1. INTRODUCTION

1.1) PROBLEM DEFINITION

Determine what tasks you want your virtual assistant to perform. This could include things like answering questions, providing weather updates, managing your calendar, sending emails, etc.

1.2) FEASIBILITY STUDY

Feasibility study can help you determine whether or not you should proceed with. your project. It is essential to evaluate cost and benefit of the proposed system.

Three key considerations involved in the feasibility analysis are:

- **Economical feasibility**
- **Technical feasibility**
- **Social feasibility**

1.2.1)ECONOMICAL FEASIBILITY:

Here, we find the total cost and benefit of the proposed system over current system. For this project, the main cost is documentation cost. User also would have to pay for microphones and speakers. Again, they are cheap and available.

1.2.2) TECHNICAL FEASIBILITY:

It includes finding out technologies for the project, both hardware and software. For virtual assistant, user must have microphone to convey their message and a speaker to listen what system speaks. These are very cheap now a days and everyone generally possess them.

Besides, system needs internet connection. It is also not an issue in this era where almost every home or office has Wi-Fi.

1.2.3)SOCIAL FEASIBILITY:

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity..

The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system..

1.3) SCOPE OF THE PROJECT

The scope of a virtual assistant project encompasses a broad array of considerations essential for its successful implementation. At its core lies the delineation of functionality, dictating the tasks the virtual assistant will undertake, whether it be answering queries, scheduling appointments, or executing commands. Platform selection is pivotal, determining where the assistant will operate, be it on web browsers, mobile apps, or messaging platforms. Integration is another critical aspect, necessitating seamless connectivity with various systems and databases to fulfill its designated functions effectively. Natural Language Processing (NLP) capabilities are paramount for understanding user queries accurately, encompassing intent recognition, entity extraction, and sentiment analysis. Designing an intuitive User Interface (UI) and optimizing User Experience (UX) are imperative for fostering user engagement and satisfaction. Personalization features, such as tailored responses based on user preferences, enhance the assistant's utility.

Security and privacy considerations are non-negotiable, ensuring the safeguarding of user data and compliance with regulatory frameworks. Scalability planning is crucial for accommodating future growth and evolving requirements. Rigorous testing and quality assurance procedures are indispensable to validate the assistant's functionality and reliability across diverse scenarios. Deployment protocols and maintenance frameworks ensure smooth operation post-launch, with continuous feedback mechanisms driving iterative improvements. Training materials and comprehensive documentation facilitate user adoption and support ongoing usage. Ultimately, a well-defined scope provides a roadmap for stakeholders, fostering alignment and guiding the virtual assistant project toward its intended objectives.

2. SYSTEM REQUIREMENT

2.1 HARDWARE SPECIFICATION

Processor	:	AMD PRO A4-4350B R4, 5 COMPUTE CORES 2C+3G 2.50GHz
RAM	:	4 GB
Hard disk	:	512GB
CPU Model	:	AMD PRO A4-4350B R4, 5
Screen Size	:	15.6 inches

2.2 SOFTWARE SPECIFICATION

Operating System	:	Windows 10
Front End	:	tkinter
Back End	:	Python
Simulation Tools	:	Visual Studio Code
Python	:	Latest Version
Packages	:	pillow, lxml, requests-html, pyttsx3 SpeechRecognition, PyAudio

3. SYSTEM DESCRIPTION

3.1 PROJECT DESCRIPTION

MODULES

- Pyttsx3
- Sapi5
- Speech recognition
- Pyaudio
- Wikipedia
- Webbrowser

Pyttsx3 (Python Text to Speech)

A python library that will help us to convert text to speech. It is a cross-platform Python wrapper for text-to-speech synthesis. It is a Python package supporting common text-to-speech engines on MacOS X, Windows, and Linux. It works for both Python2.x and 3.x versions. Its main advantage is that it works offline

Sapi5 (Speech Application Programming Interface)

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 applications. 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 the Windows OS itself.

Applications that use SAPI include Microsoft Office, Microsoft Agent and Microsoft Speech Server. Many versions (although not all) of the speech recognition and synthesis engines are also freely redistributable. SAPI 5 however was a completely new interface, released in 2000. Since, then several sub-versions of this API have been released.

Speech recognition

Speech recognition is the process of converting spoken words to text. Python supports many speech recognition engines and APIs, including Google Speech Engine, Google Cloud Speech API, Microsoft Bing Voice Recognition and IBM Speech to Text.Speech Recognition is an important feature in several applications used such as home automation, artificial intelligence, etc.

Recognizing speech needs audio input, and Speech Recognition makes it really simple to retrieve this input. This is a library for performing speech recognition, with support for several engines and APIs, online and offline.

Pyaudio

To access your microphone with Speech Recognizer, you'll have to install the PyAudio package. PyAudio provides Python bindings for Port Audio, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio on a varity of platforms.

Wikipedia

Wikipedia is a Python library that makes it easy to access and parse data from Wikipedia. It gets article summaries, get data like links and images from a page, and more. This module provides developers code-level access to the entire Wikipedia reference.

Webbrowser

The webbrowser module provides a high-level interface to allow displaying Web- based documents to users. Under most circumstances, simply calling the open() function from this module will do the right thing.

3.2. SOFTWARE DESCRIPTION

This whole project is created by the backend language name PYTHON, The frontend used are TIKINTER..

TIKINTER :

Tkinter is a Python library widely used for creating graphical user interfaces (GUIs) in desktop applications. Known for its simplicity and ease of use, Tkinter provides developers with a straightforward means of designing interactive interfaces for their Python programs. At its core, Tkinter is based on the Tk GUI toolkit, originally developed as part of the Tcl scripting language. However, Tkinter seamlessly integrates with Python, offering developers a familiar syntax and environment for GUI development. With Tkinter, developers can create various GUI elements such as windows, buttons, menus, entry fields, and more, using a combination of pre-built widgets and customizable options. Tkinter follows an event-driven programming model, where user actions like button clicks or keyboard inputs trigger corresponding event handlers. This allows for dynamic and responsive user interfaces. Despite its simplicity, Tkinter is versatile and supports a wide range of functionalities, making it suitable for both beginner-friendly projects and more complex applications. Additionally, Tkinter applications are cross-platform, meaning they can run on different operating systems without requiring significant modifications, further enhancing its appeal to Python developers. Overall, Tkinter remains a popular choice for GUI development in Python due to its accessibility, flexibility, and robust feature set.

MACHINE LEARNING :

Machine learning is a subfield of artificial intelligence (AI) that focuses on the development of algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed. At its core, machine learning leverages statistical techniques to identify patterns and extract insights from large datasets, allowing computers to improve their performance over time through experience. One of the key principles of machine learning is the ability to generalize from past observations to new, unseen data, enabling systems to make accurate predictions or decisions in real-world scenarios. Machine learning algorithms can be broadly categorized into supervised learning, where models are trained on labeled data, unsupervised learning, which involves finding patterns in unlabeled data, and reinforcement learning, where agents learn to interact with an environment through trial and error. Machine learning has a wide range of applications across various domains, including image and speech recognition, natural language processing, recommendation systems, financial forecasting, healthcare diagnostics, and autonomous vehicles. However, the adoption of machine learning also poses challenges, such as data quality issues, algorithmic bias, interpretability concerns, and ethical considerations. As machine learning continues to advance, with innovations such as deep learning and reinforcement learning, it holds the promise of revolutionizing industries, improving decision-making processes, and enhancing human capabilities in unprecedented ways.

PYTHON:

Python is a dynamically typed, interpreted programming language with a strong emphasis on simplicity and readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming styles, allowing developers to choose the approach that best suits their needs. Python's syntax is concise and expressive, with a minimalistic design that emphasizes code readability and reduces the cognitive load on developers.

One of Python's key strengths is its extensive standard library, which provides a wide range of modules and functions for tasks such as file I/O, networking, threading, and database access. This rich set of built-in tools allows developers to accomplish many common programming tasks without needing to rely on third-party libraries.

Python's versatility and ease of use make it well-suited for a variety of applications and domains. It is commonly used for web development, with frameworks like Django and Flask providing powerful tools for building dynamic and scalable web applications. In the field of data science and machine learning, Python has become the de facto language of choice, thanks to libraries such as NumPy, Pandas, Matplotlib, and scikit-learn, which provide powerful tools for data manipulation, analysis, visualization, and machine learning model development.

Python's popularity extends beyond traditional software development into areas such as scientific computing, automation, scripting, and game development. Its extensive ecosystem of third-party libraries and frameworks, combined with its active community and large user base, make it a versatile and powerful tool for a wide range of projects and industries.

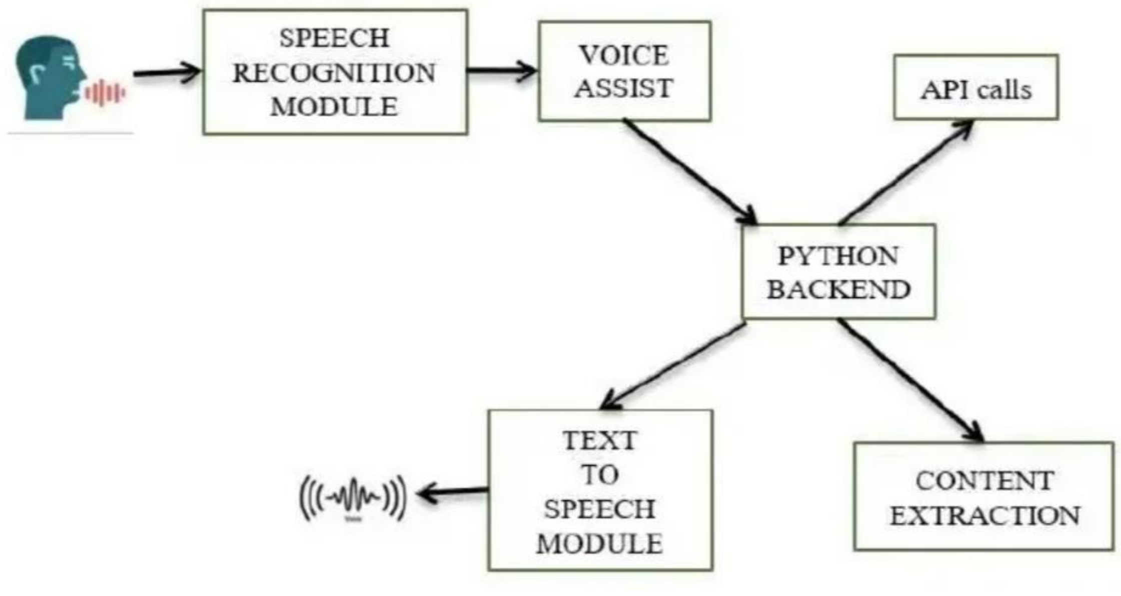
Moreover, Python's cross-platform compatibility ensures that code written in Python can run on various operating systems, including Windows, macOS, and Linux, without requiring modifications. This portability, combined with its open-source nature, makes Python an accessible and widely adopted language in both commercial and non-commercial settings.

