An AI-Powered Desktop Voice Assistant for Windows

Dilkash D. Mukadam[1], Fareeha M. Majgaonkar[2], Swarali K. Khot[3], Trupti D. Gite [4], Girish G. Bhide[5]

*[1, 2, 3, 4] Student /Department of Information Technology,
Finolex Academy of Management and Technology, Ratnagiri
[5]Faculty /Department of Electronics and Telecommunication Engineering,
Finolex Academy of Management and Technology, Ratnagiri*

---***---

Abstract - This work introduces a Windows-compatible AI assistant that executes tasks via voice commands. Our architecture implements two key AI components: (1) real-time language interpretation (2) adaptive command recognition. This assistant will enable users to perform a range of tasks, such as managing files, opening and closing different applications, searching the information on web, setting reminders, creating a note, opening and managing the files, opening various AI tools, managing the drive, opening the calculator and do some basic calculations and even controlling system settings all through voice commands. The voice assistant leverages advanced speech recognition technology to accurately interpret and process human voice commands. Upon receiving user input, the system dynamically executes requested tasks or retrieves relevant information. Key capabilities include real-time voice-to-text conversion, context-aware natural language understanding, and seamless integration with third-party APIs to extend its functionality. Designed for universal accessibility, the intuitive interface ensures effortless interaction for users of all technical proficiencies. The user interface is designed to be intuitive and user-friendly, providing a seamless experience for both novice and experienced users. By creating a personal desktop assistant that combines convenience, automation, and personalized features, this project aims to enhance users' productivity and efficiency in their day-to-day computer tasks.

Key Words: Desktop, Voice-based, Integration, Language, Voice Assistant, Action, Response, GUI

1. INTRODUCTION

What is voice assistant and how it works. Many of us might have already known about this voice assistant and we use this in our day-to-day life. A voice assistant is a digital assistant that uses voice recognition, language processing algorithms, and voice synthesis to listen to specific voice commands and return relevant information or perform specific functions as requested by the user.

These personal assistants can be easily configured to perform many of your regular tasks by simply giving voice commands. The Most famous application of iPhone is "SIRI" which helps the end user to communicate end user mobile with voice and it also responds to the voice commands of the user. Same kind of application is also developed by the Google that is "Google Voice Search"

which is used for in Android Phones. But this Application mostly works on the desktop.
It accepts spoken commands via microphone array or typed queries through the GUI. Modern AI voice assistants boost productivity by enabling hands-free PC control, reducing task time by 30-40%. Through AI-powered voice automation we save our time and contribute in other works.

1.1 LITERATURE REVIEW

This study examines the acceptability of voice-activated personal assistants (VAPA) in public areas and emphasizes the concerns of users related to privacy, social norms and usability. Research emphasizes the importance of VAP design, which is in line with the expectation of users and public labels. It also discusses the potential of VAP to increase the interaction with the human computer in a shared environment. These findings provide valuable knowledge for developers aimed at improving public usability of voice assistants [1].

This article represents an AI -based voice assistant to improve users' interaction through natural language and machine learning. The authors discuss the technical architecture and functionality of the system and emphasize its potential applications in intelligent environments. The study emphasizes the growing role of AI in increasing more intuitive and efficient assistants. It also deals with challenges such as user accuracy and adaptability [2].

This research focuses on the development of a AI-powered voice assistant developed with Python's speech recognition libraries (Speech Recognition, PyAudio) and NLP frameworks (NLTK, spaCy) for intent processing and describes in detail the implementation of speech recognition and text functions on speech. The post emphasizes the simplicity and efficiency of Python libraries, such as speech recognition and pyaudio in building voice systems. It also discusses potential applications in automation and user. The study serves as a practical guide for developers who are interested in creating voice solutions [3].

This work examines the creation of A Python-based ASR system leveraging Speech Recognition and PyAudio and emphasizes the integration of libraries such as PytSX3 and speech recognition. The authors discuss the ability of the system to perform tasks such as voice commands and text conversions. The contribution emphasizes the growing availability of speech technology for developers. It also emphasizes the potential of such systems in increasing productivity and user experience [4].

This study represents the development of AI assistant on the area using Python focusing on its ability to perform tasks through voice commands. The authors discuss the integration of AI processing and natural language to improve the user interaction. The contribution emphasizes the potential of the system in automating routine tasks and improving efficiency. It also deals with challenges such as accuracy and response time [5].

