The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

Voice Assistant For Task Automation

P. Vinod[1*], B. Himaja[2], R. Sai Kumari[3]
G. Aparanjini[4]
[1,2,3]Student, Dept of Data Engineering, MVGR College of Engineering, Vizianagaram, India
[4] Assistant Professor, Dept of Data Engineering, MVGR College of Engineering, Vizianagaram, India
Corresponding Author *: palingivinod2004@gmail.com

Abstract

Voice assistants are greatly contributing to the improvement of human-computer interactivity by providing the functionality to interact with computers through natural language interfaces. Nonetheless, the majority of available natural language interfaces are dependent solely on the internet or the commands they are trained to recognize. This raises a number of privacy issues along with an increase in the latency of the system as well as its functionality. This project is based on the concept of the design of a kind of intelligent knowledge-based system that is to serve as a Windows-based intelligent assistant with both online and offline functionality in order to achieve the desired privacy while maintaining the functionality needed from a natural language interface. Moreover, the language interface is based on the concept of intent-based understanding of the natural language commands that the system is trained to receive from the user. The language interface provides the functionality of real-time system automation by enabling the user to effectively manipulate the system settings, access information, support communication functionality, scheduling, power management, System Status, and weather updates.

Keywords:
Hybrid Voice Assistant, Whisper, Large Language Models, Intent Recognition, Windows Automation, Natural Language Processing.

1. Introduction

Virtual assistants have become a major aid in human-computer interaction, where the user can control applications, search the web, send reminders, and access information by simply using speech. Generally, a virtual assistant is a combination of ASR, NLP, and TTS, which helps in understanding the user's intention and acting accordingly. The development of virtual assistants has been marked by a few major milestones. The first major advancements were shown by OpenAI-improved personal assistants, which use sophisticated conversational AI for more natural

conversations, as well as local computation, content creation, and personalized task assistance without complete dependence on cloud infrastructure, ensuring improved privacy and interaction quality [4].Next-generation evolution has increased their potential in the direction of IoT integration and device automation, providing speech command control over external hardware, such as Arduino-based systems for smart-home applications [10]. These sophisticated capabilities are built on earlier, foundational implementations. Python-based desktop assistants are used to illustrate effective task automation using speech commands and external APIs [13]. Subsequently, the virtual assistants developed on the Windows system based on machine learning and natural language processing have also demonstrated the potential to support the normal activities of human usage, such as search engines, weather updates, and note-taking through natural interaction [14]. Overall, the above developments have clearly demonstrated the transition from the simple command-based virtual assistants to a context-based virtual assistance system that can support the normal activities of human usage.

In such a scenario, it becomes imperative to establish and deploy a hybrid concept for a personal assistant so that the overall intent identifying mechanism becomes adaptive in response to resource availability over the network. In such a scenario, the use of online intelligence comes into full play for identifying intent with the help of available internet resources; accordingly, in the case of an offline scenario, the system utilizes local intelligence for identifying the intent without developing any dependency. Accordingly, based upon the intent and parameters, the system becomes capable of executing and performing overall tasks and operations in an autonomous manner.

2. Literature Survey

Virtual assistants have become a major aid in human-computer interaction, where the user can control applications, search the web, send reminders, and access information by simply using speech. Generally, a virtual assistant is a combination of ASR, NLP, and TTS, which helps in understanding the user's intention and acting accordingly. The development of virtual assistants has been marked by a few major milestones. The first major advancements were shown by OpenAI-improved personal assistants, which use sophisticated conversational AI for more natural conversations, as well as local computation, content creation, and personalized task assistance without complete dependence on cloud infrastructure, ensuring improved privacy and interaction quality [4].Next-generation evolution has increased their potential in the direction of IoT integration and device automation, providing speech command control over external hardware, such as Arduino-based systems for smart-home applications [10]. These sophisticated capabilities are built on earlier, foundational implementations. Python-based desktop assistants are used to

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

illustrate effective task automation using speech commands and external APIs [13]. Subsequently, the virtual assistants developed on the Windows system based on machine learning and natural language processing have also demonstrated the potential to support the normal activities of human usage, such as search engines, weather updates, and note-taking through natural interaction [14]. Overall, the above developments have clearly demonstrated the transition from the simple command-based virtual assistants to a context-based virtual assistance system that can support the normal activities of human usage.