As Python continues to evolve, with regular updates and enhancements, it remains a popular choice for developers due to its simplicity, versatility, and community support. Whether you're a beginner learning to code for the first time or an experienced developer working on complex projects, Python offers a flexible and powerful platform for turning ideas into reality.

4. SYSTEM DESIGN

4.1 ARCHITECTURE DIAGRAM

The architecture diagram of a system provides a comprehensive visual overview of its structure, components, and interactions. It serves as a blueprint for understanding how various elements of the system interact and work together to achieve its intended functionality. At its core, the diagram typically includes boxes or shapes representing individual components, such as servers, databases, services, or modules. These components are interconnected with lines or arrows to illustrate the flow of data, control, or communication between them. The architecture diagram may also incorporate layers to depict the organization of the system's components, such as presentation, application, and data layers. Additionally, it may include information about deployment configurations, such as physical or virtual infrastructure, network topology, and cloud services. Annotations, labels, or color coding may be used to provide further context, such as indicating technologies or protocols employed by each component. Overall, the architecture diagram serves as a vital communication tool for stakeholders, enabling them to grasp the system's design, dependencies, and overall architecture at a glance.



4.2 UML DIAGRAM

Unified Modeling Language (UML) diagrams are graphical representations used in software engineering to visually depict different aspects of a system. They provide a standardized way to model systems, making it easier for stakeholders to understand, communicate, and design software solutions. Here are some key points about UML diagrams:

Types of UML Diagrams: There are several types of UML diagrams, each serving a specific purpose in the software development process. Some common types include:

Use Case Diagrams : Represent the interactions between users (actors) and the system to achieve specific goals.

Class Diagrams : Show the static structure of the system, including classes, attributes, methods, and their relationships.

Sequence Diagrams : Illustrate the interactions between objects in a sequential manner over time.

Activity Diagrams : Model the flow of control within the system, representing workflows and business processes.

State Machine Diagrams : Depict the behavior of objects as they transition between different states.

Component Diagrams : Display the organization and dependencies of software components.

Deployment Diagrams : Represent the physical deployment of software components to hardware nodes.

Elements : UML diagrams consist of various graphical elements such as classes, objects, actors, use cases, associations, inheritance, messages, states, transitions, etc. These elements are used to represent different aspects of the system and their relationships.

Relationships : Relationships between elements in UML diagrams are depicted using various types of connectors such as association, aggregation, composition, generalization (inheritance), realization, etc. These relationships help in understanding how different parts of the system interact with each other.

Notation : UML diagrams follow a standardized notation defined by the Object Management Group (OMG). Each type of diagram has its own set of symbols and conventions for representing elements and relationships.

Purpose : UML diagrams serve multiple purposes throughout the software development lifecycle, including requirements analysis, system design, implementation, and documentation. They help stakeholders visualize, understand, and communicate complex systems effectively.

Tool Support: There are many software tools available that facilitate the creation, editing, and analysis of UML diagrams. These tools often provide features such as automatic layout, code generation, reverse engineering, and collaboration support.

Overall, UML diagrams are invaluable tools for software engineers, architects, designers, and other stakeholders involved in the development and maintenance of software systems. They provide a common language for expressing and communicating design ideas and system requirements.

UML defines several models for representing systems

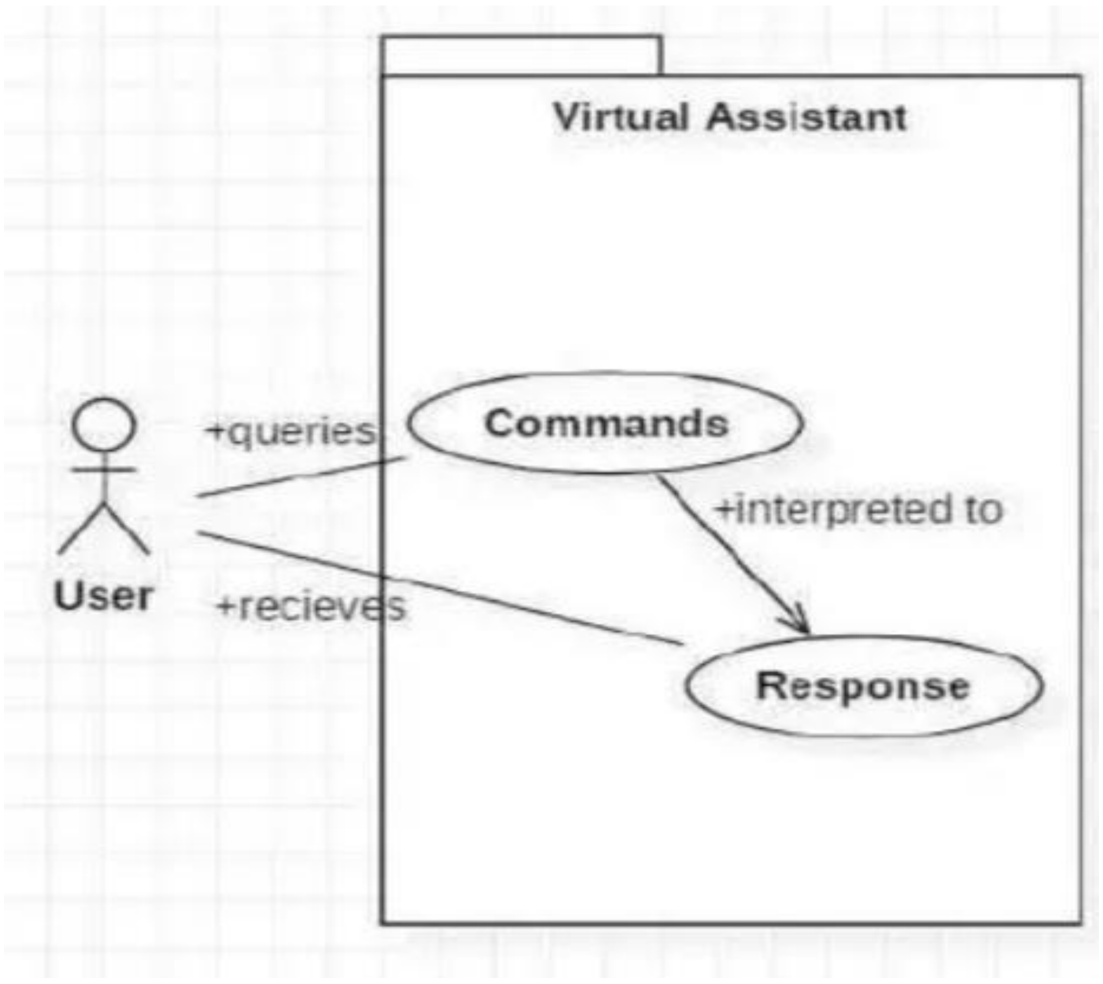
- The class model captures the static structure
- The state model expresses the dynamic behavior of objects
- The use case model describes the requirements the requirements of the user
- The interaction model represents the scenarios and messages flows
- The implementation model shows the work unit

ADVANTAGES

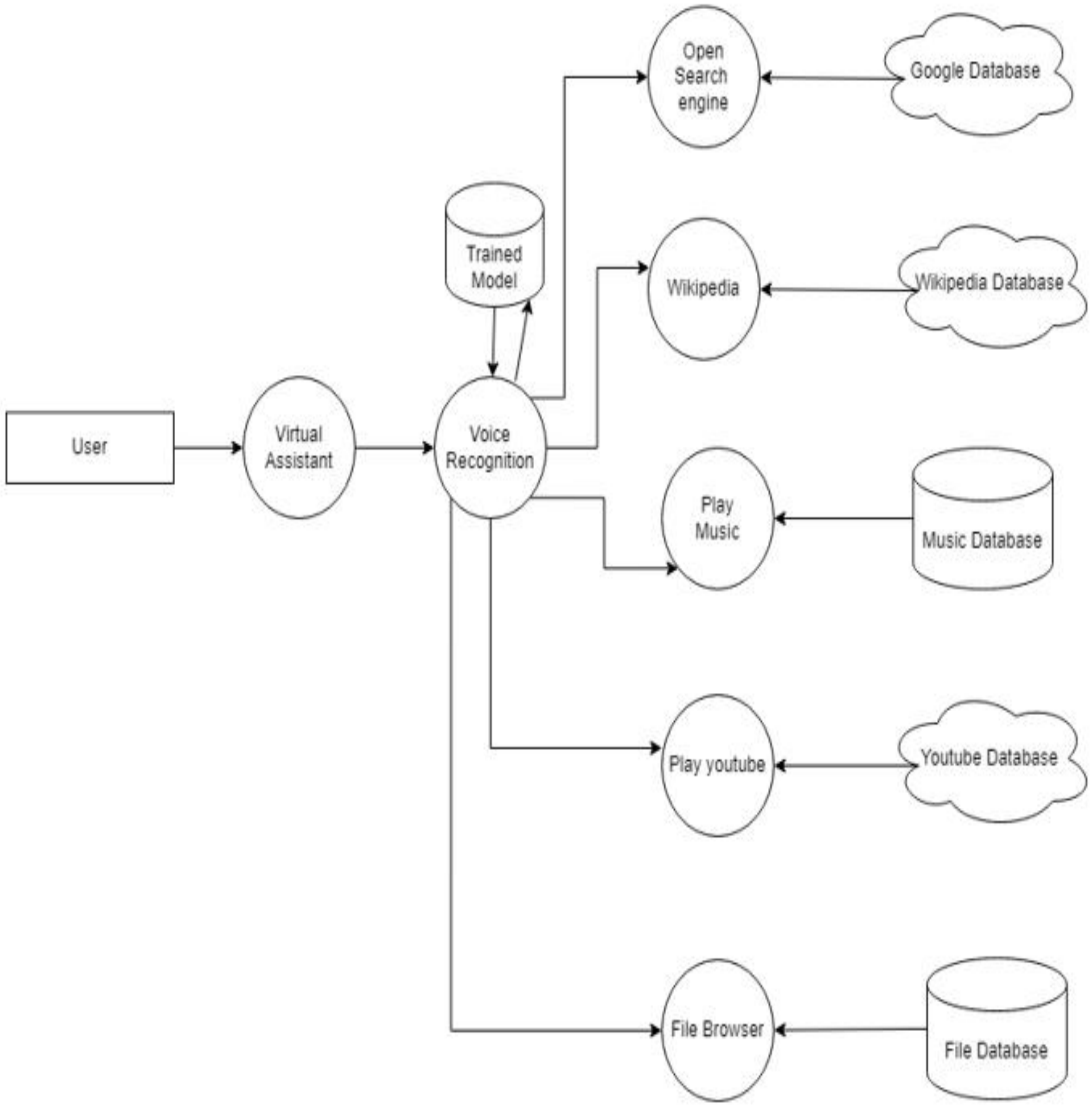
- Most used and flexible
- Development time is reduced
- Provides standard for software development
- It has large visual elements to construct and easy to follow

4.3 USE CASE DIAGRAM

A Use Case Diagram is a graphical representation within Unified Modeling Language (UML) that illustrates the interactions between users (actors) and a system to achieve specific goals or tasks. Actors, depicted as stick figures outside the system boundary, represent distinct roles interacting with the system, whether they are human users, other systems, or external entities. Within the system boundary, ovals represent use cases, each describing a particular functionality or task the system provides. These use cases depict sequences of interactions between actors and the system to accomplish specific objectives. Use Case Diagrams also depict relationships between actors and use cases, including associations, generalizations, includes, and extends, providing insights into dependencies and alternative scenarios. By visually delineating the system's scope and user interactions, Use Case Diagrams aid stakeholders in understanding system behavior, eliciting requirements, and defining project scope. They serve as invaluable tools for facilitating communication among stakeholders and guiding software development efforts from early analysis stages through implementation and beyond.