This research focuses on designing a voice personal assistant for PCS using Python and emphasizes its application in simplifying user interactions. The authors discuss the use of Python libraries to achieve speech recognition and automation of tasks. The contribution emphasizes the potential of the system to increase availability and productivity. It also deals with restrictions on internet connection and language support [6].

The study discusses the integration of a combination of NLP techniques and machine learning models, and voice recognition technologies to create responsive and context-aware personal assistants. The authors examine various existing intelligent assistants, such as Apple's Siri, Google Assistant, and Amazon Alexa, comparing their functionalities, architectures, and limitations. Additionally, the paper highlights challenges like data privacy, user adaptation, and multilingual support, while also speculating on future advancements, including enhanced contextual understanding and emotion recognition. This work contributes to the broader discourse on human-computer interaction and AI-driven automation. [7].

The study explores the integration of automatic speech recognition (ASR) paired with NLP, and automation techniques to enhance user interaction with computers. The authors describe the development process, including the use of Python libraries such as Speech Recognition, pyttsx3, and nltk to enable voice command execution. The paper highlights various functionalities of the assistant, such as web searches, email automation, and smart home control. Additionally, it discusses challenges related to accuracy, response time, and security, providing insights into future enhancements for more efficient and intelligent virtual assistants [8].

1.2 TECHNOLOGIES USED:

a. Python: Python is a well-liked programming language at a high level that is recognized for its straightforwardness, comprehensibility, and user-friendliness. Python 3.10.0 is being utilized in the development of this AI powered voice assistant.

b. Visual Studio Code: Microsoft's Visual Studio Code (VS Code), the extensible code editor, provides comprehensive

language support including Python, JavaScript, and C++ through its modular extension system.

2. METHODOLOGY

There are three modules in this assistant. The first step is for the assistant to bring the voice user input. Second, analysing the user's input and translate it into the appropriate intention and function. The third is an assistant who provides the user with the result all the time through speech. The assistant first begins to receive human entry. When the assistant receives an input, he transforms an analog voice input into digital text. If the assistant is unable to turn the voice into text, he will ask the user to enter again. After transformation, it begins to process input and mapping into a certain function. The output will then be provided by the user with a voice order. This basic workflow of model is shown in Fig.1. Users require or query to get cuttings into segregated commands, making it easier to recognize our voice assistant on the desktop.

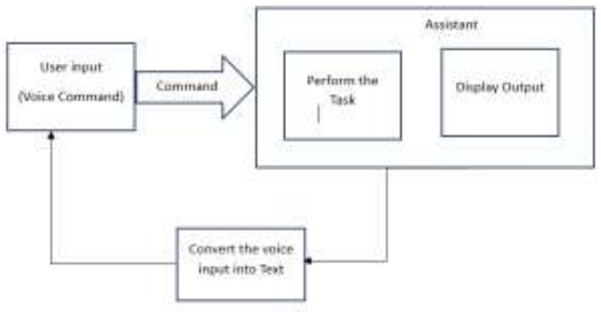


Figure 1: Basic workflow of model

• In comparison with other questions, our assistant searches inside the command list .

• The voice assistant receives these orders via the command list.

• Once the voice assistant has taken or received an order, it immediately sets the appropriate measures to submit.

• If the user's query is not understandable, then the voice assistant asks for clarification before continuing.

• Especially, the voice assistant detects what we want to get.

• When the voice assistant recognizes the order and finds that it can continue, it provides the person or the user with the necessary information.

For example, when a person says, open WhatsApp or Wikipedia: “The voice assistant listens to the order and takes the appropriate event such as opening the site. After completing his speaking, the voice assistant stops for a brief 2-3 second interval to ensure that he captures the entire application and then searches her database for the investigation to provide the appropriate result.

