INTERNATIONAL JOURNAL OF LATEST TECHNOLOGY IN ENGINEERING, MANAGEMENT & APPLIED SCIENCE (IJLTEMAS)

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

An Integrated Model for A Virtual Voice Assistant Using Modern Artificial Intelligence Technologies

Ahmed Salam AL Amour, Dr. G. Sandhya Devi

Computer Science & Systems Engineering Andhra University.

DOI: https://doi.org/10.51583/IJLTEMAS.2025.1406000123

Abstract: This paper presents the comprehensive development of a desktop-based virtual assistant application that leverages advanced speech recognition and natural language processing (NLP) technologies. The system is developed using Python programming language and is seamlessly integrated with Google Gemini to enhance performance and understanding of user intent. The assistant provides a user-friendly conversational interface for executing a variety of system-level tasks, such as launching desktop applications, recording the screen, retrieving general information, and automating repetitive commands. The primary objective of this project is to significantly enhance user productivity and improve overall system accessibility, especially through intuitive voice-based interaction. In today's world, many visually impaired, physically disabled, and elderly individuals suffer from social isolation and a diminished sense of independence in their everyday lives. Voice assistant systems provide a highly promising solution to this challenge by enabling hands-free, natural, and intuitive interaction with digital technology. This approach allows users to carry out essential tasks, access necessary information, and communicate with others without the need for visual cues or manual input. The proposed project introduces a Python-based voice assistant system specifically designed with the needs of blind, elderly, and physically challenged individuals in mind. The system is tailored to improve their quality of life by promoting continuous engagement, enhancing digital accessibility, and encouraging greater independence. By integrating both speech recognition and text-to-speech capabilities, the assistant can understand verbal commands, respond with synthesized speech, and perform vital functions such as setting alarms or reminders, reading incoming messages, providing weather or news updates, and accessing online content. This work demonstrates the transformative potential of AI-powered voice technologies in fostering inclusivity, supporting vulnerable populations, and empowering individuals with special needs through smart, accessible digital interaction.

Keywords : Virtual Assistant, Speech Recognition, Natural Language Processing (NLP), Text-to-Speech

I. Introduction

In today's era, nearly all tasks have become digitalized. With smartphones in hand, it feels as though we have the world at our fingertips. Increasingly, we no longer even need to use our fingers; we simply speak a command, and it is executed. There are systems available where one can say, "Text Dad, 'I'll be late today,'" and the message is sent automatically. This is the role of a Virtual Assistant. These systems also support specialized tasks such as booking a flight or finding the cheapest book online across various e-commerce platforms and then providing an interface to place the order. They help automate search, discovery, and online ordering processes. European population is currently 65 years or older and predicts that this percentage will increase to 23.8% by 2030. In addition, most older adults wish to stay in their homes and to age in place. These facts have led to more and more researchers aiming to improve the ageing process in the older adult home and ease the access to Information and Communications Technologies (ICT) by this population **Error! Reference source not found.** (Valera Román A, 2021 Apr 20) . In this new age of technology and innovation, the use of artificial intelligence and machine learning has made our life much easier. Virtual Assistants process audio signals, convert them to text, and perform tasks using Speech-To-Text modules, parsers, dialog managers, answer generators, and speech synthesizers (Patil J, 2021 May). Technological advancements have simplified daily tasks through oral communication with computers via conversational interfaces (CIs) (Abougarair AJ, 2022). AI technology interacts with humans both directly and indirectly. A clear example of human-AI interaction is the use of chatbots, which enhance service quality and customer engagement by enabling personalized communication without the need for human agents. Similarly, voice assistants have brought significant benefits and transformative impacts to daily life, allowing users to issue voice commands and contributing to improved customer satisfaction. AI-powered recommendation systems also play a crucial role in guiding user choices by analyzing preferences and behaviors, thereby offering tailored suggestions that enhance user experience and drive engagement. Virtual assistants' usability and efficacy are greatly influenced by natural language processing (NLP). The manner in which people engage with technology is being revolutionized by these AI-powered assistants' ability to understand, interpret, and reply to human language thanks to NLP (Entertainment:, 2023 Dec 7).

Google Gemini is integrated into the virtual assistant system through its API to provide intelligent and advanced natural language understanding. After the user's speech is converted into text using speech recognition tools, this text is sent to the Gemini model via the google-generativeai Python library, using an API key obtained from Google Cloud. The model analyzes the text, understands the user's intent, and then sends a response that clarifies the intended command. The assistant then uses this response to perform the desired action, such as opening an app or providing an answer. This integration enables the system to understand complex or ambiguous commands accurately and smoothly.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

However, this paper aims to provide an overview of the methodologies and steps involved in making a Virtual Personal Assistant, considering different research results and limitations. Technological advancements have simplified daily tasks through oral communication with computers via conversational interfaces (CIs).