In such a scenario, it becomes imperative to establish and deploy a hybrid concept for a personal assistant so that the overall intent identifying mechanism becomes adaptive in response to resource availability over the network. In such a scenario, the use of online intelligence comes into full play for identifying intent with the help of available internet resources; accordingly, in the case of an offline scenario, the system utilizes local intelligence for identifying the intent without developing any dependency. Accordingly, based upon the intent and parameters, the system becomes capable of executing and performing overall tasks and operations in an autonomous manner.

Literature survey

V. Bansal, V. K. Singh, S. Choudhary, and R. Thakur[1], focused on developing a voice assistant using Python. Their assistant is based on the following pipeline: speech recognition, natural language processing, and speech synthesis. They used popular Python libraries such as SpeechRecognition, pyttsx3, NLTK, and spaCy. Most importantly, their paper not only describes the development of the voice assistant but also discusses the challenges that such assistants face in real-life scenarios.. They analyze problems like background noise, different accents, system delays, and user privacy. A key part of their evaluation compares offline versus cloud-based processing, and they point out the clear advantages of doing more intent recognition directly on the device, mainly, it reduces the need for a network. However, it's noted that in their implementation, these offline capabilities are still somewhat limited, as the system remains highly reliant on the functionality provided by its underlying Python libraries.

T. Sasikala, J. J. D. Raj, B. Swathi, A. K., B. Ambati, and A. Lalith [2] introduced an assistant called "Boom," which is a Windows-based assistant to improve accessibility, particularly for the visually impaired. This assistant has functions such as playing music, setting alarms, fetching weather information, and summarizing news through speech recognition, pyttsx3, and BeautifulSoup. The project focuses on offline functionality and modularity for improved reliability

without internet connectivity. The voice assistant still uses basic rule-based matching, lacks deep understanding, intent accuracy analysis, robustness for multiple languages, and adaptive learning.

S. Juneja, Purnima Bakshi, Gursimran Kaur, Rahul Chandra, and Ritesh Kumar Yadav [3], developed LEO: a Python-based personal desktop assistant that can perform tasks like email management, reminders, browsing, internet speed check, and WhatsApp messaging using voice commands. The assistant uses APIs, text-to-speech, and natural language processing libraries and is designed to improve the productivity and accessibility of users, particularly those who are visually impaired. Although it is a comprehensive assistant, it is dependent on internet APIs, does not have hybrid offline-online intent processing, and is not focused on privacy-preserving local computation.

M. Subi, M. Rajeswari, J. J. Rajan, and S. Harshini[5] proposed an AI-based desktop voice assistant (VIZ) integrating OpenAI with speech recognition to perform desktop automation tasks such as browsing, application control, screenshots, system commands, and information retrieval. The assistant uses PyAutoGUI, speech recognition, and TTS to execute user commands and interact with applications. The model highlights privacy and offline capability advantages over cloud assistants. However, it mainly depends on rule-based command matching, lacks robust intent recognition, performance evaluation, and hybrid offline–online learning, and does not address low-resource environments.

Sunil Kumar, Shubham Patel, Sonam, and Vaishnav [6], Srivastav proposed a voice-based virtual assistant for Windows (ZIVA) that executes desktop tasks using speech commands. The system employs AI, NLP, and speech recognition to perform activities such as web search, application control, conversation, email reading, and media handling. Commands are mapped to predefined actions for hands-free interaction. However, the system remains largely cloud-dependent, uses rule-based command matching, and lacks hybrid offline–online intent recognition and contextual understanding.

Sai Surya Vigneshwar M. and Kiran Kumar L[7] developed an AI voice assistant using Python and APIs integrated within an augmented/virtual reality environment. Their system allows users to issue voice commands through wearable interfaces such as smart glasses and receive assistance as AR overlays. The work focuses on voice interaction in immersive environments, combining NLP and machine-learning algorithms for command interpretation. However, the system mainly emphasizes AR interaction design rather than desktop task automation, provides limited discussion of offline capability, and lacks an intent-classification mechanism optimized for resource-constrained devices

Aditya Bhardwaj, et al. [8], introduced a desktop AI assistant that automates tasks such as web searching, news extraction, emailing, weather reporting, and interacting with applications like YouTube and WhatsApp via speech commands. The system employs natural language processing

(NLP), speech recognition, and Python automation modules to interpret and execute user instructions. Nevertheless, this assistant remains highly dependent on internet connectivity, offers limited personalization and contextual awareness, lacks offline functionality and model efficiency, and does not incorporate privacy-aware on-device processing.