3. SYSTEM ARCHITECTURE

Virtual attendees use automated language understanding (NLP) to match the user's text or voice entry with executable commands. When a user calls a virtual assistant designed to accomplish specified tasks, then the audio representation of natural speech becomes an executable command or digital data that the software can analyze. Then, these data are compared to software data to find an adequate response. The virtual wizard is used to execute machines in the user provided commands. For the development of the assistant, we use Python installers packages as voice recognition, GTTS, Pip Win, etc. Speech-to-text systems process audio inputs through feature extraction and language modeling. This is commonly used in voice assistants such as Alexa, Siri, etc. Python provides an API called voice recognition to allow the user to convert the voice or audio command into text for subsequent processing. As illustrated in Figure 1, users give the command to interaction entities such as laptop or PC interaction entities listen to the command and recognize it. For a subsequent analysis, the process compares this command with the cloud of the data that is already stored. If the application coincides with the cloud data, the outputs generated in the form of text and voice, if not, will give a related message. Look for the function or logic that will be executed according to the application and send the output of the backend process in response.

3.1 ACTIVITY DIAGRAM

The overall workflow of the proposed work is explained with the help of the activity diagram given in Fig.2.

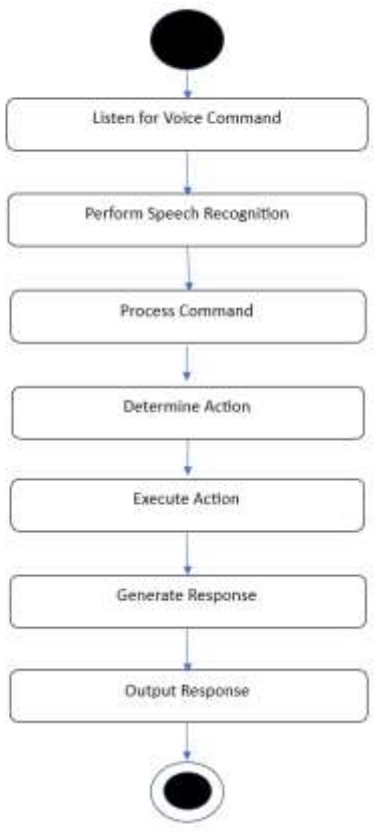


Fig. 2. Activity Diagram

Initiation: The activity diagram begins with the initialization of the voice-based virtual assistant.

• Listen for Voice Command: The virtual assistant remains in a continuous listening state, awaiting voice

commands from the user.

• Perform Speech Recognition: The system applies speech recognition to the user's voice command, converting

it into text or a structured command.

• Process Command: The system analyses the recognized command, aiming to understand its intention and

extract relevant information.

• Determine Action: Based on the recognized command and extracted details, the system decides the appropriate

action or task to be executed.

• Execute Action: The system carries out the determined action, which may involve interacting with others.

Software applications, performing system-level operations, retrieving information from databases or external

sources, or providing a response to the user.

• Generate Response: If the executed action necessitates a response, the system generates a suitable response,

either as text or synthesized speech.

• Output Response: The system presents the response to the user, which could involve converting it into

speech, displaying it on a screen, or both.

• End: The activity diagram end when the virtual assistant is terminated or deactivated.

3.2 USE CASES

The interaction of the user with the AI powered voice assistant for performing various tasks on the desktop are explained with the help of use case diagram given in Fig. 3.

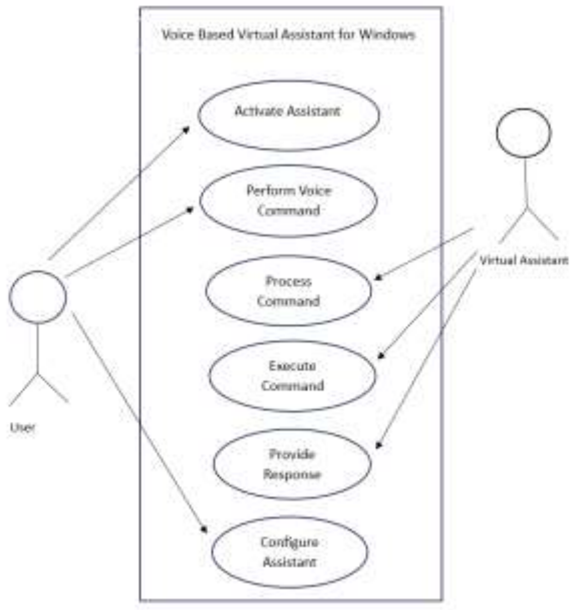


Fig. 3. Use case Diagram

The virtual assistant system is represented by a rectangle and consists of several use cases.