Motivation

With the accelerating pace of digital transformation efficiency and ease of use have become critical factors in the design of everyday computing tools, especially with the complexity of tasks and the multitude of applications required by the average user. Therefore, this virtual assistant was developed using a modular approach based on the Python programming language. Several specialized speech recognition libraries, such as Whisper and Speech Recognition were integrated to convert the user's voice commands into text, enabling natural interaction with the system without the need for manual input. Understanding the user's intent and parsing their commands was accomplished using the Google Gemini language model which is capable of extracting intents "Intent Detection" and accurately identifying key elements in commands "Slot Filling".

To facilitate the execution of commands, the assistant was linked to application interfaces at the operating system level using libraries such as os pyautogui and pywinauto. This enabled it to open common applications such as Notepad Calculator Internet Explorer and Microsoft Office as well as perform actions such as taking screenshots and videos. An interactive graphical interface (GUI) was also designed using PyQt or Tkinter, allowing users to visually monitor commands and alerts in realtime, improving the user experience and reducing reliance on traditional manual interaction. These technical solutions respond to the challenges faced by users in the modern computing environment, where many daily tasks require cumbersome repetitions of clicks and commands, leading to decreased productivity and distraction. This difficulty is exacerbated for nonexpert users or those lacking sufficient technical expertise making the need for natural and optimized interactive interfaces even more crucial.

From a research perspective, additional challenges arise in measuring the effectiveness of these systems in understanding natural language. Current language models struggle to capture the nuances of speech context or handle ambiguity in commands. Furthermore, traditional evaluation tools do not always reflect the ability of intelligent assistants to handle complex real-world scenarios. Therefore, the need to develop compact (distilled) language models that maintain the interpretability of large models, but with fewer resources, has emerged, making them suitable for mobile devices and limited systems. This requires advanced knowledge distillation techniques, along with new assessment methodologies that take into account contextual and functional accuracy in language comprehension. (Abougarair AJ, 2022)

II. Objectives of The Proposed System

The proposed voice-activated desktop assistant is designed to revolutionize the way users—particularly those who are visually impaired, elderly, or physically challenged—interact with their computers. (Shashikala KS, 2025; Masina F, 2020 Sep 25) By enabling voice-controlled execution of routine tasks such as opening commonly used applications (e.g., Notepad, Calculator, Word, Excel, PowerPoint), taking screenshots, and recording screens, the system significantly enhances productivity and reduces dependence on traditional input devices like the keyboard and mouse. Through a natural, conversational interface powered by speech recognition and natural language processing, the system creates a user-friendly environment that lowers the technological barrier for individuals who may lack technical proficiency. One of the core features is a real-time transcription interface that displays voice interactions, allowing users to monitor and verify commands as they are processed—ensuring transparency and reinforcing trust in the system. Moreover, the assistant is designed to learn from user behavior over time, gradually adapting to individual preferences and offering personalized responses and suggestions that align with the user's habits and workflow. Seamless integration with the desktop operating system ensures fast, efficient task execution, making the assistant a reliable companion for daily computing needs. Beyond functional support, the system also facilitates casual, human-like conversations, enhancing emotional engagement and reducing the sense of isolation often felt by users with physical or visual limitations. By automating repetitive actions and minimizing the cognitive effort required to perform standard tasks, the assistant enables users to focus on more meaningful activities. Additionally, the system promotes a fully hands-free computing experience, especially useful in contexts where manual interaction is difficult or impractical. Importantly, strong emphasis is placed on data privacy and security, ensuring that all interactions—especially those involving sensitive information—are handled responsibly and securely. In sum, this system represents a comprehensive, intelligent solution aimed at fostering independence, inclusivity, and digital empowerment through the power of voice technology.

Related Work

Virtual assistants such as Siri (Apple), Alexa (Amazon), and Cortana (Microsoft) have become prominent tools for voice-based interaction, offering users the ability to perform simple tasks such as setting reminders, playing music, checking the weather, or controlling smart home devices. (Hoy MB. Alexa, 2018 Jan 2) These systems rely heavily on cloud-based natural language processing and are designed primarily for mobile and smart home environments. While convenient for casual tasks, their usefulness in complex desktop workflows or professional environments remains limited. One major shortcoming of these systems is their lack of deep integration with desktop operating systems. Most assistants are not capable of navigating the file system, controlling multiple applications simultaneously, or managing context-rich tasks like editing documents, scheduling across different platforms, or providing real-time feedback based on user behavior within software environments. When desktop integration does exist, it is