S. V. S. Gunasekara, V. P. T. Madushan, and B. Fernando[9] developed a complete, AI-based, Windows-based virtual assistant(Gypsy). This assistant is voice and text-responsive and it is also capable of performing a lot of tasks. Users are able to tell it to search the web, start applications, or to send emails and messages, read notes, check the weather, set alarms or to even make online orders. It contains a Speech Recognition powered by Google to translate speech to text, a collection of Python libraries to execute applications, and a graphical interface developed in tkinter. It has a fundamental component to its intelligence through a neural network that has been trained on its own custom JSON data that enables it to produce a broader range of contextual and individualized answers. It is a strong system, but one of the most notable things about it is that it is highly reliant on a stable internet connection to perform most of its functions.

S. Uke, H. Lokhande, D. Lohar, D. Lathiya, A. Langhe, T. Lautawar, and P. Likhitkar[11]. Designed an assistant called FRIDAY, which is a Python-powered virtual voice assistant that allows booking a taxi or a ticket, and so forth. emails, playing media, weather reporting, and simple operations of the system. The system uses speech recognition, TTS, and Python APIs to aid users such as the elderly user and the visually impaired user. persons. Even though it enhances accessibility, it is very internet-reliant and predetermined. command structures, deficient in context awareness or understanding and smart intent recognition, and are not. Mechanisms of support for offline fallback.

Divisha Pandey et al. [12],developed a Python and artificial Windows-based voice assistant. intelligence to facilitate day-to-day operations like email management, alarms, web browsing, note. management, application control, and elementary conversational interaction. The work discusses the adoption of machine learning and NLP concepts and can be described as having many learning modes of task. execution. Nevertheless, the system is cloud-based and does not provide detailed intent analysis. recognition accuracy, and does not have a hybrid off-line-online architecture.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

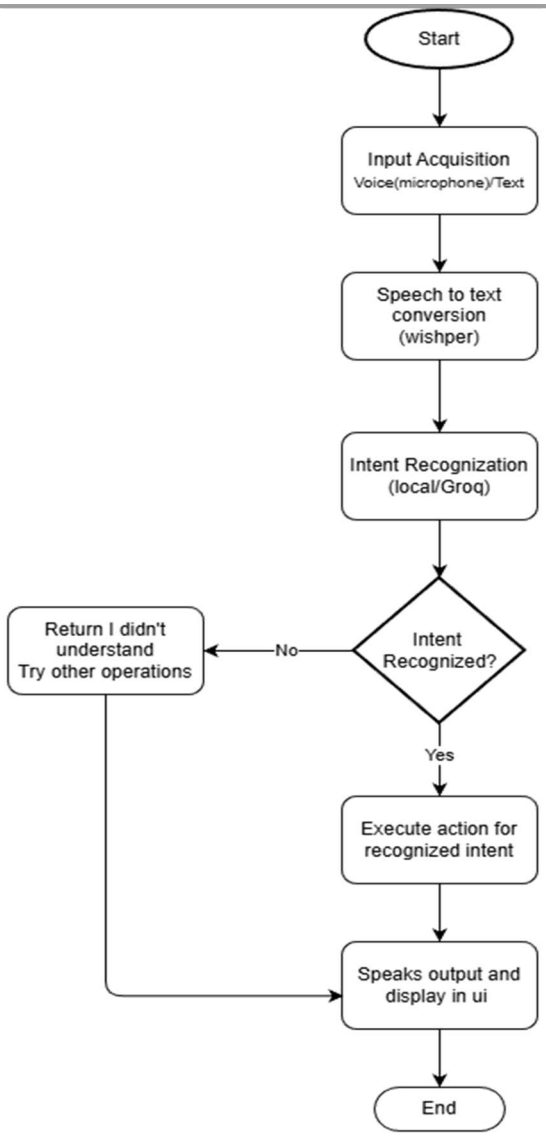
R. Paul and N. Mukhopadhyay [15] developed a Python-based desktop voice assistant system that was expected to offer hands-free functionality by enabling the system to open applications, do web searches, manage emails, take notes, media control and weather checks. The proposed system is a hybrid of speech to text, text to speech and Python automation libraries. However, such a developed voice assistant system relies primarily on predefined commands, does not have more complex intent recognition and learning capabilities, cannot do hybrid offline-online processing, and cannot operate in low-connectivity or privacy-sensitive conditions.