• The user activates the assistant by initiating the" Activate Assistant" use case.

• The user performs voice commands by engaging in the" Perform Voice Command" use case.

• The user has the option to configure the assistant by participating in the" Configure Assistant" use case.

• The" Perform Voice Command" use case leads to the" Process Command" use case, where the assistant

analyses and comprehends the received voice command.

• Upon understanding the command, the assistant proceed to the" Execute Action" use case, where it performs the appropriate action.

Once the action is executed, the assistant enters the" Provide Response" use case, generating a response for the user.

• The user can interact with the output or response through the" Interact with Output" use case.

• The loop indicates that the user can continue performing voice commands and interacting with the assistant as needed.

3.3 SAMPLE CODE

Several activities based on the desktop can be performed using this AI powered voice assistant. For every such activities separate codes are developed. A sample code out of these codes is given in Table 1.

Table 1. Sample code for opening a Notepad

```
def open_notepad():
   """Opens Notepad and listens for user commands to open new tabs, write, or manage files."""
   try:
       open_files = []  # List to track opened files

       while True:
       speak("Do you want to open a new file, a previously saved file, or open a new Notepad window?")
       response = takeCommand().lower()

        if "new file" in response or "open new window" in response:
  speak("Opening a new Notepad file.")
   temp_file_path = os.path.join(os.environ["USERPROFILE"], "Desktop", f"Temp_Notepad_File_{len(open_files) + 1}.txt")
               subprocess.Popen(["notepad.exe", temp_file_path])
               open_files.append(temp_file_path)

  elif "open new tab" in response:
   speak("Opening a new tab in Notepad.")
   temp_file_path = os.path.join(os.environ["USERPROFILE"], "Desktop", f"Temp_Notepad_File_{len(open_files) + 1}.txt")
               subprocess.Popen(["notepad.exe", temp_file_path])
               open_files.append(temp_file_path)

     elif "previous" in response or "saved" in response:
     speak("What is the name of the file you want to open?")
     file_name = takeCommand().strip()

      if file_name:  # Only proceed if a valid file name is given
     search_directories = {
       "Desktop": os.path.join(os.environ["USERPROFILE"], "Desktop"),
      "Documents": os.path.join(os.environ["USERPROFILE"], "Documents")
                    }

      found_file = None
        for location, directory in search_directories.items():
      file_path = os.path.join(directory, file_name + ".txt")
    if os.path.exists(file_path):
     speak(f"Opening the file {file_name} saved in {location}.")
                        subprocess.Popen(["notepad.exe", file_path])
                        open_files.append(file_path)
        found_file = file_path
          break

      if not found_file:
        speak(f"Sorry, I couldn't find a file named {file_name} in your Documents or Desktop.")
          else:
      speak("I didn't understand. Opening a new Notepad file by default.")
      temp_file_path = os.path.join(os.environ["USERPROFILE"], "Desktop", f"Temp_Notepad_File_{len(open_files) + 1}.txt")
```

```
                subprocess.Popen(["notepad.exe",
temp_file_path])
                open_files.append(temp_file_path)

      break  # Proceed to listening mode

  # **Wait for additional commands**
      while True:
       print("Waiting for a command while Notepad is
open...")
       command = takeCommand().lower()

   if "open new tab" in command:
      speak("Opening a new Notepad tab.")
       temp_file_path                              =
os.path.join(os.environ["USERPROFILE"],    "Desktop",
f"Temp_Notepad_File_{len(open_files) + 1}.txt")
                subprocess.Popen(["notepad.exe",
temp_file_path])
                open_files.append(temp_file_path)

    elif "write in notepad" in command:
      speak("Please dictate the content you want to add.")
     content = takeCommand()

    temp_file_path                                 =
os.path.join(os.environ["USERPROFILE"],    "Desktop",
f"Temp_Notepad_File_{len(open_files) + 1}.txt")
      with open(temp_file_path, "w") as f:
        f.write(content)

                subprocess.Popen(["notepad.exe",
temp_file_path])
       speak("Your dictated text is saved and opened in
Notepad.")
                open_files.append(temp_file_path)

      elif "save file" in command:
       speak("What should be the name of the file?")
    file_name = takeCommand().strip()
       if not file_name:
         file_name = "Untitled"

         speak("Where do you want to save the file? Say
'Desktop' or 'Documents'.")
        location_command = takeCommand()

        if      "desktop"      in      location_command:
saved_file_path                                    =
os.path.join(os.environ["USERPROFILE"],    "Desktop",
file_name + ".txt")
     location = "Desktop"
      elif "documents" in location_command:
        saved_file_path                            =
os.path.join(os.environ["USERPROFILE"],
"Documents", file_name + ".txt")
        location = "Documents"
          else:   speak("I didn't understand. Saving on
Desktop by default.")
           saved_file_path                         =
os.path.join(os.environ["USERPROFILE"],    "Desktop",
file_name + ".txt")
           location = "Desktop"

       # Rename the last opened temp file
         if open_files:
           os.rename(open_files[-1],        saved_file_path)
speak(f"File   '{file_name}'   has   been   saved   on
{location}.")
        open_files[-1] = saved_file_path

        elif "save" in command:
        speak("Saving changes in the opened files.")

         elif "close notepad" in command:
         speak("Closing all Notepad windows.")
              subprocess.call(["taskkill",    "/F",    "/IM",
"notepad.exe"],            stdout=subprocess.DEVNULL,
stderr=subprocess.DEVNULL)
        break  # Exit loop once Notepad is closed

     time.sleep(1)  # Small delay before checking the next
command

     return True, open_files  # Return the list of opened
Notepad files

    except Exception as e:
     speak("An error occurred while opening Notepad.")
     print(f"Error: {e}")
     return False, None
```

