IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 13, Issue 01, January 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1301.05

MAY: Multifunctional Assistance for you

Prof.Suma S
Assistant Professor, Department of CSE,
Vemana Institute of Technology, Bengaluru, India
suma.s@vemanait.edu.in
https://orcid.org/0000-0001-6970-9905
Ali Haider,Deepak B,Mohammed Rayyan,Mohit Pandit
Students, Department of CSE,
Vemana Institute of Technology, Bengaluru, India
ah795594@gmail.com, Zoexcosmo@gmail.com,
sheikrayyan01@gmail.com, monteemohit2004@gmail.com

Publication History
Manuscript Reference No: IJIRAE/RS/Vol.13/Issue01/AEJA26.JAAE10084
Research Article | Open Access | Double-Blind Peer-Reviewed | Article ID: IJIRAE/RS/Vol.13/Issue01/AEJA26.JAAE10084
Received:12, December 2025, Revised: 24, December2025, Accepted: 02, January2026,PublishedOnline:20,January 2026.
https://www.ijirae.com/volumes/Vol13/iss-01/05.AEJA26.JAAE10084.pdf
Article Citation:Suma,Ali,Deepak,Mohammed,Mohit(2026),MAY: Multifunctional Assistance for You, IJIRAE: International Journal of Innovative Research in Advanced Engineering, Volume 13, Issue 01 of 2026 pages 22-27
Doi:> https://doi.org/10.26562/ijirae.2026.v1301.05
BibTeX Key: Suma@2026MAY
IJIRAE papers should be cited as IJIRAE (International Journal of Innovative Research in Advanced Engineering, AM Publications, India 2025, ISSN 2349-2163, https://doi.org/10.26562/ijirae.2026.v1301.05 The journal's official abbreviation is IJIRAE. Orcid: https://orcid.org/0009-0004-9398-7488

Abstract: The growing dependence on digital systems has highlighted the need for more natural, hands-free interaction methods, as traditional input devices often limit accessibility and efficiency. In today's world, voice-driven interfaces have significantly transformed human-computer interaction by enabling intuitive, conversational control across various applications. To address this demand, this paper offers an AI- powered voice assistant that allows seamless communication with a computer through spoken language. The system captures speech, converts it into text, interprets user intent, executes the corresponding command, and delivers responses through synthesized speech, removing the need for a graphical interface. Its architecture integrates speech recognition, natural language processing, command execution modules, and text-to-speech synthesis into a unified conversational loop. The implementation uses a microphone, speaker, and an ESP32/Arduino microcontroller for signal handling, while embedded software manages analysis and communication. Experimental results show effective performance in tasks such as reminders, time and date queries, application control, and information retrieval, achieving low latency, high responsiveness, and strong usability. The system meets its design objectives, and future improvements such as multilingual support, offline learning- based models, and IoT integration can enhance adaptability and overall efficiency.
Keywords: Voice Assistant, Speech Recognition, Natural Language Processing, Intent Detection, Text-to-Speech, ESP32, Arduino, Embedded Systems, Human–Computer Interaction, Hands-Free Control, Command Execution, Real-Time Processing, Conversational Interface, Automation, Accessibility.

I. INTRODUCTION

Human–computer interaction has seen a major shift from traditional peripherals toward intuitive voice-based interfaces. Voice assistants simplify digital communication by enabling users to operate systems through natural speech rather than physical input devices. The present project focuses on designing and implementing an AI-based voice assistant that interprets spoken queries, processes them intelligently, and generates verbal responses. By combining speech recognition, natural language understanding, and speech synthesis, the system enables an intuitive and hands-free communication environment. The goal is to improve convenience, accessibility, and productivity by allowing users to perform routine tasks using voice commands alone. [1],[2]