3. Methodology

The proposed system is based on an intent-driven architecture where the input given by the user is first processed to create structured meaning and then carried out as actions of the system. The methodology includes the following steps:

1. **Input Acquisition**
   To execute the command, initially, input is taken from the user through voice or text commands
2. **Convert voice into text**
   If the input is given through voice, then the voice commands are converted into text using whisper locally
3. **Intent Extraction**
   The converted text (or direct text input) is transmitted to a locally executed Large Language Model (LLM) through Ollama, or Online Groq if internet connectivity is available
   The LLM processes the text and retrieves:
   - Intent(eg, reduce voice or brightness etc)
   - Required parameters (slots)
4. **Decision Making**
   Once the intent is recognized, the system identifies the corresponding action module. If the intent is not recognized, it returns I didn’t understand that. Try other commands.
5. **Action Execution**
   Each execution module performs tasks based on the decision, like increasing volume, brightness, weather information retrival etc

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48



Algorithm:

The below algorithm explains how the assistant is actually working. Initially, the user needs to initialize the system, and then it takes the input from the user through voice by microphone and text through the text input chat interface. If the input is voice, then it is converted to text. Converted text data is sent to llm to identify the intention, like reducing voice, sending emails, etc. Once the intent is identified then particular action is executed and sends the output to ui it shows the output, logs to the user, and speaks out the output. If intent is not recognized it sends a fallback message. I didn’t understand that

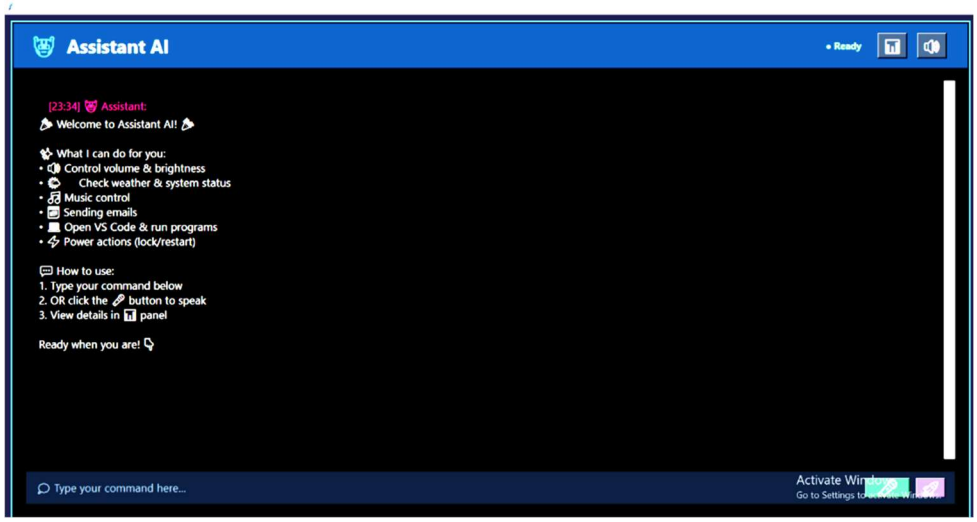
The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

1. *START*
2. *INITIALIZE system*
   INITIALIZE GUI interface
3. *WHILE system is active DO*
   WAIT for user input
4. *IF input is voice THEN*
   CAPTURE audio from the microphone
5. *CONVERT speech to text*
   ELSE
6. *READ text from GUI*
7. *END IF*
   PREPROCESS input text
8. *EXTRACT intent and parameters from text*
9. *SELECT action based on intent*
   IF action is valid THEN
10. *EXECUTE corresponding system operation*
    STORE execution result
11. *ELSE*
    SET response as "I didn't understand that"
12. *END IF*
    GENERATE response message
13. *SPEAK response to user*
    DISPLAY response and logs in the GUI
14. *END WHILE*
15. *STOP*

5. Results

Below are the results of the Assistant doing tasks

1. User Interface



The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

Fig 1. Assistant User Interface

Figure 1 shows the user interface of the assistant. The user can interact with the assistant using either text or voice commands.

2. Email Sending operations

Fig 2. Given the assistant voice command to send an email to the default, it sent