4.4 BLOCK DIAGRAM



.

5. SYSTEM CODING AND IMPLEMENTATION

Systems implementation is the process of defining now the information system should be built, ensuring that the information system meets quality standard.

Methods of implementation:

There are many ways of implementing a new system. The method chosen will depend on the organization and the type of system being implemented.

Methods of implementing system include:

- Direct cutover
- Parallel
- Phased
- Pilot

Implementation is a process of ensuring that the information system is operational. It involves,

- Constructing a new system from scratch.
- Constructing a new system from the existing one.
- Implementation allows the users to take over its operation for use and evaluation.it involves training the users to handle the system and plan for a smooth conversion.

Goals of System Implementation:

- Complete as necessary the design contained in the approved system design document. For example, the detailed contents of new or revised documents, computer screens, and database must be laid out and created.
- Write, test and document the programs and procedures required by the approved system design document.
- Ensure by completing the preparation of user manuals and other documentation and by training personnel, that the organizations personnel can operate the new system.
- Determine, by the thoroughly testing the system with user, that the system satisfies the users requirement

5.1. SYSTEM CODING

System coding definition is the process of translating system requirements into a programming language that can be executed by a computer. This process involves the following steps:

1. System analysis and design: This step involves gathering and analyzing user requirements, and then designing a system that meets those requirements.

2. Code generation: This step involves translating the system design into a programming language.
3. Testing: This step involves testing the code to ensure that it meets the system requirements and works as expected.
4. Deployment: This step involves making the system available to users.

System coding definition is a critical step in the software development process. It is important to get the system coding definition right, because any errors in the code can have a negative impact on the performance and reliability of the system.

5.2 SYSTEM TESTING

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behaviour of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS). System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. System Testing is a black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

5.3 UNIT TESTING

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit

Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

5.4 INTEGRATION TESTING

Integration testing for a budget and expense tracker involves testing the interactions between the different components of the system. This includes testing the interactions between the following components:

- Database: The database stores all of the budget and expense data. Integration testing should ensure that the budget and expense tracker can correctly read and write data to the database.

- API: The API allows users to interact with the budget and expense tracker. Integration testing should ensure that the API can correctly handle all of the supported requests.

- User interface: The user interface allows users to view and manage their budget and expenses. Integration testing should ensure that the user interface can correctly display and update the budget and expense data.

5.5 FUNCTIONAL TESTING

Functional testing is a type of software testing whereby system is tested against the functional requirements/specification. Function are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. During functional testing Black Box testing technique is used in which the testing is normally performed during the levels of System testing and Acceptance testing.

Typically, functional testing involves the following step

- Identity functions that the software is expected to perform.
- Create input data based on the function’s specifications.
- Execute the test ease
- Compare the actual and expected outputs.

5.6. ACCEPTANCE TESTING

Acceptance testing for a car rental management is the final phase of testing before the system is released to production. This phase of testing is performed by users or stakeholders to ensure that the system meets their requirements and is ready to be used.
Acceptance testing for a budget and expense tracker should cover the following areas:

- Functionality: Test all of the features of the system to ensure that they work as expected. This includes testing the ability to manage car rental, add and edit cars, and generate reports.
- Usability: Test the system to ensure that it is easy to use and navigate. This includes testing the user interface, the help documentation, and the overall user experience.
- Performance: Test the system to ensure that it can handle the expected load and perform well under pressure. This includes testing the system with a large number of users and a large number of transactions.
- Security: Test the system to ensure that it is secure and that user data is protected. This includes testing the system for vulnerabilities and attack vectors.

5.7 REGRESSION TESTING

Regression testing for a car rental management is the process of testing the system to ensure that existing functionality is not broken after new features are added or bugs are fixed. This is important because changes to the system can unintentionally introduce new bugs.
Regression testing for a car rental management should cover the following areas:

- All of the core features of the system: This includes the ability to manage car rental, add and edit cars , and generate reports.
- Any new features that have been added to the system: These new features should be thoroughly tested to ensure that they work as expected and do not break any existing functionality.
- Any areas of the system that have been fixed: These areas should be tested to ensure that the bugs have been fixed and that no new bugs have been introduced.

5.8. VALIDATION TESTING

This is final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or system testing. Black Box testing method focuses on the functional requirements of the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories: incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

5.9. UI (USER INTERFACE) TESTING

The user interface testing is also known as GUI (Graphical User Interface) testing. It refers to testing the functions of an application that are visible to user. GUI testing is testing the systems graphical user interface of the application under controls like menus, buttons, icons, toolbar, menu bar, dialog boxes and windows. The first interaction between a user and software takes place to a GUI.

In this project all the user interfaces like text boxes, icons, menu bar are working properly. The text box will be to own the user to enter the text.

5.10 BLACK BOX TESTING

Black Box Testing is also known as behavioural, opaque-box, closed-box specification based or eye-to-eye testing. It is a software testing method that analyse the functionality of a software/application without knowing much about the internal structure/design off the item that is being tested and compares the input value with the output value.

The main focus of Black Box Testing is on the functionality of the system as a whole. The term 'Behavioural Testing' is also used for Black Box. A majority off the application are tested using the black Box method. We need to cover the majority of test cases so that most of the bugs will get discovered by the BlackBox method.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input condition that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories: incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and terminated errors.

5.11 WHITE BOX TESTING

White Box testing technique analyse the internal structure the used data structure, internal design, code structure and the working of the software rather than just the functionality ass in black box testing.

White Box Testing is a testing technique which evaluates the code and the internal structure of a program.

White Box Testing involves looking at the structure off the code. When you know the internal structure of a product, tests can be conducted to ensure that the internal operations performed according to the specification and all internal components have been adequately exercised

5. SOURCE CODE

action.py

```
import datetime

import speak

import webbrowser

import weather

import os

def Action(send) :

   data_btn  = send.lower()

   if "what is your name" in   data_btn or "name" in data_btn:

     speak.speak("my name is itachi uchiha")

     return "my name is itachi Uchiha"

   elif "hello" in data_btn  or "hye" in data_btn  or "hay" in data_btn:

      speak.speak("Hey sir, How i can  help you !")

      return "Hey sir, How i can  help you !"

   elif "how are you" in  data_btn :

         speak.speak("I am doing great these days sir")

         return "I am doing great these days sir"

   elif "thanku" in data_btn or "thank" in data_btn:

        speak.speak("its my pleasure sir to stay with you")

        return "its my pleasure sir to stay with you"

   elif "good morning" in data_btn:

        speak.speak("Good morning sir, i think you might need some help")

        return "Good morning sir, i think you might need some help"
```

```
    elif "time now" in data_btn:
          current_time = datetime.datetime.now()
          Time = (str)(current_time.hour)+ " Hour : ", (str)(current_time.minute) + "
Minute"
          speak.speak(Time)
          return str(Time)
    elif "shutdown" in data_btn or "quit" in data_btn:
            speak.speak("ok sir")
            return "ok sir"
    elif "play music" in data_btn or "song" in data_btn:
        webbrowser.open("https://gaana.com/")
        speak.speak("gaana.com is now ready for you, enjoy your music")
        return "gaana.com is now ready for you, enjoy your music"
    elif 'open google' in data_btn or 'google'  in data_btn:
        url = 'https://google.com/'
        webbrowser.get().open(url)
        speak.speak("google open")
        return "google open"
    elif 'youtube' in data_btn or "open youtube" in  data_btn:
        url = 'https://youtube.com/'
        webbrowser.get().open(url)
        speak.speak("YouTube open")
        return "YouTube open"
    elif 'weather' in data_btn :
       ans   = weather.Weather()
       speak.speak(ans)
```

```
        return ans
      elif 'music from my laptop' in data_btn:
          url = 'D:\\music'
          songs = os.listdir(url)
          os.startfile(os.path.join(url, songs[0]))
          speak.speak("songs playing...")
          return "songs playing..."
elif 'search' in usr_data:
    # Extract the keyword to search
        keyword = usr_data.split('search', 1)[1].strip()
    # Construct the Google search URL
        base_url = 'https://www.google.com/search?q='
        search_url = base_url + '+'.join(keyword.split())
    # Open the search URL in a web browser
        webbrowser.get().open(search_url)
    # Speak out the confirmation message
        text_to_speech.text_to_speech("Searching for " + keyword + " on Google.")
        return "Searching for " + keyword + " on Google."
    else :
        speak.speak( "i'm able to understand!")
        return "i'm able to understand!"
```

gui.py :

```
from tkinter import*
from PIL import Image , ImageTk
import action
import spech_to_text
def User_send():
    send = entry1.get()
    bot = action.Action(send)
    text.insert(END, "Me --> "+send+"\n")
    if bot != None:
        text.insert(END, "Bot <-- "+ str(bot)+"\n")
    if bot == "ok sir":
          root.destroy()
       def ask():
    ask_val= spech_to_text.spech_to_text()
    bot_val = action.Action(ask_val)
    text.insert(END, "Me --> "+ask_val+"\n")
    if bot_val != None:
       text.insert(END, "Bot <-- "+ str(bot_val)+"\n")
    if bot_val == "ok sir":
        root.destroy()
def delete_text():
    text.delete("1.0", "end")
root = Tk()
root.geometry("550x675")
root.title("AI Assistant")
root.resizable(False,False)
root.config(bg="#6F8FAF")
# Main Frame
```

```
Main_frame = LabelFrame(root , padx=100 ,  pady=7 , borderwidth=3 ,
relief="raised")

Main_frame.config(bg="#6F8FAF")

Main_frame.grid(row = 0 ,  column= 1 ,  padx= 55 ,  pady =  10)

# Text Lable

Text_lable = Label(Main_frame, text = "AI Assistant" , font=("comic Sans ms" ,  14 ,
"bold" ) , bg = "#356696")

Text_lable.grid(row=0 ,  column=0 , padx=20 , pady= 10)

# Image

Display_Image = ImageTk.PhotoImage(Image.open("image/assitant.png"))

Image_Lable = Label(Main_frame, image= Display_Image)

Image_Lable.grid(row = 1 ,  column=0 , pady=20)

# Add a text widget

text=Text(root , font= ('Courier 10 bold') , bg = "#356696"

)

text.grid(row = 2,  column= 0)

text.place(x= 100, y= 375, width= 375, height= 100)

# Add a entry widget

entry1 = Entry(root, justify = CENTER)

entry1.place(x=100 , y = 500 , width= 350, height= 30)

# Add a text button1

button1 =  Button(root,  text="ASK" , bg="#356696" , pady=16 ,  padx=40,
borderwidth=3 , relief=SOLID ,  command=ask)

button1.place(x= 70, y= 575)

# Add a text button2

button2 =  Button(root,  text="Send" , bg="#356696" , pady=16 ,  padx=40,
borderwidth=3 , relief=SOLID ,  command=User_send)
```

```
button2.place(x= 400, y= 575)

# Add a text button3

button3 = Button(root, text="Delete", bg="#356696" , pady=16 ,  padx=40,
borderwidth=3 , relief=SOLID ,command=delete_text)

button3.place(x= 225, y= 575)

root.mainloop()
```

text_to_speach.py

```
# pip install pyttsx3

import pyttsx3

def speak(text):

    engine = pyttsx3.init()

    rate = engine.getProperty('rate')

    engine.setProperty('rate', rate-70)

    engine.say(text)

    engine.runAndWait()
```