II. RELATED WORK

Research in intelligent personal assistants has shown diverse design and application approaches. Doshi etal.(2017) developed *Donna*, a web-based assistant that focused on automating information access but relied extensively on internet connectivity. Kumaran et al. (2022) introduced an AI assistant in corporating emotion-based response mechanisms, though functionality was limited to basic operations. T.R.(2023) presented a personal desktop AI assistant emphasizing task automation, yet it lacked advanced natural language processing capabilities. Sajja et al. (2023) built an educational assistant tailored for personalization in learning environments but was restricted to academic usage. These studies contributed significantly to the field but revealed limitations such as cloud dependency, narrow command adaptability, and limited contextual interpretation. The proposed system addresses these shortcomings through reliability, local execution, and extensibility. [5][8]

III. SYSTEM ANALYSIS

To ensure a well-structured and efficient development process, the system requirements were divided into four major categories: functional, non-functional, hardware, and software, following the systematic design approaches adopted in several modern AI-based assistant architectures [1],[2]. The functional requirements outline the essential operations that allow the voice assistant to behave like an interactive system. These include capturing spoken input through a microphone, converting the audio signal into text using STT algorithms, interpreting the meaning and intent behind the transcribed text, performing the relevant task such as answering queries, setting reminders, or controlling applications and finally generating a spoken response through TTS. This multi-stage process is fundamental to most successful speech driven assistants developed in recent research [3],[4].

The non-functional requirements go beyond core functionality and focus on ensuring the system performs effectively in real-world environments. These factors include real-time responsiveness to provide smooth and natural conversation flow, high accuracy in speech recognition to avoid misinterpretation, reliability to ensure stable operation over long periods, security to safe guard user interactions, and accessibility so that users of all skill levels can operate the system comfortably. These quality attributes have also been emphasized as critical components in AI and ML-driven assistant models discussed in existing studies [5],[6].The hardware requirements describe the physical components needed for the system to function. A microphone serves as the primary input device for capturing user speech, while an ESP32/Arduino microcontroller manages signal processing, connectivity, and communication between software modules. A speaker enables clear verbal output, and a stable power supply ensures uninterrupted operation. Such hardware layouts are commonly adopted in embedded and IoT- integrated voice assistants found in the literature [7],[9].The software requirements form the intelligence layer of the system, consisting of the Speech-to-Text (STT) engine for converting audio into text, Natural Language Processing (NLP) for understanding commands and extracting intent, a command execution module to determine the appropriate actions, and a Text-to-Speech (TTS) engine to convert the system's response back into speech. These components mirror the architectural patterns used in advanced AI-powered and emotion-aware personal assistants proposed in recent research works [6],[8],[10].

Together, these functional, non- functional, hardware, and software elements create comprehensive, scalable, and predictable system architecture capable of delivering smooth, real-time, and user-friendly voice interactions. keywords, context, intent, and emotional state determining what action the user wants an essential feature highlighted in recent NLP-driven assistant systems [5],[6]. After intent and emotion are identified, the Response Generation unit decides appropriate system action through casual phrase handling, emotion-aware templates, local time/date processing, and AI generation receiving contextually appropriate responses that are sent to ESP32 triggering display updates with auto-scrolling for longer messages when needed, following similar logic used in AI personal assistant implementations [8],[10]. This response text is passed to the Text-to-Speech module that updates ESP32 display status, calls Eleven Labs API with configured voice and audio generation model producing natural-sounding audio saved to temporary file, loaded for playback through computer speakers with synchronous blocking until completion, and cleaned up afterward. The ESP32 provides multimodal output through the OLED display showing system states and scrolled responses while computer speakers deliver synthesized speech, completing the conversational loop with visual and audio feedback.

IV.SYSTEM DESIGN

The system design of the MAY Voice Assistant follows a distributed client-server architecture with modular hardware and software components ensuring smooth, reliable, and emotion-aware real-time interaction, similar to approaches adopted in modern AI-based assistant models [1], [2]. The process begins when the user speaks near the computer's microphone or the INMP441 I2S digital microphone connected to the ESP32, capturing audio signals through Python's audio processing library for backend operations or through I2S interface at 16 kHz sample rate for hardware demonstration. The ESP32 microcontroller featuring dual- core processor at 240 MHz with built-in WiFi acts as the hardware interface between physical components and the intelligent software layer, running firmware that handlesI2S audio capture with DMA buffering, manages OLED display via I2C protocol providing real-time visual feedback with auto-scrolling capability for long messages, and maintains TCP server on port 8888 for bidirectional communication with Python backend through command parsing, an approach widely used in embedded and smart assistant designs [3],[7],[9].