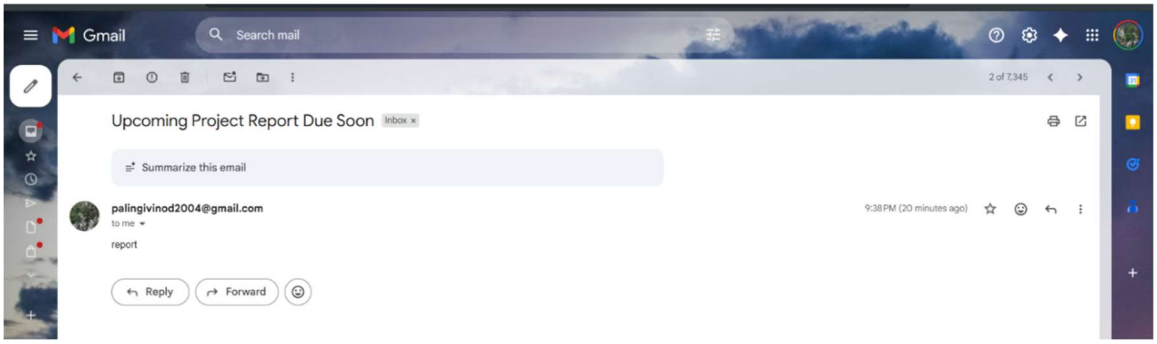


Fig 3. Received an email that was sent from the assistant.

Figure 2 shows that the user gives a voice command to send an email to the default email address with the message project report assistant understands the command and sends an email with a generated subject using LLM. Figure 3 shows the received email sent by the assistant with the generated subject Upcoming Project Report Due Soon.

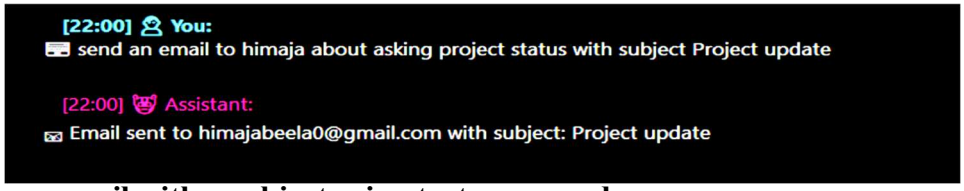


Fig 4. Sending an email with a subject using text commands

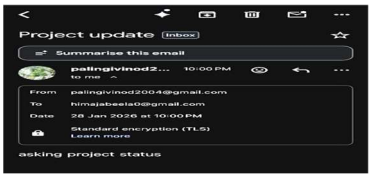


Fig 5. Received an email with the given subject.

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

Figure 4 shows that the user sends an email to someone with a subject using text commands. Figure 5 shows the received email with the given subject and message.

3. Weather Information

Fig 6. Displayi

Figure 6 shows the temperature in Mumbai based on user voice command.

4. VS Code Operations

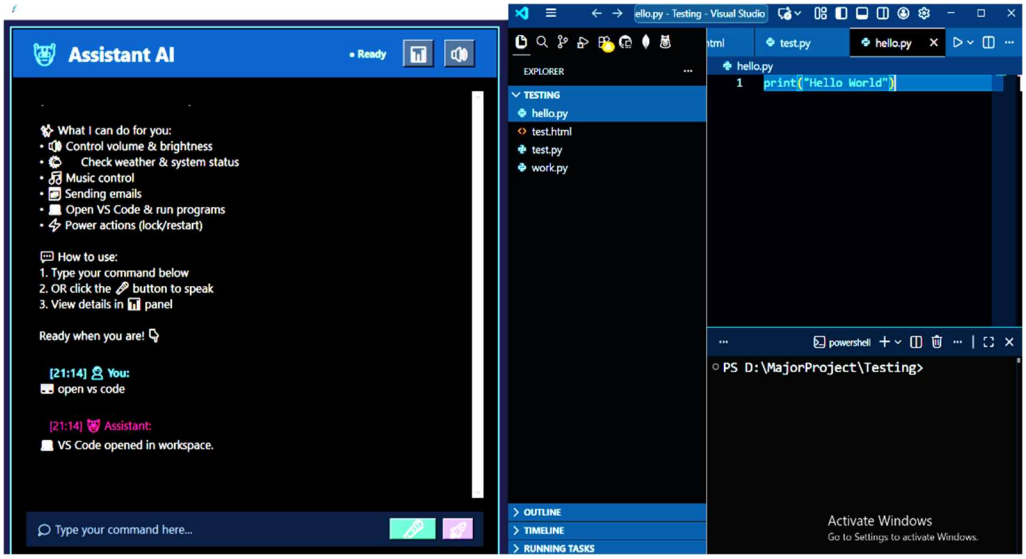
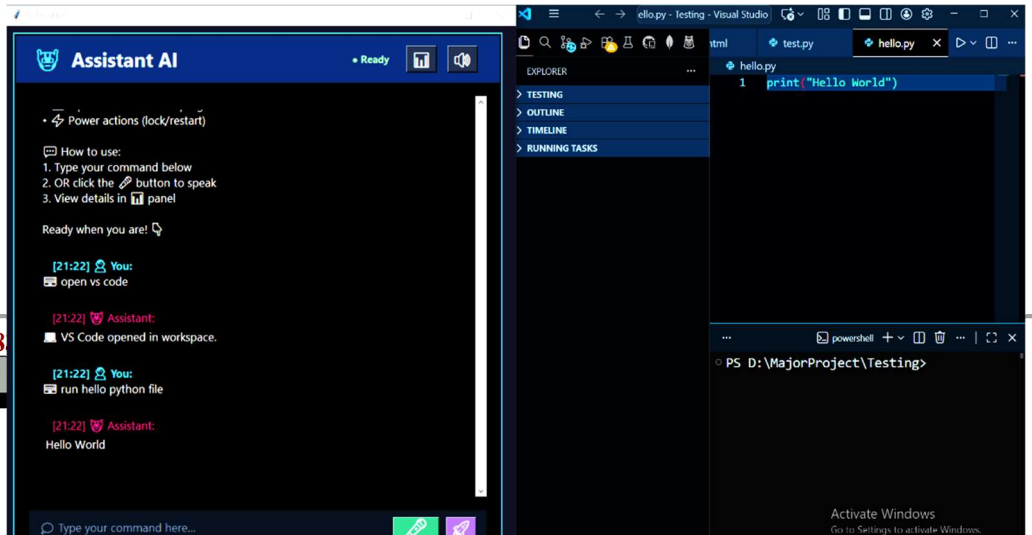