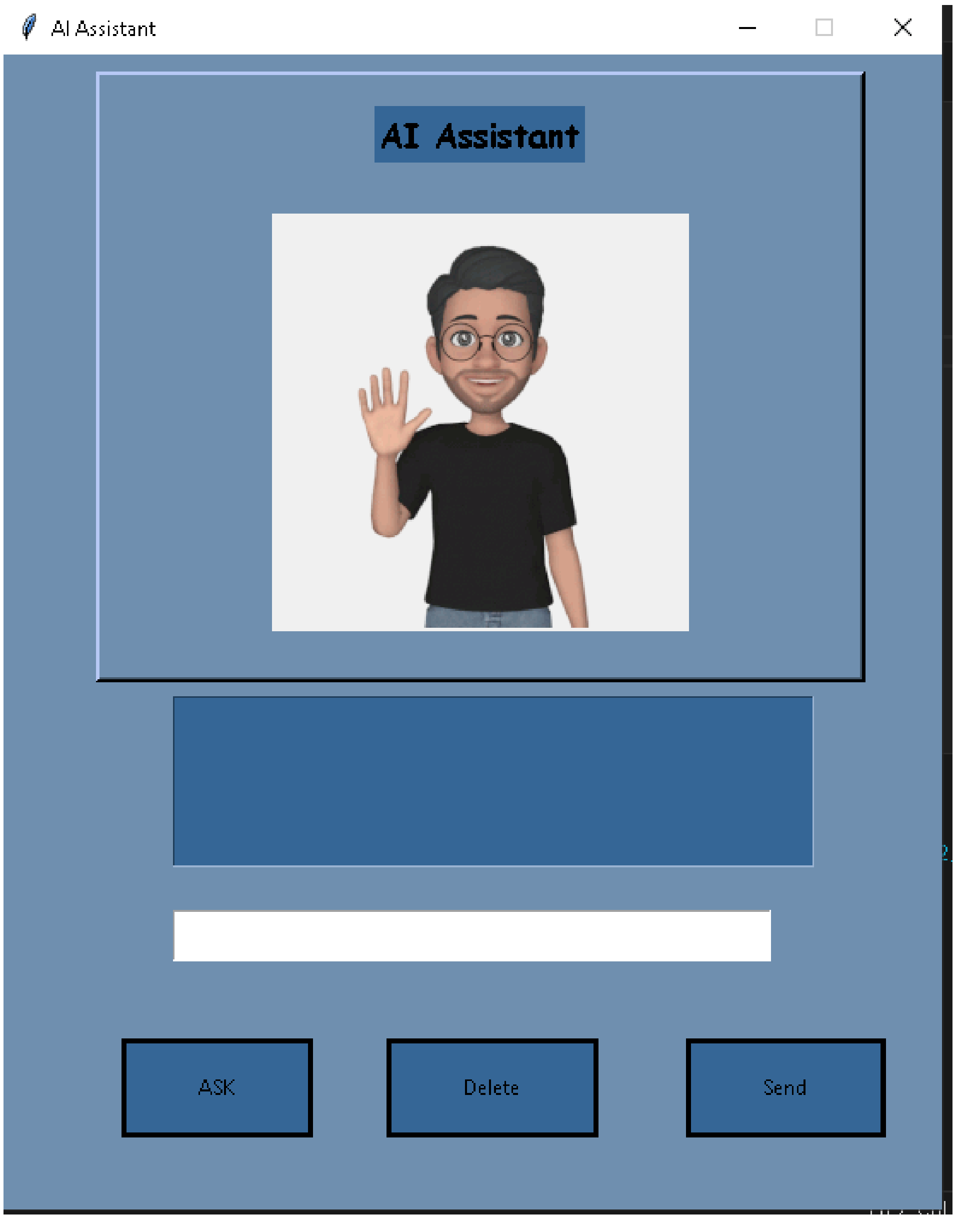
speech_to_text:

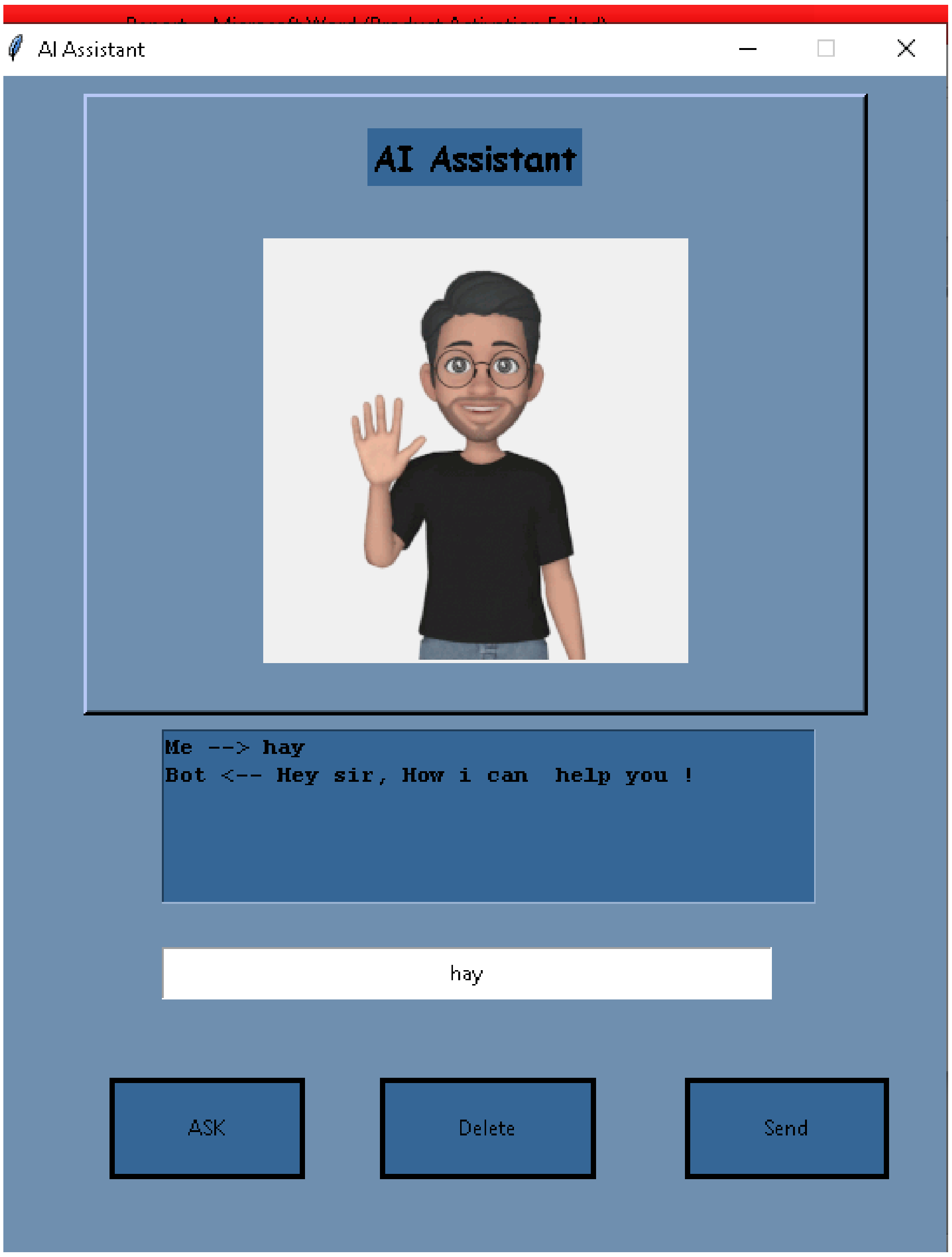
```
import speech_recognition as sr

from requests_html import HTMLSession

import speak

def spech_to_text():

    r =  sr.Recognizer()

    with sr.Microphone() as source:

      audio = r.listen(source) # methord

      voice_data = ''

      try:

        voice_data = r.recognize_google(audio)
        return voice_data
      except sr.UnknownValueError:
            speak.speak("sorry")
      except sr.RequestError:
           speak.speak('No internet connect please turn on you internet')
```

weather.py :