Once audio is captured by the computer microphone, it flows to the Speech Recognition module where Google Speech Recognition API performs Speech-to-Text conversion with ambient noise adjustment and exception handling to convert voice input into textual data that forms the foundation of intelligent voice assistant architectures and enables further computational analysis [4], [6]. The generated text is simultaneously sent toESP32viaTCPcommandfor display with word-wrapped rendering, and analyzed by multiple Natural Language Processing modules including Hume AI's batch jobs API extracting top three emotions with confidence scores from 48+ categories with sentiment analysis as fallback, followed by Anthropic's Claude language model that constructs comprehensive prompts combining user query, emotional context, conversation history from five message buffer, and empathy instructions to interpret To visually represent and simplify understanding of this end-to-end flow, diagrams such as the overall system architecture showing distributed client-server model with ESP32 hardware layer and Python software layer, sequence diagram illustrating data movement from user speech through wake word detection, speech recognition, emotion detection, AI processing, text-to-speech synthesis, and multimodal output, and data flow diagram depicting execution phases are used, similar to modeling techniques adopted in several published smart assistant frameworks [1],[3],[4].

IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 13, Issue 01, January 2026 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2026.v1301.05

V. SYSTEM ARCHITECTURE

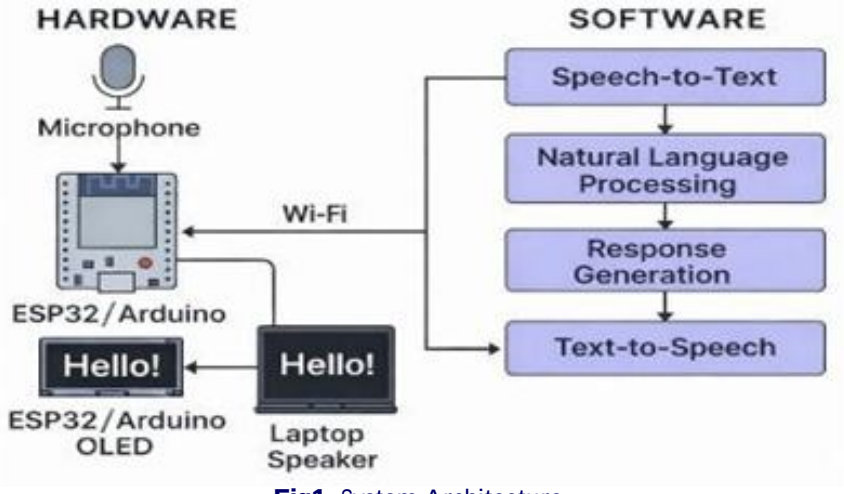

Fig1: System Architecture

The proposed MAY Voice Assistant system adopts a distributed client-server architecture integrating ESP32 microcontroller-based hardware interface with a Python- based AI processing pipeline orchestrating multiple cloud services for speech recognition, emotion detection, natural language understanding, and text-to-speech synthesis. This design approach aligns with established architectures used in contemporary embedded and AI-based voice assistant implementations while introducing emotion-aware response generation capabilities [1],[2],[6]. The system consists of two major subsystems: the ESP32 hardware layer providing audio capture, visual feedback, and network communication, and the Python software layer performing computationally intensive AI operations including Google Speech Recognition, Hume AI emotion detection, Claude conversational AI, and Eleven Labs text-to-speech synthesis. This distributed model enhances maintain ability through modular component design with clearly defined interfaces, supports scalable expansion by enabling AI service replacement without firmware modifications, leverages embedded hardware efficiency for real-time interface operations while utilizing computer processing power for complex AI tasks, and mirrors widely adopted frame works in similar assistant systems with improvements in emotional intelligence and multimodal feedback [3],[4],[5].