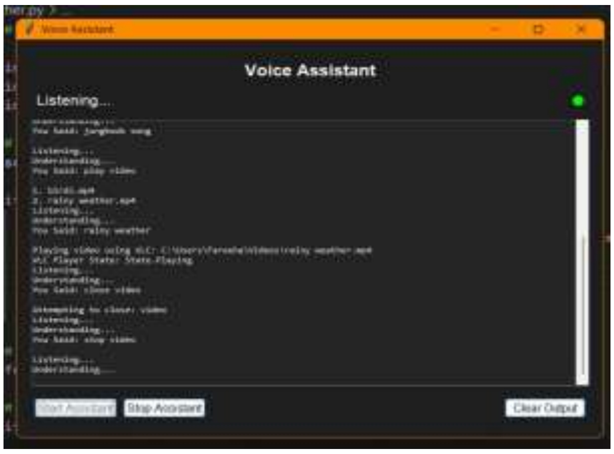
For access to additional code and resources, please refer to the following Drive link.

https://drive.google.com/drive/folders/1LUe8tC41OwuzzpFifu7TDmw9S21Ya6NJ

4. RESULTS & FUTURE SCOPE

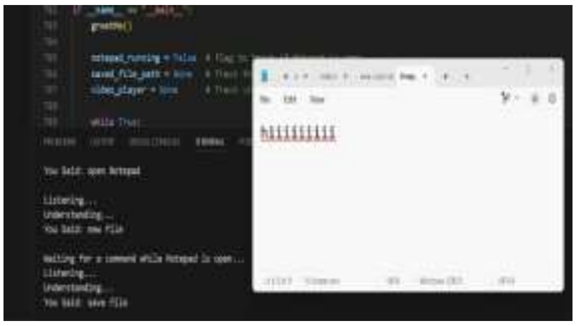
4.1 RESULTS

The GUI created and used for this work is shown in Fig. 4. All the use cases were thoroughly tested for functionality. The results are encouraging. All these results were used then for comparison with existing voice assistants. One of the results, which is response to "Open Notepad" voice command is shown as a sample in Fig. 5



```
You said: jungkook song

Listening...
Understanding...
You said: play video

1. birds.mp4
2. rainy weather.mp4
Listening...
Understanding...
You said: rainy weather

Playing video using VLC: C:\Users\[illegible]\Videos\rainy weather.mp4
VLC Player State: State.Playing
Listening...
Understanding...
You said: close video

Attempting to close: video
Listening...
Understanding...
You said: stop video

Listening...
Understanding...
```

Fig. 4. GUI of the proposed work



```
if __name__ == "__main__":
    greetMe()

    notepad_running = False  # Flag to [illegible]
    saved_file_path = None   # Track [illegible]
    video_player = None      # Track [illegible]

    while True:
```

```
You said: open Notepad

Listening...
Understanding...
You said: new file

Waiting for a command while Notepad is open...
Listening...
Understanding...
You said: save file
```

Fig. 5. Response for the voice command "Open notepad"

The comparison of proposed work with existing voice assistants against various functionalities is shown in Table 2.