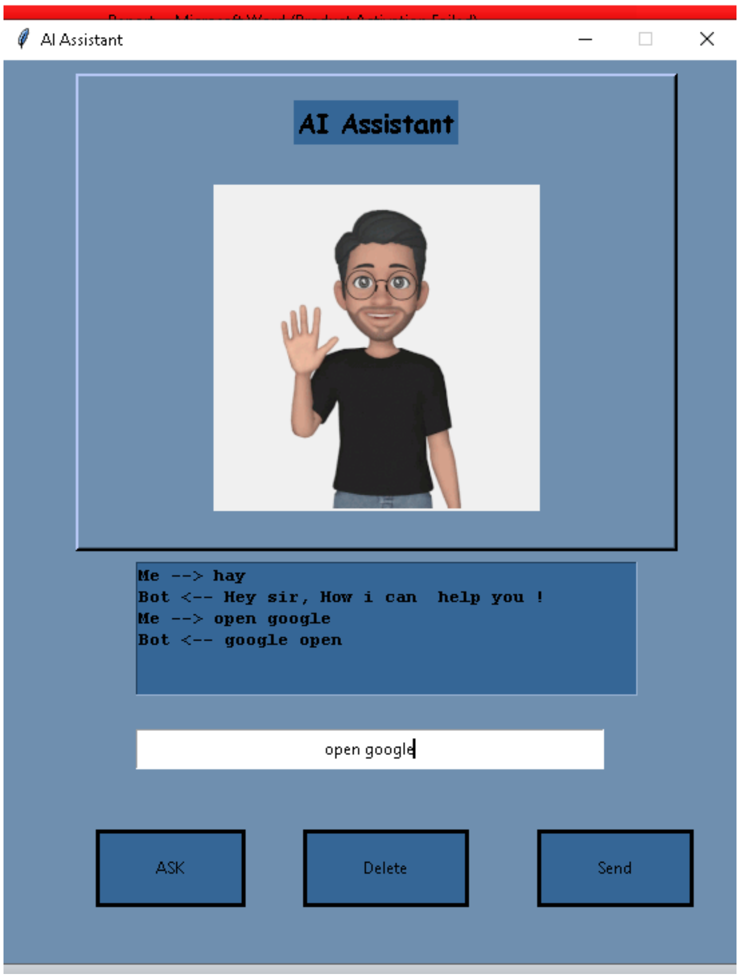
```
# my user agent is : Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36
# print(r.html.find('title' , first= True).text)
# requests-html==0.10.0
# lxml==4.9.1 (first install  this one)
from requests_html import HTMLSession
import spech_to_text
def Weather():
   s  =  HTMLSession()
   query = "patna"
   url = f'https://www.google.com/search?q=weather+{query}'
   r  = s.get(url , headers={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36'})
   temp  = r.html.find('span#wob_tm' , first= True).text
   unit = r.html.find('div.vk_bk.wob-unit span.wob_t' , first= True).text
   desc  = r.html.find('span#wob_dc' , first= True).text
   return temp+" "+unit+" "+ desc
```

7.SAMPLE OUTPUT



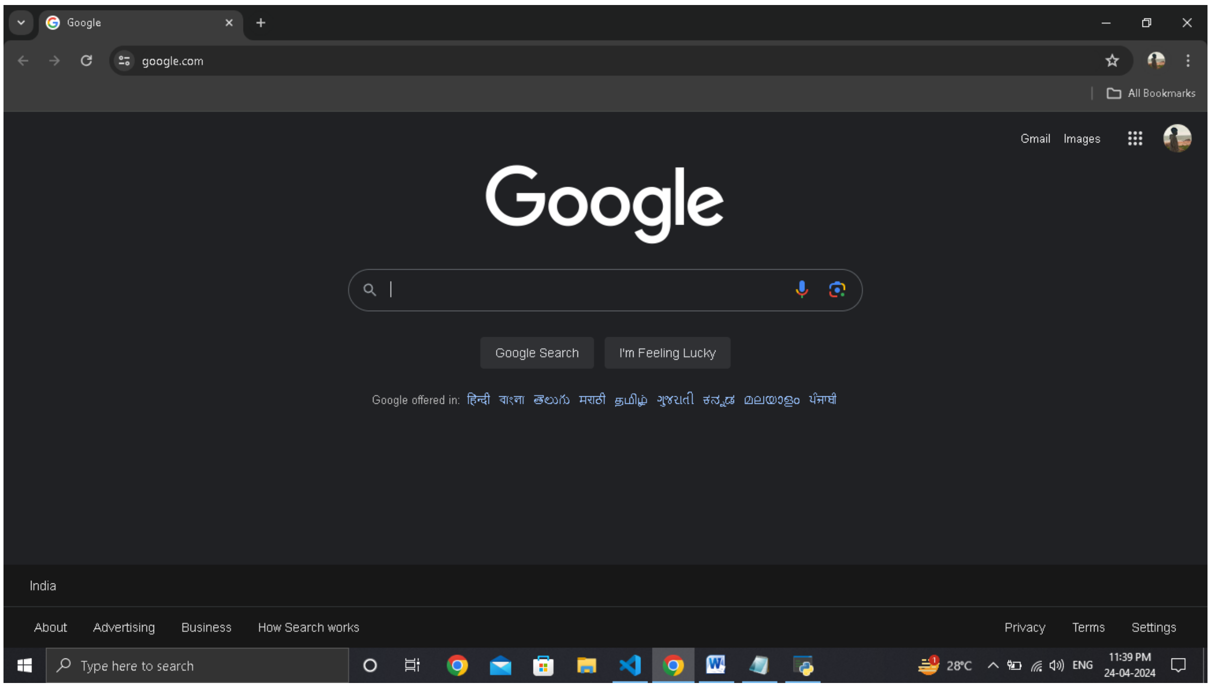


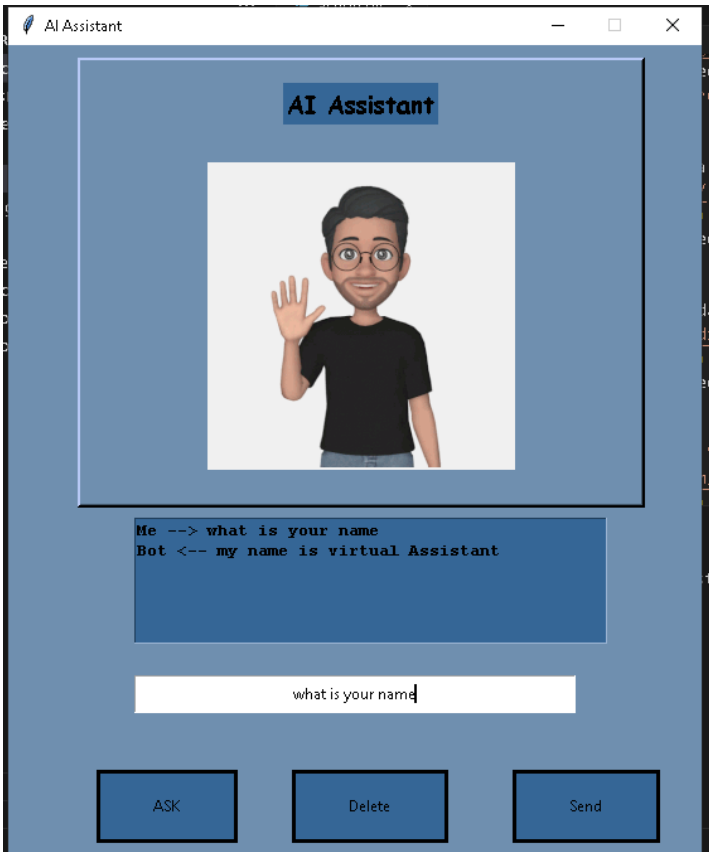
```
Me --> hay
Bot <-- Hey sir, How i can  help you !
```



```
Me --> hay
Bot <-- Hey sir, How i can  help you !
Me --> open google
Bot <-- google open
```