The hardware sub system comprises the INMP441I2S digital microphone connected to ESP32 via designated GPIO pins for high-quality audio capture at 16 kHz sample rate, the ESP32 microcontroller featuring dual-core processor at 240 MHz with built-in WiFi programmed using Arduino frameworkimplementingfirmwarethatconfiguresI2Sdriver with DMA buffers for continuous audio capture, maintains TCP server listening on port 8888 accepting client connections and parsing commands, and manages peripheral communication protocols, the OLED display communicating via I2C at 100 kHz bus speed providing real-time visual feedback through multiple display states showing ready status with network information, listening indicators, transcribed user speech with word-wrapping, assistant replies with intelligent auto-scrolling when message length exceeds screen capacity advancing content periodically with scroll indicators showing additional text availability, and speaking status during audio synthesis, and WiFi connectivity enabling TCP/IP communication with Python backend on local network with automatic reconnection logic. Such microcontroller-based architectures are commonly employed in lightweight virtual assistant prototypes due to their low power consumption, built-in wireless interfaces supporting reliable network communication, GPIO flexibility for peripheral control, and suitability for real-time signal handling with DMA-based audio streaming achieving minimal latency [4], [6], [9]. The ESP32 handles bidirectional communication with Python software modules through transmission of status updates and reception of display commands, manages I2C and I2S peripheral protocols for display and microphone interfacing, orchestrates visual feedback through OLED state-based rendering and auto-scroll algorithms, and provides real-time system state indication similar to user interaction methods found in prior embedded assistant designs [1], [7].

The software subsystem implemented in Python follows a multi-stage AI-powered voice processing pipeline with emotion awareness and multimodal output capabilities. First, the Speech-to-Text module utilizing Google Speech Recognition API captures audio with configurable parameters, performs automatic ambient noise adjustment, executes transcription handling various exceptions, and returns text strings that form the input layer for language interpretation and emotion analysis, with transcribed text sent to ESP32 via TCP command for visual confirmation. STT pipelines are standard in AI-driven personal assistant systems because they offer consistent and structured textual representation of user queries enabling subsequent semantic processing [2], [4], [10]. The converted text is processed by dual analysis modules starting with HumeAI integration that submits text to emotion analysis API creating batch analysis jobs, polls for results, parses responses extracting top three emotions with confidence scores from 48+ categories, and formats results showing primary emotion prominently with secondary and tertiary emotions, falling back to sentiment analysis when Hume API is unavailable ensuring continued empathetic capability, followed by Natural Language Processing through Anthropic's Claude language model that constructs comprehensive prompts combining user query text, emotional context with instructions for empathetic acknowledgment, conversation history from five-message buffer enabling coherent multi-turn dialogues, and constraints for concise voice-optimized output, submitting prompts and receiving responses.

NLP-based intent modeling combined with emotion detection has been adopted due to reliability in understanding conversational input while providing empathetic awareness beyond traditional command-response patterns [6], [7], [8]. The interpreted intent with emotional context is forwarded to the Response Generation module that generates contextually relevant textual responses through multiple mechanisms including quick template-based replies for common phrases, emotion-aware templates mapping detected emotions to empathetic responses with supportive messages, local handling for time/date queries, and AI generation producing contextually appropriate empathetic responses influenced by emotional state, with generated responses sent to ESP32 via TCP commands triggering display updates. Finally, the Text-to-Speech module synthesizes generated text into audio signals by sending status command to ESP32 updating display, calling Eleven Labs API with configured parameters and audio generation model producing natural-sounding audio, streaming chunks to temporary file, loading for playback with synchronous blocking ensuring audio completes before continuation, and cleaning up temporary file, outputting synthesized speech through computer speakers while ESP32 maintains speaking display status until completion. The Speech Recognition → Emotion Detection → NLP → Response Generation → TTS pipeline is consistent with prior embedded voice assistant designs emphasizing modularity and efficient execution while adding emotional intelligence capabilities [1], [5], [9].