Fig 6. Opening VS Code

Figure 6 shows that VS Code opened in the default folder Testing based on the user text command.



The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

Fig 7. Running the code files.

Figure 7 shows that the assistant is running the Python file and showing the output in the ui.

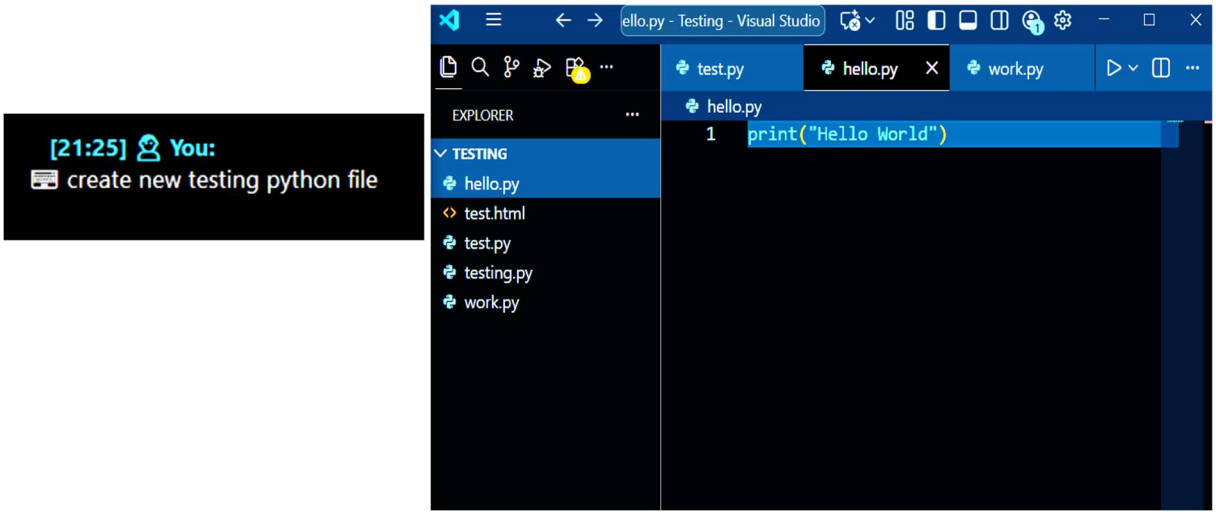
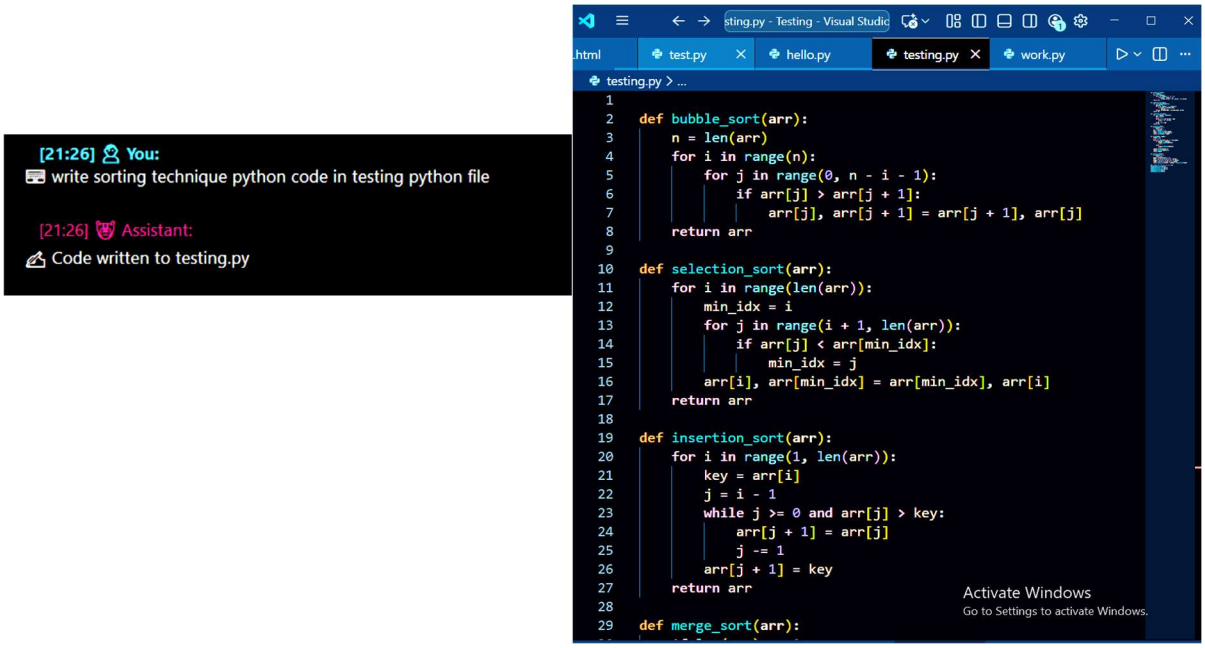