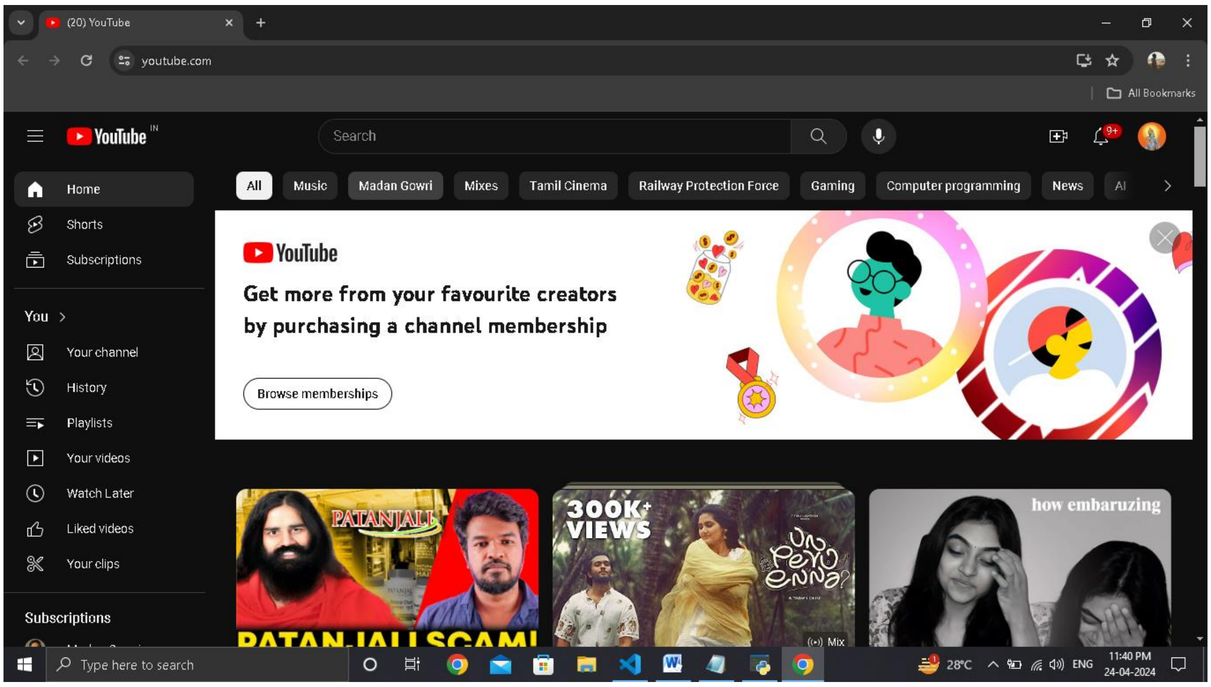
open google

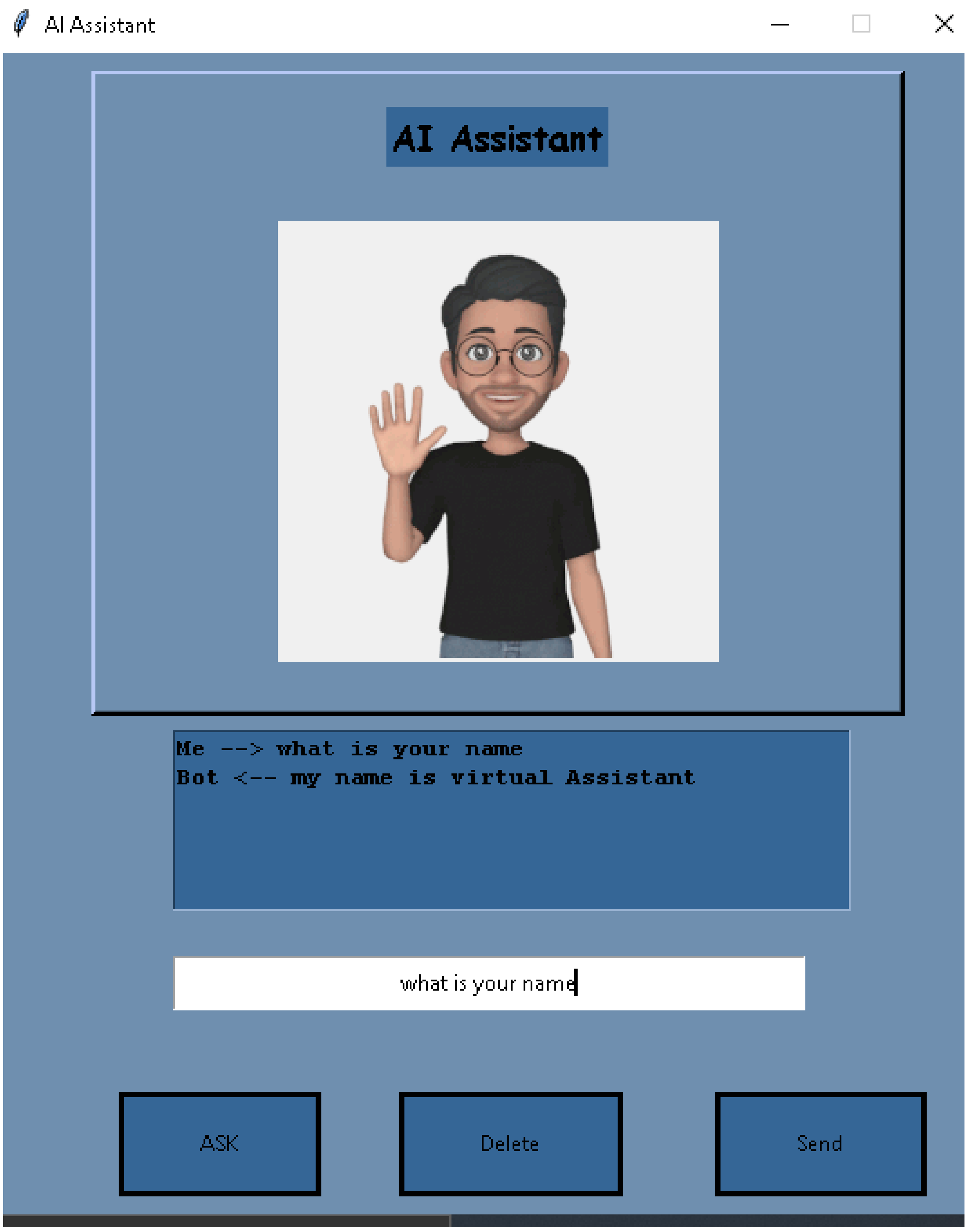




```
Me --> what is your name
Bot <-- my name is virtual Assistant
```

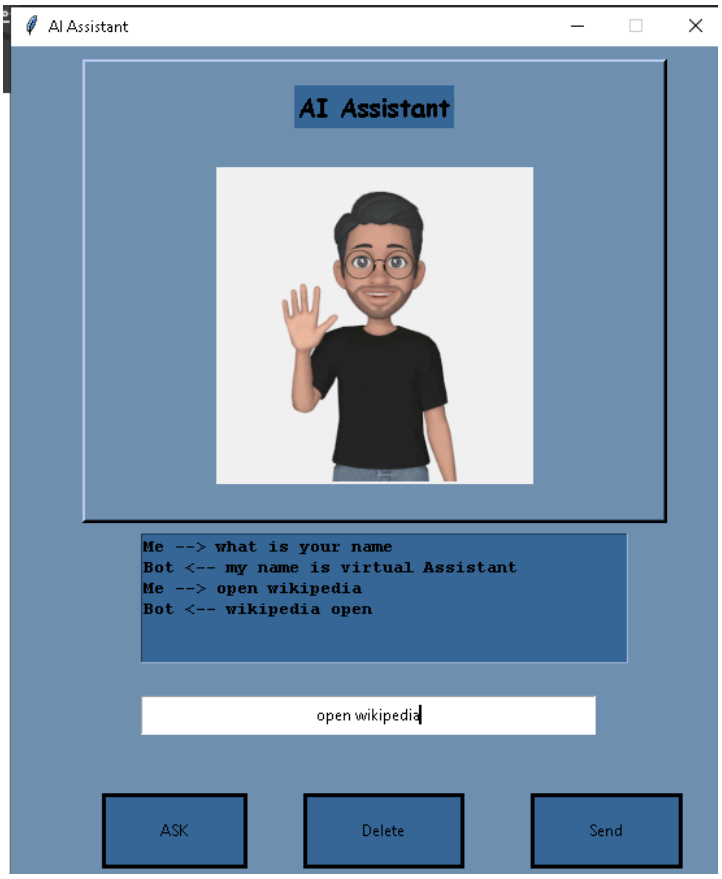
what is your name





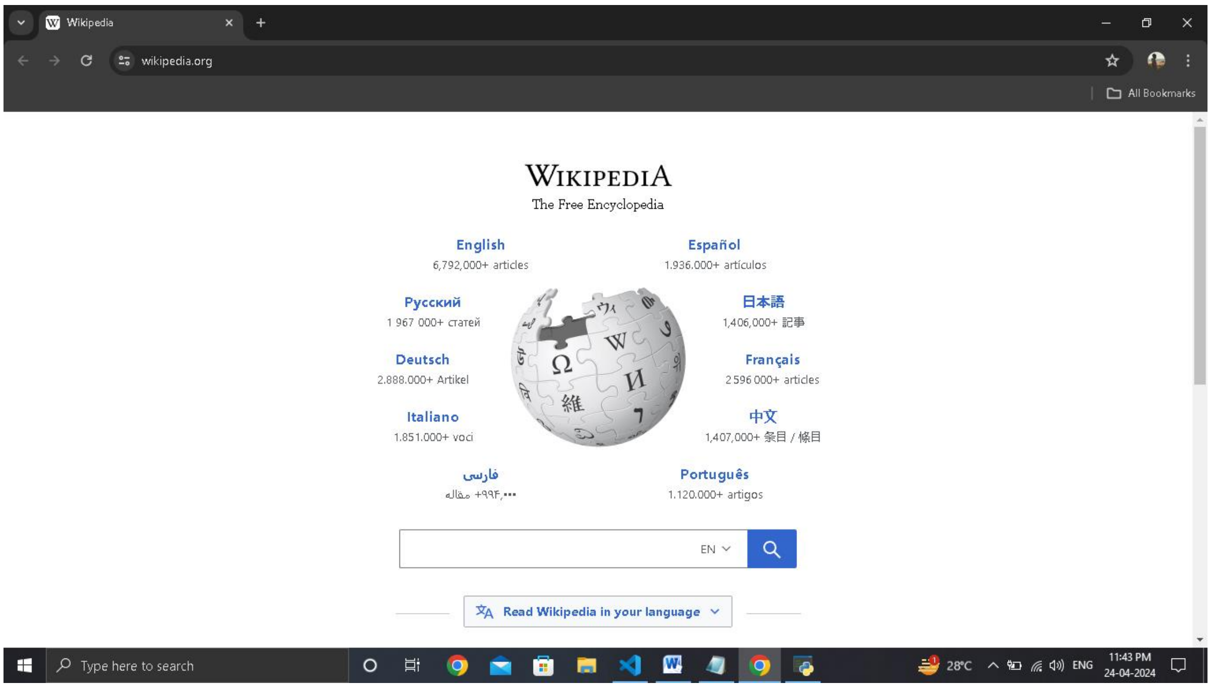
```
Me --> what is your name
Bot <-- my name is virtual Assistant
```

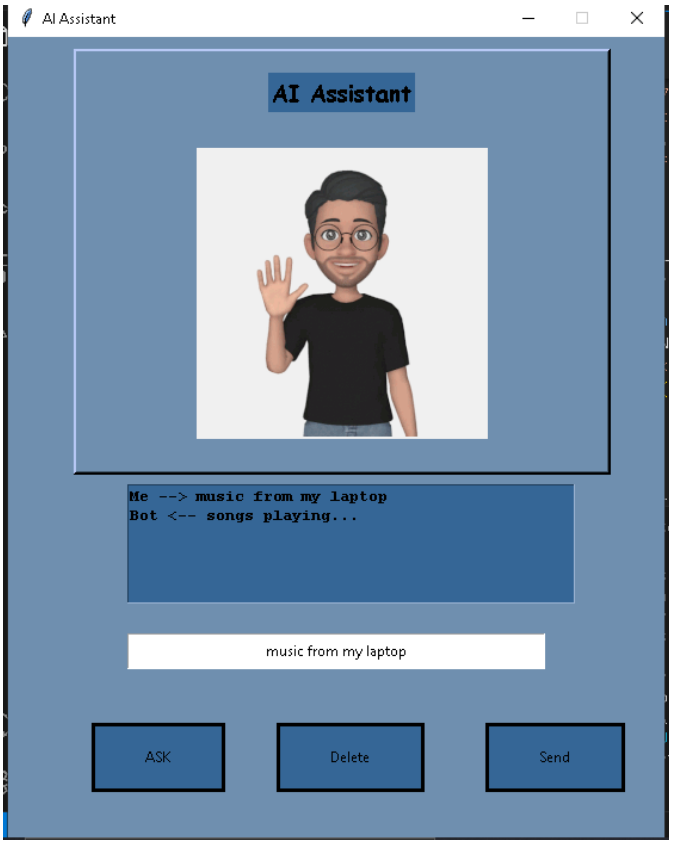
what is your name



```
Me --> what is your name
Bot <-- my name is virtual Assistant
Me --> open wikipedia
Bot <-- wikipedia open
```

