
#     engine = pyttsx3.init("sapi
#     voices = engine.getProperty(
#     engine.setProperty('voice',
#     engine.setProperty('rate',
#     print("")
#     print(f"You said : {Text}.")
#     print("")
#     engine.say(Text)
#     engine.runAndWait()

def My_Location():

    ip_add = requests.get('https://
```

```
Jarvis : Jammu and Kashmir is a state in no
an to the west, and China to the north and 

Listening...
Recognizing...
You said: where is Jammu and Kashmir?

Jarvis : {'city': '', 'state': 'Jammu and Kashmir', 'country': 'India'}


Jarvis : Jammu and Kashmir is a state in northern India, located in the Himalayan mountains. It is bordered by the states of Himachal Pradesh and Punjab to the south, Pakist
an to the west, and China to the north and east.
```

Jammu and Kashmir

Quick facts

Jammu and Kashmir is a region administered by India as a union territory and consists of the southern portion of the larger Kashmir region, which has ... More

Figure 5.1.8 Maps

CHAPTER-6

6.1 Functionality

Here, we test the system's functioning to see if it accomplishes the job for which it was designed. Each function was examined and tested to determine its functionality; if it can successfully do the needed task, the system passes that particular functionality test. For instance, to see if CASPER can search on Google or not, as we can see in figure 7.1, the user stated "Open Google", then CASPER asked, "What should I search on Google?" The user then responded, "What is Python", and CASPER opened Google and looked for the necessary input number.

6.2 USABILITY

The ease of the programme and how user-friendly it is for the user to use, as well as how it responds to each question that the user asks, are used to assess a system's usability.

Any task can be completed more quickly because Python's core modules and libraries are used automatically and in a conversational manner. Because of the conversational contact required to provide input and obtain the intended output of a task, any user who gives it a task feels as though they are delivering it to a human helper.

The desktop assistant is reactive, meaning it understands human language very well, as well as the context that the user provides, and responds in a manner that is humanly intelligible, such as English. As a result, the user reacts in a wise and educated manner.

Its ability to multitask may be the key application. It may keep asking the user for instructions again until they "quit" it. Without a trigger phase, it asks the user for an instruction, listens to their response, and only then starts the task.

6.3 SECURITY

The security testing mainly focuses on vulnerabilities and risks. As CASPER is a local desktop application, hence there is no risk of data breaching through remote access. The software is dedicated to a specific system so when the user logs in, it will be activated.

6.4 STABILITY

Stability of a system depends upon the output of the system, if the output is bounded and specific to the bounded input then the system is said to be stable. If the system works on all the poles of functionality, then it is stable.

CHAPTER 7

7.3 CONCLUSIONS AND FUTURE WORK

The ease of the programme and how user-friendly it is for the user to use, as well as how it responds to each question that the user asks, are used to assess a system's usability.
Any task can be completed more quickly because Python's core modules and libraries are used automatically and in a conversational manner. Because of the conversational contact required to provide input and obtain the intended output of a task, any user who gives it a task feels as though they are delivering it to a human helper.

The desktop assistant is reactive, meaning it understands human language very well, as well as the context that the user provides, and responds in a manner that is humanly intelligible, such as English. As a result, the user reacts in a wise and educated manner.

Its ability to multitask may be the key application. It may keep asking the user for instructions again until they "quit" it. Without a trigger phase, it asks the user for an instruction, listens to their response, and only then starts the task.
Here, we test the system's functioning to see if it accomplishes the job for which it was designed. Each function was examined and tested to determine its functionality; if it can successfully do the needed task, the system passes that particular functionality test. For instance, to see if CASPER can search on Google or not, the user stated "Open Google", then CASPER asked, "What should I search on Google?" The user then responded, "What is Python", and CASPER opened Google and looked for the necessary input number.

7.2 LIMITATIONS

Security is somewhere an issue, there is no voice command encryption in this project.

Background voice can interfere misinterpretation because of accents and may cause inaccurate results.

CASPER cannot be called externally anytime like other traditional assistants like Google Assistant can be called just by saying, “Ok Google!”

7.3 SCOPE FOR FUTURE WORK

Make CASPER to learn more on its own and develop a new skill in it. CASPER android app can also be developed.

Make more CASPER voice terminals.

Voice commands can be encrypted to maintain security.

CHAPTER-8

8.1 REFERENCES

[1] M. Dahmane and J. Meunier, "Emotion recognition using dynamic grid- based HoG features," *Face and Gesture 2011*, Santa Barbara, CA, 2011, pp.884-888.

[2] M. S. Hossain and G. Muhammad, "An Emotion Recognition System for Mobile Applications," in *IEEE Access*, vol. 5, pp.2281-2287, 2017.

[3] A. V. Iyer, V. Pasad, S. R. Sankhe and K. Prajapati, "Emotion based mood enhancing music recommendation," *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, 2017, pp. 1573-1577.

[4] T. Moriyama, K. Abdelaziz and N. Shimomura, "Face analysis of aggressive moods in automobile driving using mutual subspace method," *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*