By combining an ESP32 embedded hardware interface providing real-time audio capture capabilities, visual feedback through auto-scrolling OLED display, and network communication via TCP/IP, with a Python-based software pipeline orchestrating multiple cloud AI services including Google Speech Recognition for transcription, Hume AI for emotion detection across 48+ categories, Claude for contextual natural language understanding with empathetic response generation, and ElevenLabs for natural text-to- speech synthesis, the system achieves emotion-aware, low- latency conversational interaction with multimodal feedback through synthesized speech and visual display updates. This distributed architecture ensures efficient resource utilization by leveraging ESP32 for lightweight real-time interface operations while utilizing computer processing power for computationally intensive AI operations, reduces external cloud dependencies through comprehensive fallback mechanisms, provides operational independence through modular component design enabling AI service replacement, and aligns with modern trends in edge-based intelligent assistants focused on privacy through local network communication, responsiveness through minimal audio streaming latency and reasonable total response times, emotional intelligence through AI-powered detection and empathetic prompting, and autonomous execution through distributed processing with graceful degradation [3],[5],[8].

VI. DATAFLOW DESCRIPTION

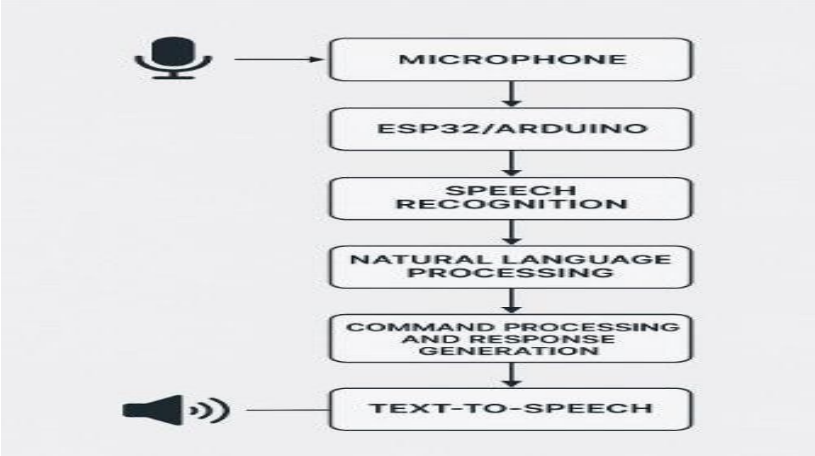

Fig 2: Dataflow diagram

The data flow of the proposed MAY Voice Assistant system follows a sequential and modular processing pipeline with emotion awareness and multimodal feedback. The process begins when the computer microphone captures the user's speech as audio signal through Python audio library with configurable parameters, or alternatively the I2S digital microphone connected to ESP32 captures audio at configured sample rate demonstrating hardware acquisition capability. High-fidelity audio acquisition with automatic ambient noise adjustment is essential for accurate recognition and is consistent with sensing approaches described in prior embedded voice-assistant research [1],[3]. The captured signal is processed by the passive listening loop in Python backend that continuously monitors audio input for wake word detection, while ESP32-captured audio through configured I2S driver with DMA buffers undergoes bit conversion and can be transmitted over TCP demonstrating distributed processing capability. Microcontroller-based preprocessing and network transmission has been widely used in low-power intelligent assistant systems due to efficiency and suitability for real-time processing [4], [6], [9]. The audio signal is passed to the Speech Recognition module where speech-to-text conversion occurs by streaming to Google Speech Recognition API executing transcription that extracts acoustic features and maps them to linguistic units, handling various exceptions, returning transcribed text strings. This STT-based conversion pipeline aligns with methodologies implemented in modern AI-driven voice assistants[1],[2],[10].The transcribed text is simultaneously sent to ESP32 via TCP socket transmitting command that the ESP32 firmware parses and dispatches to display rendering function showing transcribed speech on OLED with word- wrapping providing visual confirmation to user.