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

often restricted to a few predefined commands, making the interaction rigid and task-specific rather than adaptive or conversational. Furthermore, real-time GUI-based transcription is either non-existent or highly limited in current systems. Users have minimal visual feedback during interactions, which can lead to misunderstandings or errors, especially when issuing complex or ambiguous commands. Without an intuitive graphical interface, these assistants fail to offer the transparency and control users expect from desktop tools. Additionally, conversational depth remains a challenge. Existing assistants struggle with maintaining context over extended dialogues and often require users to repeat or rephrase commands (Wired, 2023 , October 5). This lack of continuity interrupts the flow of interaction and prevents the formation of a more natural, human-like communication experience. Academic research has attempted to address some of these challenges by proposing multimodal systems that combine voice, gesture, and visual feedback. However, such approaches often remain at the prototype stage and are not widely adopted in consumer or enterprise environments. This project builds upon these limitations by developing an intelligent assistant that emphasizes desktop integration, real-time GUI-based transcription, and extended conversational capabilities. The proposed system is designed to support a wider range of tasks, provide visual context, and sustain dynamic interactions with users in a more natural and productive manner.

Proposed System

The desktop virtual assistant represents a significant step forward in intelligent personal productivity tools, combining the latest advancements in artificial intelligence, natural language processing (NLP), and speech technologies. Developed primarily in Python, the assistant is engineered to integrate deeply with the Windows operating system, ensuring reliable interaction with key applications such as Notepad, Calculator, Microsoft Word, Excel, PowerPoint, and multiple web browsers. At its foundation, the system uses advanced natural language understanding (NLU) models to parse user inputs, accurately detect intents, and extract slot values, thereby allowing users to communicate with the assistant using natural, conversational language. Voice commands are handled through a sophisticated speech recognition engine that converts spoken input into text with high accuracy, enabling hands-free operation and expanding accessibility for users with physical limitations. In parallel, a responsive text-to-speech system generates clear verbal responses, creating a smooth two-way conversational experience. To further enhance transparency and control, the assistant features a modern, interactive graphical user interface (GUI) developed using frameworks like PyQt or Tkinter. This interface includes a real-time transcription panel that displays both user inputs and assistant responses, helping users monitor system behavior, catch misinterpretations, and interact more confidently. The assistant is not only reactive but context-aware, designed to handle multi-step interactions by maintaining short-term conversational memory and offering follow-up suggestions. Built with modularity in mind, the architecture supports easy extension, allowing developers to incorporate additional services such as email automation, calendar scheduling, summarization of documents, file system navigation, and integration with cloud-based productivity platforms. Looking ahead, the platform can be augmented with personalization features driven by machine learning, enabling the assistant to adapt to individual usage patterns, preferences, and task histories over time. By uniting intelligent language processing with a real-time, user-friendly interface, this virtual assistant not only streamlines daily computing tasks but also sets a foundation for more natural and effective human-computer interaction on the desktop.

Methodology

The virtual assistant was developed using a modular approach with Python as the primary programming language, leveraging various libraries for speech recognition, natural language understanding (NLU), and graphical user interface (GUI) development. For speech recognition, Python libraries such as Speech Recognition or whisper were employed to convert voice inputs into text, enabling voice-based interaction. Natural language understanding was powered by Google Gemini, a large language model, which processes and interprets user commands, extracting intent and identifying key slots to understand complex requests. To facilitate seamless interaction with the system, the assistant uses OS-level APIs to interface with native desktop applications such as Notepad, Calculator, and Microsoft Office tools. These APIs, accessed through Python's os module, pyautogui, and pywinauto, allow the assistant to execute commands, open files, or control other applications based on the interpreted user input. Additionally, the assistant features a modern GUI, built with frameworks like PyQt or Tkinter, which provides real-time feedback and transcription, enabling users to track their interactions and commands visually. The system's architecture and interaction flow were modeled using use case diagrams**,** data flow diagrams (DFDs)**,** and system models**,** which provided clear visual representations of user-system interactions, data movement, and module relationships. This structured approach guided the development process, ensuring the assistant was both efficient and scalable, while allowing for future enhancements and integrations.