Fig 8. Creating a new Python file with the name testing.

In Figure 8 user sends a command to the assistant to create python file with the name testing. The assistant created the Python file with the given name.



The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

Fig 9. Writing Python code for sorting in the created testing file.

Figure 9 shows that the user send an command to the assistant to write Python code for sorting assistant writes bubble sort code in a testing Python file.

5.1 Security Guardrails

The assistant has three safety mechanisms to avoid accidental deletion of the files or any unauthorized activities. To begin with, command whitelisting prevents destructive actions such as deletion of files, editing of registries or system files during the intent extraction phase. Second, there should be explicit confirmation dialogues that the user must approve explicitly (voice or text yes) prior to performing any state-modifying operation like overwriting files or exiting applications. Third, parameter validation only allows file paths to user writable directories (like C:Users... Documents) and blocks the access to system directories such as C:Windows. These guardrails will make sure that voice misinterpretation, or LLM errors will not result in harmful operations without the explicit consent of a user.

6. Evaluation

Based on performance testing, the Voice Assistant for Task Automation system is:

1. Accurate in recognizing voice commands
2. Efficient in understanding user intent
3. Reliable in executing automation tasks
4. Stable in offline environments

A series of 10 experimental trials is used to evaluate the performance of the system by measuring the latency of each stage within the pipeline to ascertain statistical consistency. The Speech Recognition module (Whisper) had a constant average latency of about 4.68 seconds across these runs. One of the main areas of interest of the experiment was the comparison of LLM response times, as the online LLM (Groq) had a much lower mean response time of about 1.08 seconds. Conversely, the offline LLM took an average of 39.73 seconds and the first trial of each set had an extreme delay because of the overhead of loading the model in memory.

Latency Analysis on the Components and the overall time to perform the task in the online mode.

Table 1: Overall latency of assistance in online mode

The Journal of Computational Science and Engineering (TJCSE)
ISSN 2583-9055 (Media Online)
Vol 4 No 2, Feb 2026
PP 34– 48

Component	Time(seconds)
Whisper(STT)	4.68
LLM(Groq)	1.08
TTS	2.73
App Execution	1.04
Total Response Time	9.53

Latency Analysis of the Components and total time for executing the task during offline mode

Table 2: overall latency of assistance in offline mode

Component	Time(seconds)
Whisper(STT)	4.68
LLM(local llama)	39.73
TTS	2.73
App Execution	1.04
Total Response Time	~48-50

Offline performance of the suggested assistant system is very much related to the hardware specifications of the system. The offline LLM can use all available local computing capabilities, including CPU or low-capacity GPUs, unlike online models which can use high-performance cloud-based GPUs. This leads to a very high latency in the intent extraction part when offline. Empirical results of indicate that the significant lag in the pipeline is only due to the local LLM inference, and all other components, such as speech recognition, action execution, and text-to-speech response, do not have significantly different performance than those of the online setup.