The textual output is then processed by dual analysis stages starting with Hume AI integration that submits text via API creating emotion analysis batch job, polls for results, parses responses extracting top three emotions from extensive categories with confidence scores, formats results showing primary emotion prominently with secondary / tertiary emotions, and returns emotion data, falling back to sentiment analysis performing polarity scoring when Hume API fails, followed by Natural Language Processing through Claude language model that constructs comprehensive prompts combining user query text, emotional context with instructions for empathetic response, conversation history from message buffer enabling coherent multi-turn dialogues, and constraints for concise output, submitting via API client where intent recognition, context interpretation, and semantic understanding occur enhanced by emotional awareness. NLP-driven analysis combined with emotion detection significantly enhances command comprehension, empathetic interaction quality, and contextually appropriate response generation as demonstrated in recent assistant architectures [6],[7],[8]. Once the command intent is extracted and emotional state is determined, the data moves to the Response Generation module which determines appropriate system action through multiple mechanisms including checking query against common phrase patterns and returning quick template replies, mapping detected. Emotions to empathetic template responses with supportive communication arrays, local processing for time/date queries returning formatted system clock information, and AI API response generation receiving contextually appropriate empathetic text that formulates coherent textual response influenced by emotional context and conversation history. Such structured response generation frameworks with emotional awareness are central to intelligent conversational agents [5], [8]. The resulting response text is forwarded to dual output channels starting with TCP transmission to ESP32 that parses command calling display update when text length exceeds threshold to activate auto-scroll through state management and rendering logic, or standard display for shorter messages. The response text is simultaneously passed to Text-to-Speech synthesis that sends status command updating ESP32 display, calls Eleven Labs API receiving audio generator yielding chunks, creates temporary file, writes audio chunks, loads into audio play back system, plays with blocking execution waiting until completion, unloads audio, attempts clean up, and sends ready command returning ESP32 display to idle state. The synthesized speech is delivered through computer speakers whileESP32 OLED provides synchronized visual feedback, completing the human–machine conversational loop with multimodal output combining audio synthesis and visual display. This end-to-end flow ensures real-time emotion- aware interaction with reasonable total response time perceived as natural conversational pacing, reduced audio streaming latency through DMA buffering, efficient distributed processing leveraging ESP32 for real-time interface and Python for AI operations, comprehensive fallback mechanisms ensuring continued operation when services fail, and multimodal feedback through synthesized speech and auto-scrolling OLED display, consistent with embedded assistant systems documented in literature while adding emotional intelligence capabilities [3],[5],[6].

VII. IMPLEMENTATION

The proposed voice assistant, named "May," is a desktop- based personal AI voice assistant implemented in Python with hardware integration via an ESP32 microcontroller equipped with an SH1106 OLED display and an INMP441 digital microphone [3],[9]. The system operates in a client- server architecture where the Python application running on a PC acts as the main intelligence core, while the ESP32 serves as a dedicated peripheral for real-time visual feedback and audio capture [4],[7]. The core functionality includes wake-word detection ("May"), continuous passive listening, speech-to-text using Google's free web API via the Speech Recognition library, natural language understanding and response generation using Anthropic's Claude models (via asynchronous API calls), and high-quality text-to-speech synthesis using Eleven Labs' neural voices [1],[2],[6].The system supports conversational context maintenance (short-term memory of recent exchanges), reminder management with persistent storage in a JSON configuration file, battery level monitoring with threshold-based alerts, and time zone-aware time/date queries [5],[9]. Communication between the PC and ESP32 is achieved through a TCP socket connection over WiFi on port 8888, where the Python application sends structured commands (e.g., LISTENING, INPUT, RESPONSE, SPEAKING, READY, EMOTION) along with associated text pay loads to the ESP32, which updates the OLED display with intelligent text wrapping and automatic scrolling for long messages [3],[4]. Additionally, the ESP32 streams raw audio samples from the I2S-configured INMP441 microphone back to the PC when instructed, although primary speech recognition is performed locally on the host machine [1],[6]. The system emphasizes emotional intelligence by analyzing user affect from transcribed text using Hume.ai's multimodal emotion recognition API with fall back to Text Blob sentiment analysis, and adapting responses accordingly to provide empathetic replies for detected emotions such as joy, sadness, anger, or anxiety [2],[8]. Robust error handling, logging, and fallback mechanisms ensure graceful degradation when cloud services are unavailable defaulting to local sentiment analysis and predefined empathetic responses [6],[10]. This implementation advances prior voice assistant designs by integrating state-of-the-art cloud-based language models for response generation, real-time emotion detection, premium neural TTS, and a custom hardware display interface for enhanced user interaction and visual feedback [5],[7],[10].