Finalized Architecture of The Model

The user provides a voice input, which is captured and processed by the Speech Recognition Module. This module converts the spoken language into text form. The recognized text is then sent to the Python Backend, which acts as the system's brain.The backend analyzes the input and determines what action needs to be taken. Depending on the user's request, the system can either make an API call to fetch external information, perform Content Extraction to analyze or retrieve specific data from the text, or issue a System Call to perform an action directly on the computer (like opening an app). After processing, the result is forwarded to the Text to Speech Module, which converts the response into audio form. Finally, the system produces an output voice response that the user can hear, completing the interaction in a natural and human-like way.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

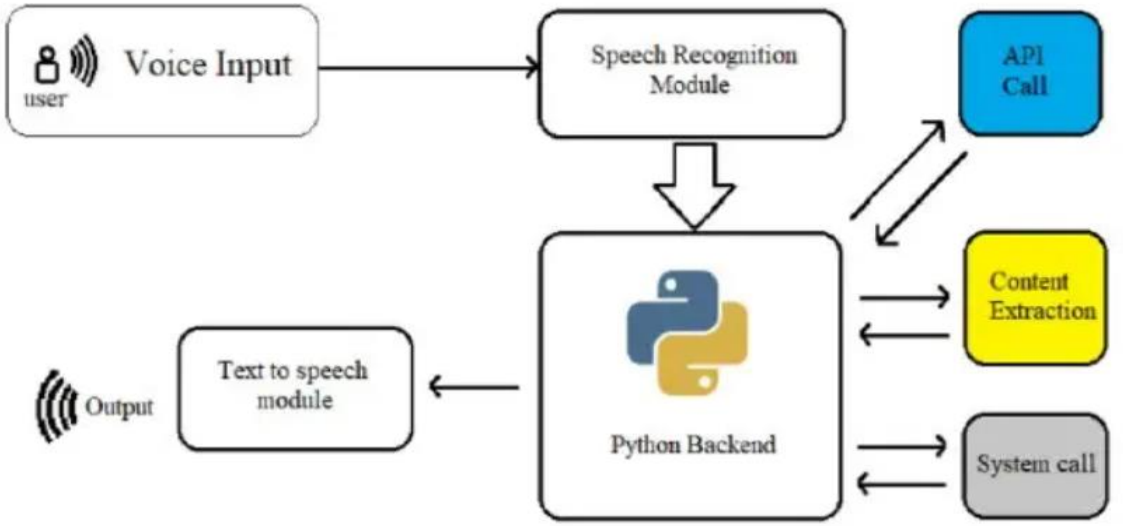

Figure 1. Architecture of the model

Source: Created by authors

System Design and Block Diagram

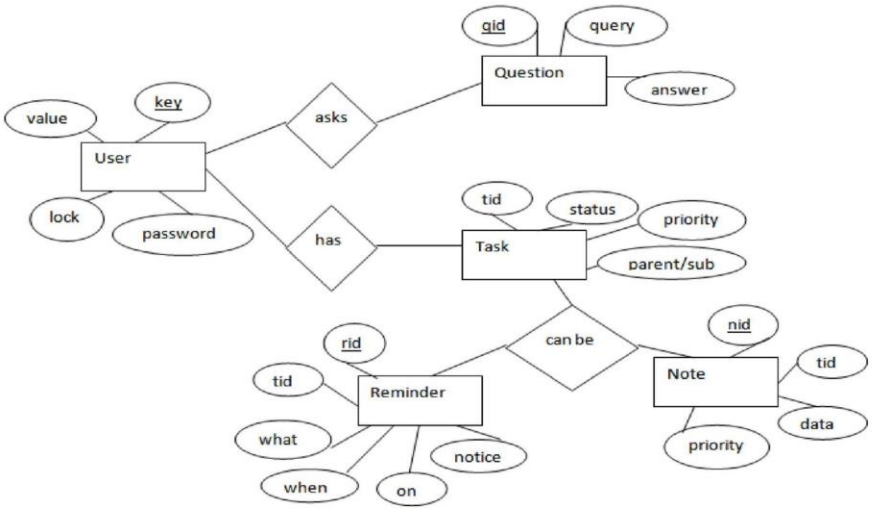

Figure 2. ER Diagram

Source: Created by authors

The above diagram shows entities and their relationship for a virtual assistant system. We have a user of a system who can have their keys and values. It can be used to store any information about the user. Say, for key “name” value can be “Jim”. For some keys user might like to keep secure. There he can enable lock and set a password (voice clip). Single user can ask multiple questions. Each question will be given ID to get recognized along with the query and its corresponding answer. User can also be having n number of tasks. These should have their own unique id and status i.e. their current state. A task should also have a priority value and its category whether it is a parent task or child task of an older task. The Entity-Relationship Diagram (ERD) is a widely adopted tool in structured analysis and conceptual data modeling. It provides an intuitive and effective means of representing real-world systems by modeling business entities, the relationships among them, and the attributes that describe their properties. The ER approach is valued for its clarity, expressiveness, and its ability to be easily transformed into a relational database schema. Common semantic components of ER modeling include cardinality constraints, participation (optional or mandatory involvement in relationships), and generalization/specialization hierarchies, which further enhance its descriptive power (10.Song IY, 1995) .Additionally, the main components of ER model are entity set, relationship set and integrity constraints. Entity set shows the objects in the real world which are different from other objects (Mohammed MA, 2015 Oct).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