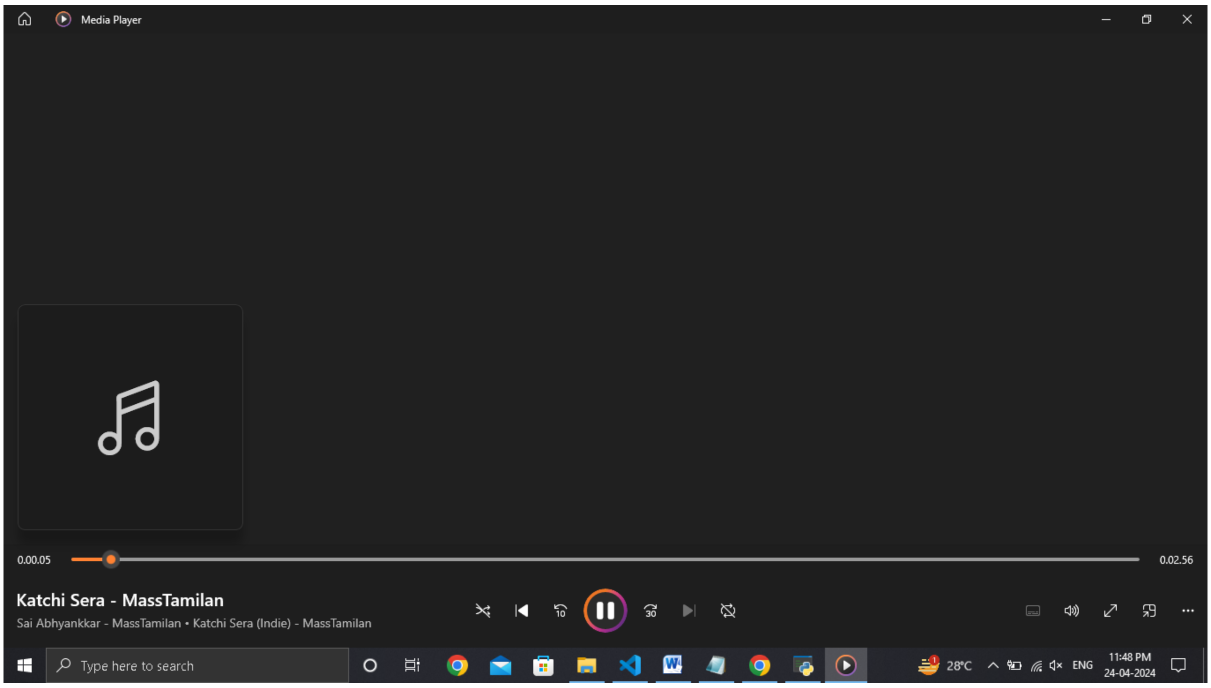
open wikipedia

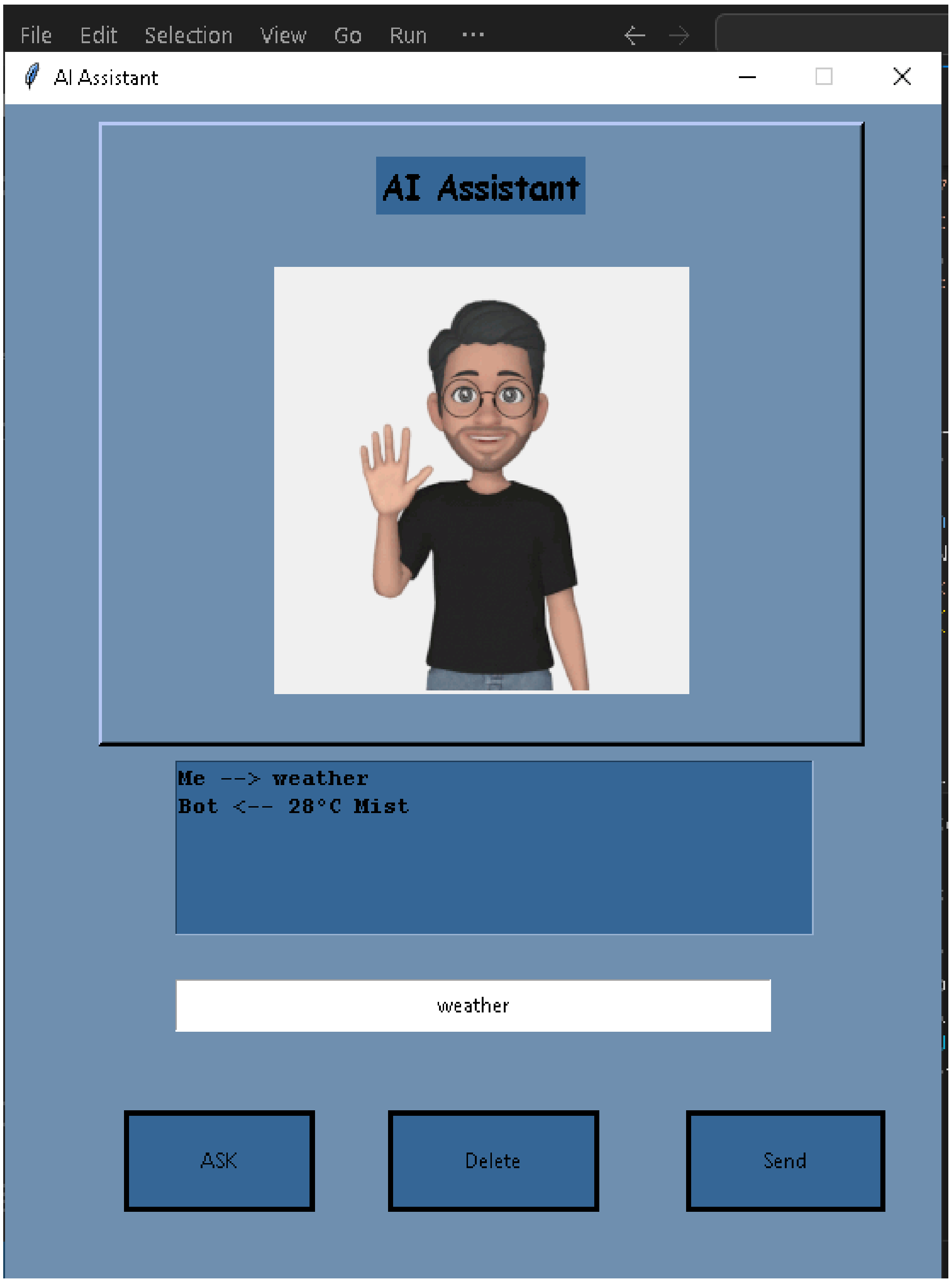




```
Me --> music from my laptop
Bot <-- songs playing...
```

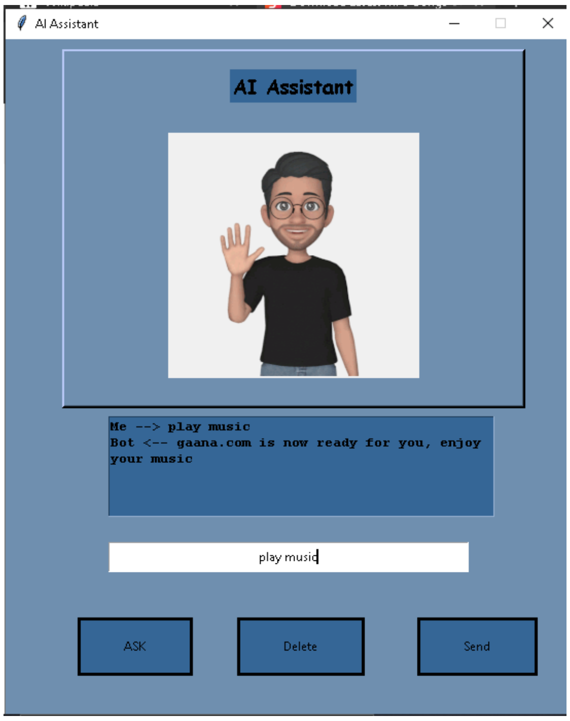
music from my laptop


Media Player
0.00.05
0.02.56
Katchi Sera - MassTamilan
Sai Abhyankkar - MassTamilan • Katchi Sera (Indie) - MassTamilan
Type here to search
28°C
ENG
11:48 PM
24-04-2024



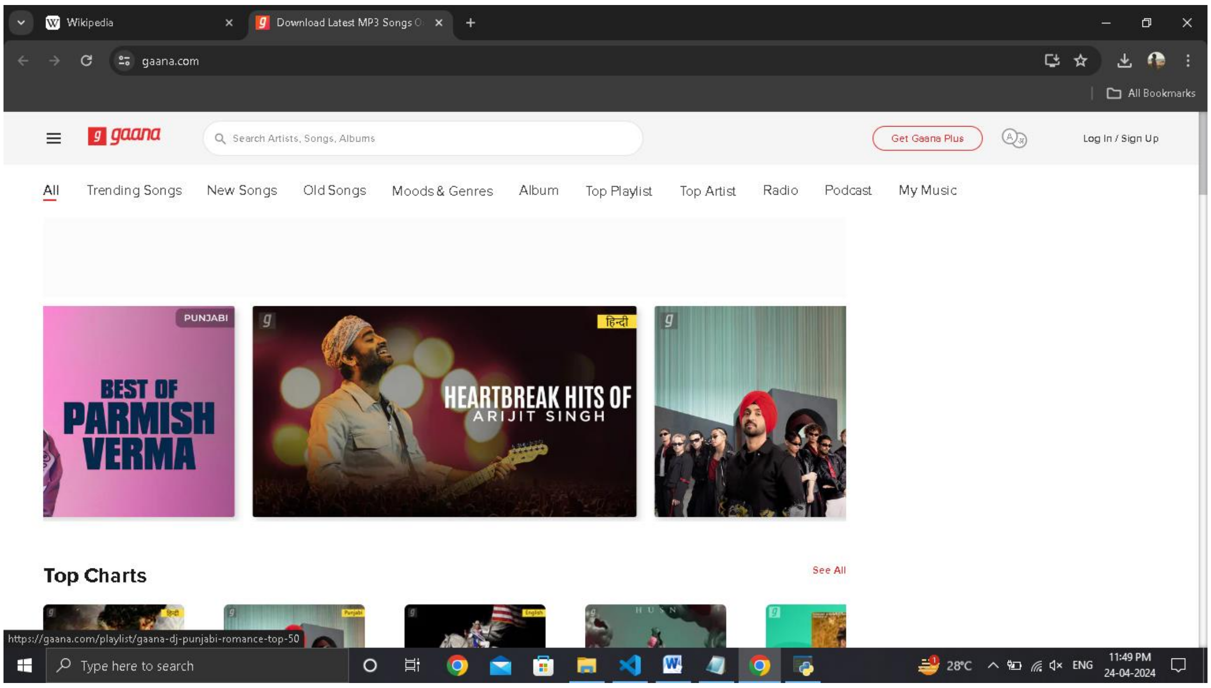
```
Me --> weather
Bot <-- 28°C Mist
```