7. Conclusion

A hybrid intelligent assistant for Windows systems has been designed and successfully implemented to enable natural and efficient human-computer interaction via voice and text inputs. The assistant adopts an intent-centric design paradigm, in which the speech-to-text task is completely performed offline using the Whisper model, and the intent extraction task is performed in a hybrid fashion, utilizing an online llm model if internet connectivity is available and seamlessly transitioning to a local llm model in offline scenarios. This is to enable human inputs to be interpreted appropriately, and desktop automation activities such as email communications, taking weather information, managing applications, working on files, and system-level activities can be undertaken successfully. The system exploits the strengths of online intelligence capabilities and the power of an offline fallback solution and has enhanced natural language,

flexible, reduced latency, better privacy, and smooth operation even under low-connectivity or offline conditions. The handiness of the assistant in practical applications is confirmed by experimental results that have effectively overcome major weaknesses experienced in the traditional Windows-based assistant, which exclusively uses the cloud infrastructure or command lines.

8. References

1. Bansal, V., et al. (2025). Implementation and design of a Python-based voice assistant for seamless user interaction. International Journal of Novel Research and Development (IJNRD).
2. Sasikala, T., Raj, J. J. D., Swathi, B., K., A., Ambati, B., & Lalith, A. (2025). Enhancing accessibility through AI: Design and functionality of a Windows voice assistant. In Proc. ICSCSA 2025 (pp. 673–678).
3. Juneja, S., Bakshi, P., Kaur, G., Chandra, R., & Yadav, R. K. (2025). Leo: Personal desktop assistant using Python. In Proc. 2025 3rd International Conference on Disruptive Technologies (ICDT) (pp. 1508–1512).
4. Benny, R., Muralidharan, A., & Subramanian, M. (2024). OpenAI-enhanced personal desktop assistant: A revolution in human-computer interaction. In Proc. ICETITE 2024 (pp. 1–7).
5. Subi, M., Rajeswari, M., Rajan, J. J., & Harshini, S. (2024). AI-based desktop VIZ: A voice-activated personal assistant – Futuristic and sustainable technology. In Proc. ICCSP 2024 (pp. 1095–1100).
6. Kumar, S., Patel, S., Sonam, & Srivastav, V. (2024). Voice-based virtual assistant for Windows (ZIVA – AI companion). In Proc. 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT) (pp. 960–965).
7. M., S. S. V., & L., K. K. (2024). AI voice assistant using Python and API. In Proc. 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS) (pp. 1–6).
8. Bhardwaj, A., Singh, D. P., & Sahu, H. (2024). Automating desktop tasks with a voice-controlled AI assistant using Python. International Journal of Research Publication and Reviews, 5(5), 12615–12620.
9. Madushan, V. P. T., Gunasekara, S. V. S., & Fernando, B. (2023). Gypsy: AI-powered virtual assistant for Windows OS. CINEC Academic Journal, 6(2), 81–86.
10. Akash, A. S., Jayaram, N., & Jesudoss, J. A. (2022). Desktop-based smart voice assistant using Python language integrated with Arduino. In Proc. ICICCS 2022 (pp. 374–380).
11. Uke, S., Lokhande, H., Lohar, D., Lathiya, D., Langhe, A., Lautawar, T., & Likhitkar, P. (2022). Virtual voice assistant in Python (FRIDAY). In Proc. ICCCMLA 2022 (pp. 164–168).
12. Pandey, D., Ali, A., Dubey, S., Srivastava, M., Dwivedi, S., & Raza, M. S. (2022). Voice assistant using Python and AI. International Research Journal of Engineering and Technology (IRJET), 9(5), 832–838.
13. Geetha, V., et al. (2021). The voice enabled personal assistant for PC using Python. International Journal of Engineering and Advanced Technology (IJEAT), 10(4).

14. Aggarwal, A., Goel, A., Sharma, M. L., Sengar, N., & Bahl, V. (2021). Virtual personal assistant for Windows. International Research Journal of Modernization in Engineering, Technology and Science (IRJMETS), *3*(12), 1030–1035.

15. Paul, R., & Mukhopadhyay, N. (2021). A novel Python-based voice assistance system for reducing the hardware dependency of modern age physical servers. International Research Journal of Engineering and Technology (IRJET), 8(5), 1425–1431.