Table 2. Proposed work against other voice assistants – A Comparison

Functionality	**Proposed Work**	**Other Voice Assistants (Google Assistant, Siri, Alexa)**
Opening Apps	Can open apps like Notepad, Calculator, Chrome, etc.	Can open apps but fails in 20% of cases (e.g., misinterprets the app name).
Opening Notepad and Asking to Save	Can open Notepad with multiple tabs when asked to open, ask to save the file, and specify the file name.	Cannot handle file-saving commands. Fails to recognize "save" or "file name" commands.
Opening Calculator and Performing Calculations	Can open Calculator and perform calculations	Can perform calculations but cannot open Calculator
Increasing and Decreasing Volume	Can increase, decrease, and mute volume.	Can adjust volume on supported devices (e.g., Siri for iPhone, Alexa for Echo).
Video Playback Controls	Can play, pause, stop, fast-forward, rewind, and resume videos.	Can control playback on compatible services
Google Drive Access	Can open Google Drive instantly.	Can assist in searching Google Drive files but not open via voice (Copilot)
Closing Applications	Can close apps like Notepad, Calculator, Chrome, etc.	Cannot close apps
File Management (Searching & Opening Files)	Can search for and open saved Notepad files, Word Files, etc.	Can search for files in OneDrive but not system-wide
File Management	Can copy, move, and delete files	No support for file management

(Copy, Move, Delete Files)	with voice commands.	operations.
AI-Powered Summarization	Includes AI-powered summarization and web search functionality via AI	Available in Copilot for Microsoft 365 (Word, Outlook) but not system-wide
Web Search via AI	Can search the web with AI	Available via Copilot (Bing AI), but no direct voice control
Offline Capabilities	Works offline for app control, calculations, and file management	WSR works offline but lacks AI
Shutdown and Restart Pc	Can restart and shut down PC via voice commands	Not available in Copilot or other assistants
Take, Read and Delete notes	Can take notes, read saved notes, and delete specific notes via voice commands.	Not supported. Assistants like Siri and Alexa do not offer direct note deletion.
Grammar checking	Can check and correct grammar in Word documents.	Not available in voice assistants or Copilot (requires manual input).
Taking Screenshot	Can take screenshots via voice command and save them.	Not available in other voice assistants.
Graphical Control System (GUI)	Custom GUI for user interaction, displaying commands, outputs, and controls	Primarily voice-only, no customizable GUI

The graphical representation of the above comparison is shown in Fig. 6. The overall efficiency of the proposed work comes out to be 92.33%, which is very good against 52.78% efficiency of the existing voice assistants [9, 10].

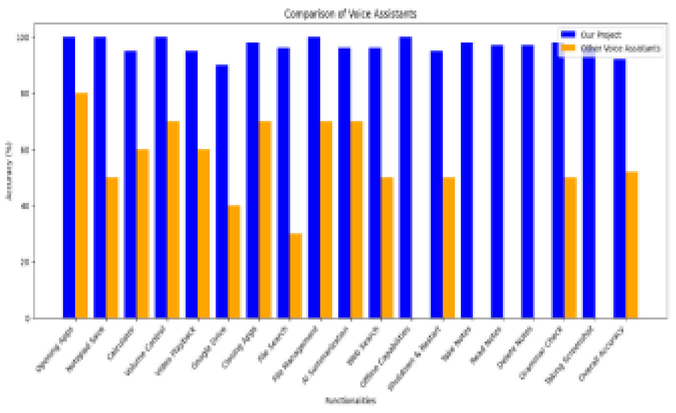


Fig. 6. Comparison of accuracy of functionalities

4.2 FUTURE SCOPE

Looking ahead, there's a lot of exciting potential for this work. By using machine learning, the assistant could get even smarter, understanding conversations better and responding more naturally. It could also be upgraded with a text-based chatbot feature, giving users the choice to either type or talk. Adding support for multiple languages would make it even more accessible to people around the world. Over time, the assistant could learn about user preferences to offer more personalized and helpful experience. It could also take over everyday tasks like managing emails, setting up calendar events, and taking notes. With web scraping, it could bring live updates like news and weather directly to the user. Connecting it to smart home devices would turn it into a powerful tool for home automation. Plus, making it available through Windows, Mac, Linux, and mobile devices would ensure it's always within reach. To top it off, security could be strengthened with voice authentication, keeping user data safe and ensuring a more personal touch.