Activity Diagram

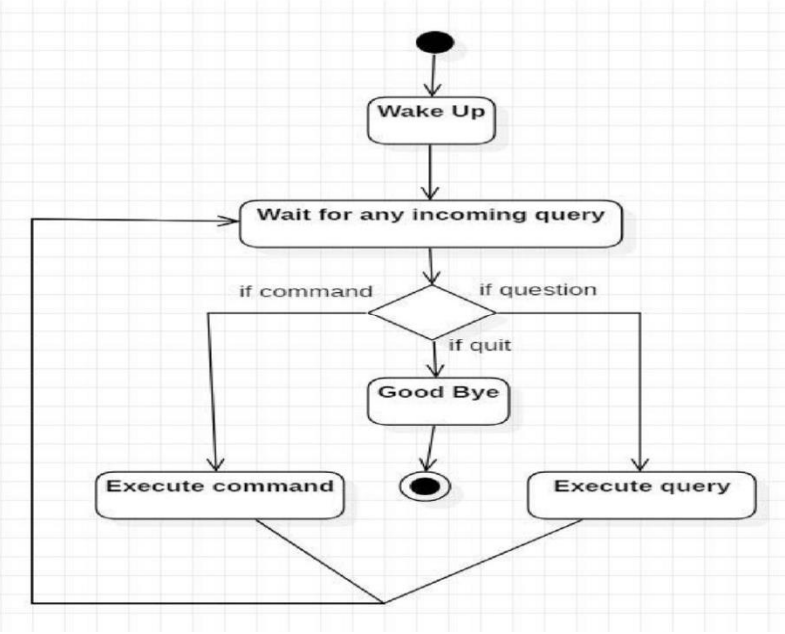

Figure 3. Sequence diagram for Query-Response

Source: Created by authors

Initially, the system is in idle mode. As it receives any wake up cal it begins execution. The received command is identified whether it is a questionnaire or a task to be performed. Specific action is taken accordingly. After the Question is being answered or the task is being performed, the system waits for another command. This loop continues unless it receives quit command. At that moment, it goes back to sleep.

Sequence Diagram

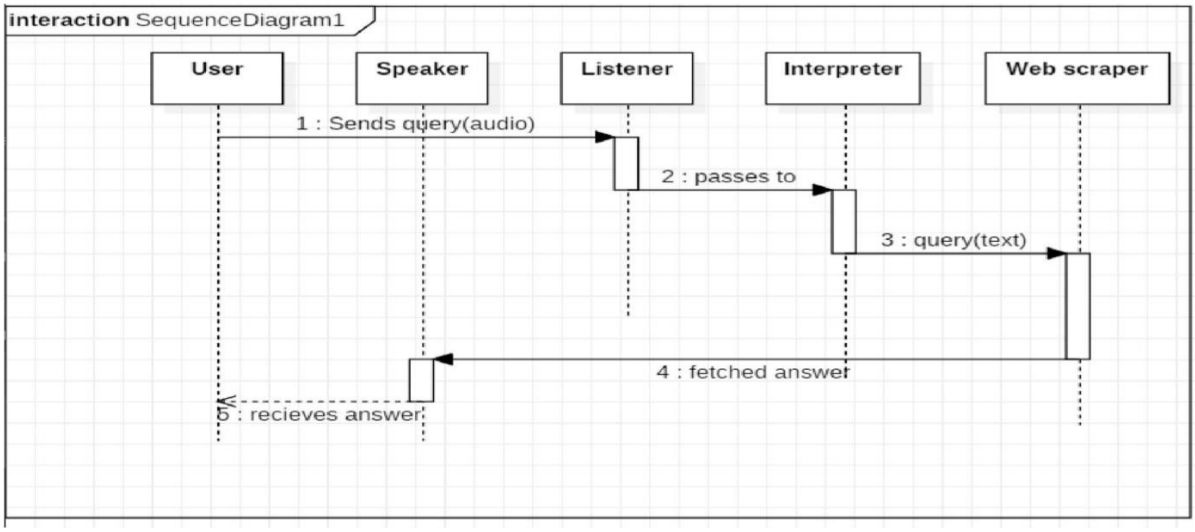

Figure 4. Sequence diagram for Query-Response

Source: Created by authors

The above sequence diagram shows how an answer asked by the user is being fetched from internet. The audio query is interpreted and sent to Web scraper. The web scraper searches and finds the answer. It is then sent back to speaker, where it speaks the answer to user. This is followed by analyzing classes with the intent of encapsulation (bundling data and methods) while still keeping data and operations separate. Then, the analysis moves to the task of specifying operations, which define the behavior of objects, including the communication that occurs between objects by passing messages to one another (S, 2021 May 31.). An interaction is a sequence of messages passed between objects to accomplish a particular task (Swain SK, 2010 Jul).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