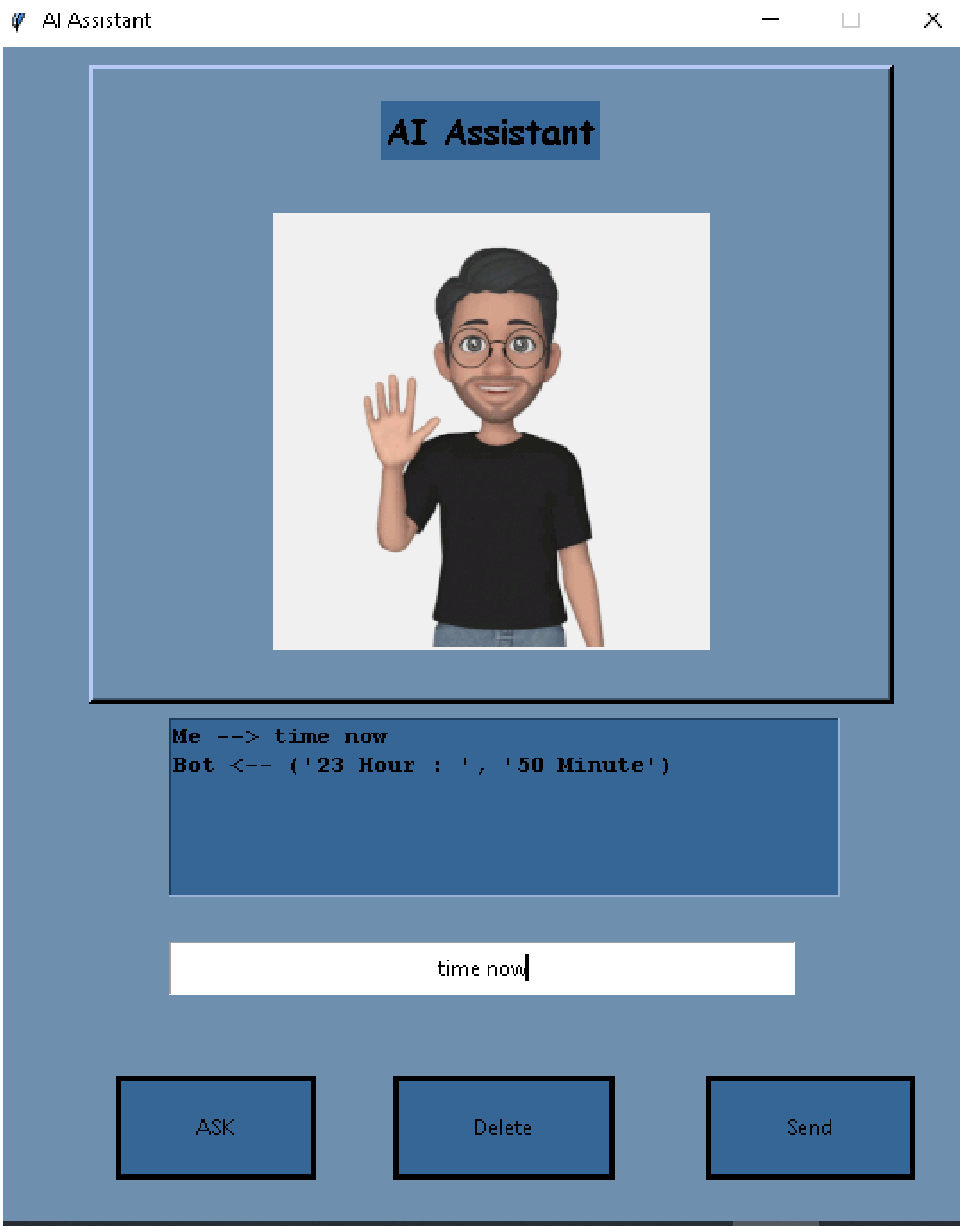
weather



```
Me --> play music
Bot <-- gaana.com is now ready for you, enjoy
your music
```

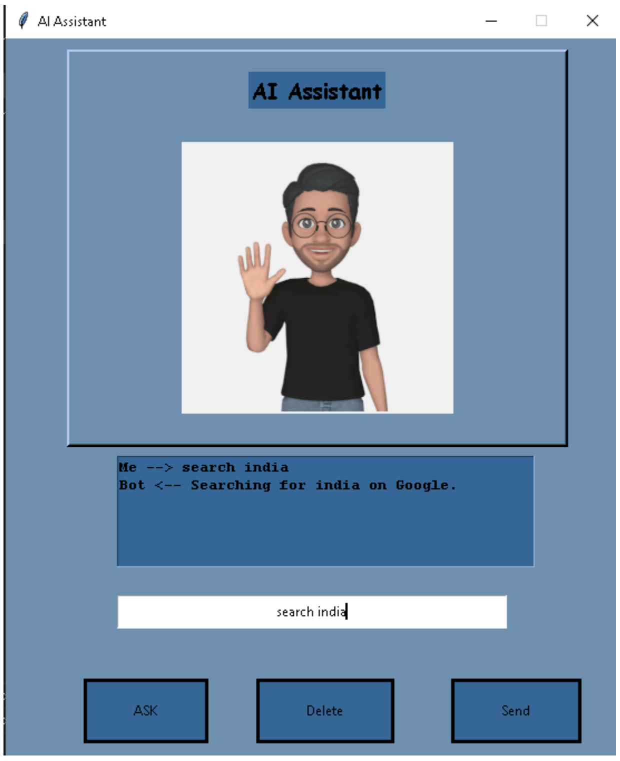
play music





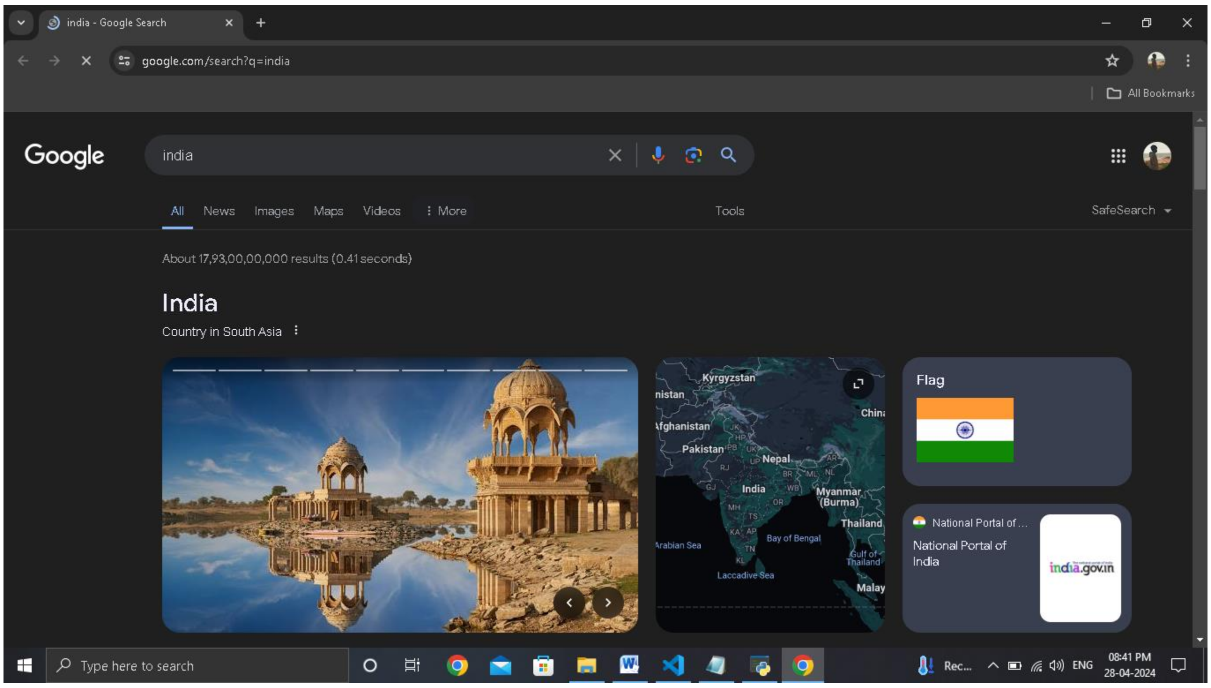
```
Me --> time now
Bot <-- ('23 Hour : ', '50 Minute')
```

time now



```
Me --> search india
Bot <-- Searching for india on Google.
```

search india



About 17,93,00,00,000 results (0.41 seconds)

India

Country in South Asia

8. CONCLUSION

Virtual assistants have fundamentally changed the way we interact with technology and have become indispensable tools in our daily lives. Their ability to understand natural language commands, perform tasks, and provide instant information has revolutionized how we shop, communicate, work, and manage our schedules. As we continue to embrace the convenience and efficiency they offer, virtual assistants are likely to become even more integrated into various aspects of our lives.

Looking ahead, the future of virtual assistants holds promise for further advancements. As artificial intelligence and natural language processing technologies continue to improve, virtual assistants will become more adept at understanding context, emotions, and user preferences, providing increasingly personalized and intuitive experiences. Moreover, with the proliferation of smart home devices, virtual assistants will play a central role in orchestrating connected ecosystems, enhancing convenience and automation in our living spaces.

However, along with these opportunities come important considerations and challenges. Privacy concerns regarding the collection and use of personal data by virtual assistants remain paramount. Striking the right balance between convenience and privacy will be crucial in ensuring user trust and acceptance. Moreover, as virtual assistants become more ubiquitous, there will be a need for clear guidelines and regulations to govern their ethical use, particularly in areas such as data security, bias mitigation, and transparency.

In conclusion, virtual assistants have transformed the way we interact with technology, offering unparalleled convenience and efficiency. Their evolution will continue to shape our digital landscape, presenting both opportunities and challenges that must be navigated thoughtfully. As we harness the potential of virtual assistants, it is essential to prioritize user privacy, ethical considerations, and inclusivity to create a future where these intelligent agents enhance our lives while respecting our rights and values.

.

9. FUTURE ENHANCEMENT

In considering future enhancements for our virtual assistant project, several areas of focus present themselves to further improve functionality, user experience, and versatility:

Advanced Natural Language Understanding (NLU): Enhance the virtual assistant's ability to comprehend complex and contextually nuanced language inputs. This includes understanding colloquialisms, slang, and regional dialects, as well as accurately interpreting user intent in ambiguous situations.

Proactive Assistance:Implement proactive features that anticipate user needs based on historical interactions, preferences, and context. This could involve suggesting relevant information, reminders, or actions without explicit user input, thereby enhancing efficiency and user engagement.

Emotional Intelligence:Integrate emotional intelligence capabilities into the virtual assistant to recognize and respond appropriately to user emotions. This may involve detecting sentiment from voice tone or text inputs and adapting responses accordingly to provide empathetic and supportive interactions.

Multi-modal Interaction: Enable multi-modal interaction capabilities to support a wider range of input and output modalities, including voice, text, touch, and gestures. This allows users to interact with the virtual assistant using the most convenient and natural method for their current context.

10. REFERENCES

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

2. Deny Nancy, Sumithra Praveen, Anushria Sai, M.Ganga, R.S.Abisree, "Voice Assistant Application for a college Website", International Journal of Recent Technology and Engineering (URTE) ISSN: 2277-3878, Volume-7,April 2019.

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

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

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

6. Kishore Kumar R, Ms. J. Jayalakshmi, Karthik Prasanna, "A Python based Virtual Assistant using Raspberry Pi for Home Automation", International Journal of Electronics and Communication Engineering (IJECE), Volume 5,July 2018.

7. M. A. Jawale. A. B. Pawar, D. N. Kyatanavar, "Smart Python Coding through Voice Recognition", International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8, August 2019.

8. Rutuja V. Kukade, Ruchita G. Fengse, Kiran D. Rodge, Siddhi P. Ransing, Vina M. Lomte, "Virtual Personal Assistant for the Blind", International Journal of Computer Science and Technology (JCST), Volume 9, October - December 2018.

9. Tushar Gharge, Chintan Chitroda, Nishit Bhagat, Kathapriya Giri, "AI-Smart Assistant", International Research Journal of Engineering and Technology (IRJET), Volume: 06, January 2019.

10. Veton Kepuska, "Next-Generation of Virtual Personal Assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google 11. Home)", PyCon, Cleveland, 2018