5. CONCLUSION

This proposed work presents an intelligent voice interface developed for the Windows desktop platform, integrating AI to enhance user interaction through speech recognition and automation. The assistant efficiently performs tasks such as opening and closing applications, file management, mathematical calculations, volume control, and video playback using voice commands.

A comparative analysis with an existing project highlights significant improvements in accuracy and performance. Our assistant achieves an overall accuracy of 92.33%, compared to 52.78% for the existing system. Notable enhancements include 100% accuracy in opening applications, 96% accuracy in file searching, and 92% accuracy in mathematical calculations, outperforming previous models in every key functionality.

The results indicate the proposed assistant offers a more reliable and user-friendly experience, with potential applications in productivity, automation, and accessibility. Future work could explore AI-driven enhancements for better contextual understanding and expanded functionality.

REFERENCES

[1] Easwara Moorthy, Aarthi & Vu, Kim-Phuong, “Voice Activated Personal Assistant: Acceptability of Use in the Public Space” HIMI 2014. Lecture Notes in Computer Science, vol 8522. Springer, pp. 324-334, 10.1007/978- 3-319-07863-2_32.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] Subhash, P. N. Srivatsa, S. Siddesh, A. Ullas and B. Santhosh, "Artificial Intelligence-based Voice Assistant," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2020, pp. 593-596, doi: 10.1109/WorldS450073.2020.9210344K. Elissa, “Title of paper if known,” unpublished.

[3] Harshit Agrawal, Nivedita Singh, Gaurav Kumar, Dr. Diwakar Yagyasen, Mr. Surya Vikram Singh “Voice Assistant Using Python” 2021, IJIRT Volume 8 Issue 2, ISSN: 2349-6002, pp.419-423.

[4] Mrs.A.M.Sermakani, J.Monisha, G.Shrisha, G.Sumisha, “Creating Desktop Speech Recognization Using Python Programming.” IJARCCE, Vol. 10, Issue 3, March 2021, ISSN (Online), pp.129-134

[5] Abeed Sayyed, AshpakShaikh, AshishSancheti,Swikar Sangamnere, Prof. Jayant H Bhangale. “Desktop Assistant AI Using Python” (2021) International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), Volume 6, Issue 2, June 2021. ISSN (Online): 2581-9429

[6] V Geetha & Gomathy, C K & Kottamasu, Manasa & Kumar, Nukala. (2021). The Voice Enabled Personal Assistant for Pc using Python. International Journal of Engineering and Advanced Technology. 10. 162-165. 10.35940/ijeat.D2425.0410421

[7] Aditya Sinha, Gargi Garg, GouravRajwani, Shimona Tayal, “Intelligent Personal Assistant”, International Journal of Informative Futuristic Research, Volume. 4, Issue 8, April 2017.

[8] Vadaboyina Appalaraju, V Rajesh, K Saikumar, P. Sabitha” Design and Development of Intelligent Voice Personal Assistant using Python” 2021 3rd International Conference on Advances in Computing, Communica- tion Control and Networking (ICACCCN)

[9] Silky Sharma, Prof.(Dr.) Gurinder Singh, “Comparison of Voice Based Virtual Assistants fostering Indian Higher Education – A Technical Perspective”, 2021 International Conference on Technological Advancements and Innovations (ICTAI), November 2021, DOI: 10.1109/ICTAI53825.2021.9673307

[10] Andreas M. Klein, Maria Rauschenberger, Jorg Thomaschewski, and Maria Jos´e Escalona, “Comparing Voice Assistant Risks and Potential with Technology-Based Users: A Study from Germany and Spain”, Journal of Web Engineering, Vol. 20 7, pp 1991–2016, doi: 10.13052/jwe1540-9589.2071, November 2021.