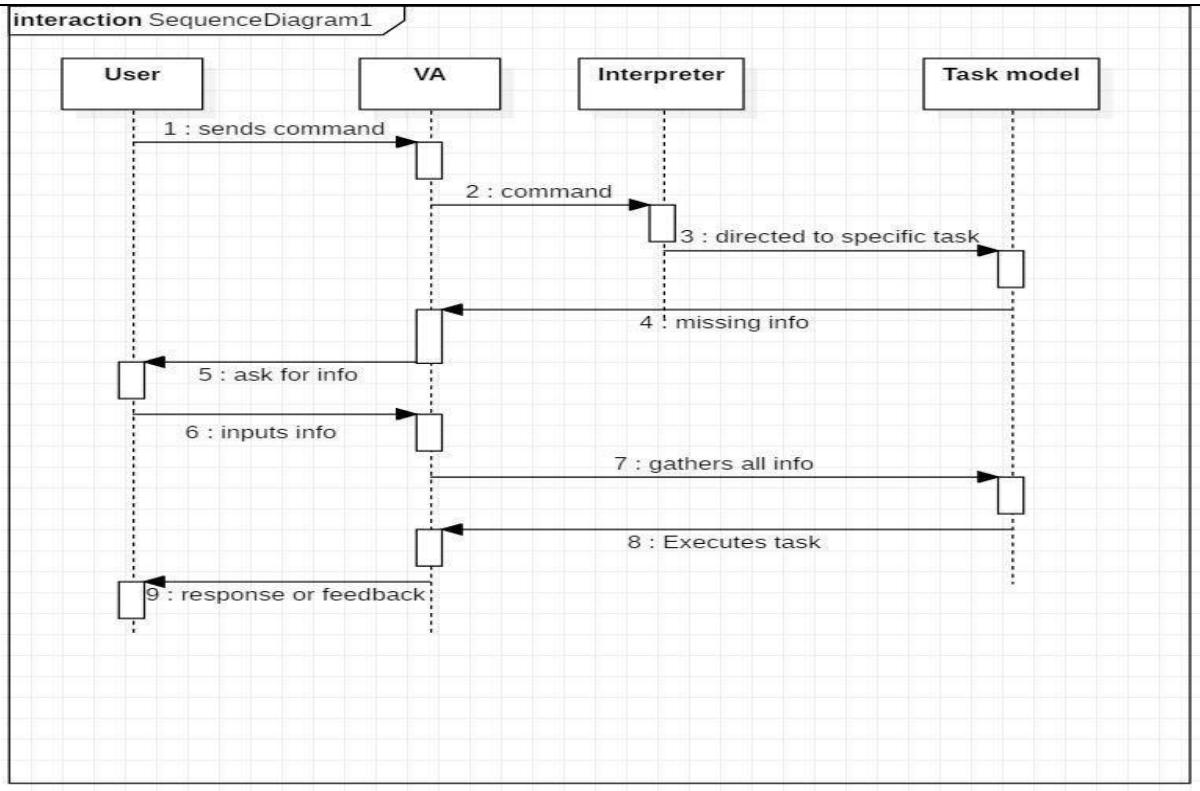

Figure 5. Sequence diagram for Task Execution

Source: Created by authors

The user sends command to virtual assistant in audio form. The command is passed to the interpreter. It identifies what the user has asked and directs it to task executer. If the task is missing some info, the virtual assistant asks user back about it. The received information is sent back to task and it is accomplished. After execution feedback is sent back to user.

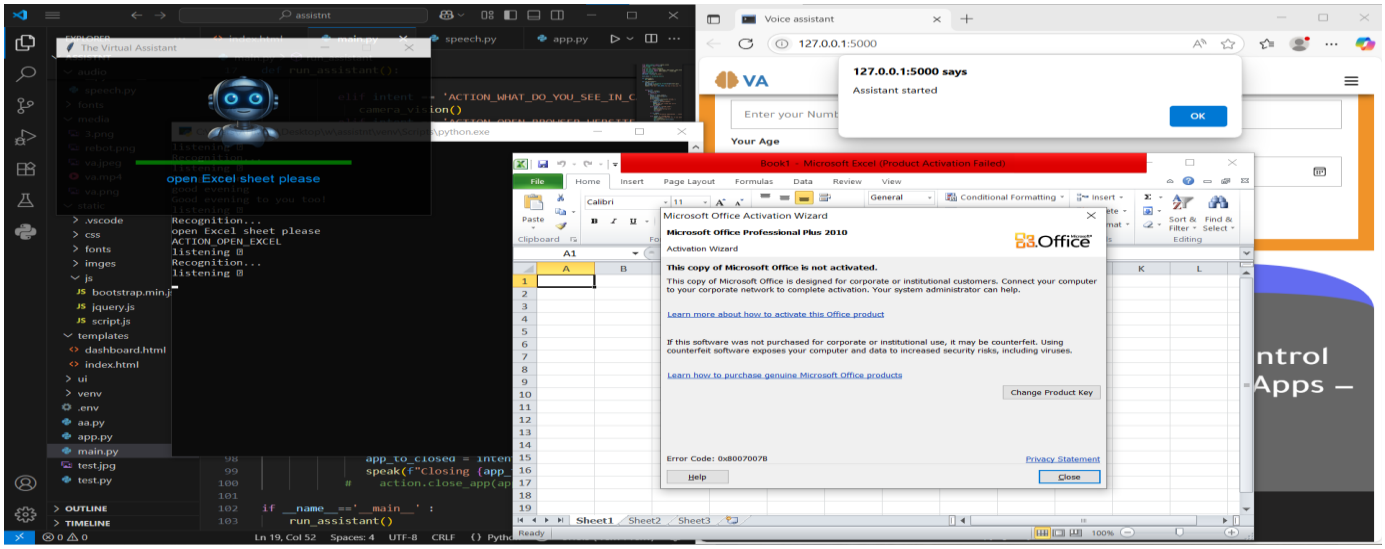

Figure **6**. test the system

The system's performance was evaluated under various acoustic conditions, as shown in Figure 6. The evaluation included measuring response time, speech recognition accuracy, and word error rate in both quiet environments (such as closed rooms) and noisy environments (such as streets or public spaces). The results showed that the system performed significantly better in low-noise environments, with higher accuracy and faster responses. In contrast, high noise led to increased speech recognition errors and increased response time. These results underscore the importance of incorporating advanced noise filtering techniques to ensure reliable performance in various real-world scenarios.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

User Study and Feedback:

To evaluate the ease of use and effectiveness of the voice assistant in real world scenarios, a mini-usage study was conducted with five participants from the target audience, including the blind and the elderly. Each participant was asked to interact with the assistant by performing a set of predefined tasks, such as opening apps, obtaining information, or giving general voice commands. After completing the experiment, participants were asked a series of questions to evaluate their experience, focusing on three main areas: (1) ease of use, (2) speech recognition accuracy, and (3) their preference for using voice commands over traditional typing. The results showed that most users found the system easy to use with four out of five participants indicating that the assistant understood their commands well. Blind users expressed high satisfaction with the hands free voice interaction feature while the elderly expressed a significant preference for using voice over typing. However, some observations were noted that required improvement, such as response speed and difficulty understanding unclear speech. This study provided important insights into the real world usability of voice assistants and emphasized the importance of inclusive design and testing with diverse audiences to ensure accessibility and effectiveness for all users.

IV. Evaluation and Results

The virtual assistant system was evaluated in a controlled desktop environment running a standard operating system configuration. The goal of the testing phase was to determine the system's responsiveness, reliability, and overall functionality in handling user commands and managing tasks. The assistant was able to successfully recognize and execute voice commands related to common operations such as launching desktop applications**,** retrieving stored user information**,** recording notes or activities**,** and managing scheduled tasks. Throughout the testing phase, the system maintained stable performance and exhibited accurate voice recognition, even in the presence of minor background noise. This was made possible by integrating a pre-trained speech-to-text engine, which effectively translated voice input into executable instructions. Additionally, the voice-based password system used for securing user-defined attributes functioned as intended, requiring a matching voice clip for access to sensitive data. This feature added a layer of personalized security without compromising usability. The assistant also demonstrated the ability to handle multiple concurrent tasks without freezing or crashing, indicating efficient resource management. For example, while a voice command was being processed, the GUI remained responsive, allowing users to interact with other elements such as task lists or query logs. The system's task management feature properly categorized tasks as parent or child, tracked their status changes, and prioritized execution based on user-defined importance levels. To further document these outcomes, screenshots were taken at various stages, capturing the assistant's interface while processing input and delivering feedback. These visuals confirm that the system adheres to expected design standards and maintains a user-friendly interface throughout usage.

In summary, the virtual assistant performed reliably under standard conditions, meeting its functional requirements and demonstrating a high degree of usability and efficiency. These initial results suggest that the system is well-suited for deployment in a personal or professional desktop setting, with potential for further expansion to other platforms.

A practical evaluation of the voice assistant's performance was conducted in various environments to measure its efficiency and responsiveness under various acoustic conditions. The system was tested in two main environments: the first in a closed quiet room, and the second in a noisy crowded environment such as a street or public space. During each experiment, several performance indicators were recorded, including response time, speech recognition accuracy and word error rate. In a quiet environment, the system demonstrated high performance, with a response time of approximately 2 seconds and a speech recognition accuracy rate exceeding 95%, while the word error rate was very low (approximately 1 to 2 words out of 100). In a noisy environment performance deteriorated somewhat due to acoustic interference. The response time was approximately 5 seconds the accuracy rate dropped to approximately 80% and the word error rate rose to approximately 10 words out of 100.These results demonstrate the importance of testing the voice assistant under various conditions to ensure its reliability in real-world use. There is also a need to improve noise filtering algorithms and develop models that are more capable of handling audio inputs in noisy environments.

V. Conclusion and Future Work

As part of our project's development, we aim to expand the capabilities of the virtual voice assistant in the future by incorporating advanced and intelligent features that enhance its practical value to users. Among these features, we are working on adding the ability to automatically generate code in various programming languages, summarize videos for quick understanding of their content, and improve the text-to-speech (TTS) technology to be more natural and interactive. We also plan to expand the system's capabilities by developing a large language model (LLM) specifically for the project, supporting multiple languages, including Arabic, French, and Kannada. This will enable broader interaction with users from diverse cultures and linguistic backgrounds, and improve the system's ability to understand and respond to commands in multiple contexts. One of the applications we are particularly interested in is customizing this assistant to serve students, specifically Andhra Pradesh University students, by designing an intelligent assistant system that guides them through the steps of preparing and writing academic papers in a professional academic manner. The assistant will focus on guiding students, especially those who have never published a research paper, step by step in writing a successful research paper.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue VI, June 2025

This will include:

- Collecting and analyzing similar research papers from Google Scholar based on the student's field.
- Deriving the optimal structure for the paper (introduction, methodology, results, discussion, etc.).
- Providing detailed instructions on academic writing for each part of the paper.
- Assisting in drafting titles and abstracts in appropriate and professional language.
- Academic proofreading and improving the quality of written language using specialized artificial intelligence models.
- Plagiarism detection and guidance on proper rewording to reduce similarity and achieve originality.
- Suggesting appropriate journals or conferences for publication based on the paper's field.

In this way, the assistant is not just a technical tool, but an academic partner that helps students enter the world of scientific publishing with confidence and mitigates the challenges they face at the beginning of their research career.

References

1. Valera Román A, Pato Martínez D, Lozano Murciego Á, Jiménez-Bravo DM, de Paz JF. Voice assistant application for avoiding sedentarism in elderly people based on IoT technologies. Electronics. 2021 Apr 20;10(8):980.
2. Patil J, Shewale A, Bhushan E, Fernandes A, Khartadkar R. A voice-based assistant using Google dialogflow and machine learning. International Journal of Scientific Research in Science and Technology. 2021 May;8(3):6-17.
3. Abougarair AJ, Aburakhis MK, Zaroug M. Design and implementation of smart voice assistant and recognizing academic words. International Robotics & Automation Journal. 2022;8(1):27-32.
4. Mei Y. AI & Entertainment: The Revolution of Customer Experience. Lecture Notes in Education Psychology and Public Media. 2023 Dec 7;30:274-9.
5. Shashikala KS, Vadlamudi S, Gurupriya M, Teja KD, Reddy MP, Reddy JK. Smart helper: A voice guided assistance for visually impaired. InChallenges in Information, Communication and Computing Technology 2025 (pp. 597-600). CRC Press.
6. Masina F, Orso V, Pluchino P, Dainese G, Volpato S, Nelini C, Mapelli D, Spagnolli A, Gamberini L. Investigating the accessibility of voice assistants with impaired users: mixed methods study. Journal of medical Internet research. 2020 Sep 25;22(9):e18431.
7. Hoy MB. Alexa, Siri, Cortana, and more: an introduction to voice assistants. Medical reference services quarterly. 2018 Jan 2;37(1):81-8.
8. Wired. (2023, October 5). Why voice assistants still can't hold a conversation. Wired. https://www.wired.com/story/voice-assistants-ambient-computing/
9. Song IY, Evans M, Park EK. A comparative analysis of entity-relationship diagrams. Journal of Computer and Software Engineering. 1995;3(4):427-59.
10. Mohammed MA, Muhammed DA, Abdullah JM. Practical Approaches of Transforming ER Diagram into Tables. International Journal of Multidisciplinary and Scientific Emerging Research. 2015 Oct;4(22):2349-6037.
11. Al-Fedaghi S. UML sequence diagram: an alternative model. arXiv preprint arXiv:2105.15152. 2021 May 31.
12. Swain SK, Mohapatra DP, Mall R. Test case generation based on use case and sequence diagram. International Journal of Software Engineering. 2010 Jul;3(2):21-52.