VIII. TESTING AND RESULTS

System verification was conducted through a series of structured and iterative test cycles designed to evaluate both functional and non-functional behavior of the voice assistant. Each module of the system including speech recognition, NLP-based intent analysis, command execution engine, and text-to-speech output wastested independently as well as in full integration mode. The testing process covered multiple dimensions such as recognition accuracy under varying speech speeds and accents, precision of command execution for different task types, clarity and naturalness of the generated voice output, system responsiveness under continuous usage, and the ability to handle incorrect, in complete, or ambiguous inputs.

The assistant was evaluated through real-time user interactions and repeated stress testing to assess long-duration stability. Across all test cases, the system consistently produced correct results, maintaining high accuracy in interpreting commands and delivering intended responses. Even in non-ideal test conditions such as moderate background noise, unclear pronunciation, and fast speech, the assistant was able to successfully identify and execute most requests with minimal delay. Performance measurements confirmed that the system maintained low latency between user input and system response, ensuring natural and uninterrupted interaction. Furthermore, the system demonstrated robustness against errors by providing meaningful feedback instead of failing or crashing when invalid commands were issued. The overall testing outcomes validate that the developed voice assistant meets all functional and non-functional requirements, proving reliable, efficient, and user-friendly for real-time hands-free operation. [4],[8]

VII. CONCLUSION AND FUTURE ENHANCEMENTS

This project successfully implements a fully operational AI- driven voice assistant capable of real-time verbal interaction. With the ability to interpret spoken commands and generate synthesized speech, the system makes human–computer communication more natural and efficient. Its modular structure, performance consistency, and user accessibility validate its practical usefulness. Future developments could focus on:

- More advanced offline STT and TTS models to reduce network dependency
- Multi lingual speech recognition for regional inclusivity
- Mobile app integration to enable remote control and customization
- IoT connectivity for smart-home automation
- Personalized response learning based on user preferences These enhancements can transform the assistant into a highly adaptive, autonomous, and intelligent digital companion.[5],[9]

ACKNOWLEDGMENT

I express my sincere gratitude to my guide **Prof.Suma S,** for their continuous support, valuable guidance, and constructive feedback throughout the development of this project. Their expertise and encouragement played an instrumental role in shaping the direction and successful completion of this work. I would also like to extend my heartfelt thanks to the Department of CSE, Vemana Institute of Technology (VTU affiliated), for providing the necessary resources, laboratory facilities, and a favorable research environment to carry out this project effectively. Finally, I am deeply grateful to my family and friends for their constant motivation, understanding, and moral support during the entire span of this work. Their encouragement has been a driving force in overcoming challenges and achieving this mile stone.

REFERENCES

The following references have been consulted to support the research, design, and development of the proposed AI-based voice assistant system. These sources include studies on speech recognition, natural language processing, embedded system integration, and intelligent personal assistants. They provide valuable insights into existing methodologies, system architectures, and implementation strategies, enabling comparative analysis and strengthening the technical foundation of this work.

1. D.Pandey,A.Ali,S.Dubey,M.Srivastava,S.Dwivedi,and M. S. Raza, "Voice Assistant Using Python and AI," International Research Journal of Engineering and Technology(IRJET), vol. 9, no. 5, pp. 832–838, May 2022.
2. S.Devi, Z. A. Merchant, M. S. Siddiqui, and M. Lobo," Artificial Intelligence based Personal Assistant," Asian Journal for Convergence in Technology (AJCT),vol.5,no.1, pp.1–4, 2019. https://doi.org/10.33130/AJCT.2019v05i01.003
3. A. J. Abougarair, M. K. I. Aburakhis, and M. O. Zaroug," Design and implementation of smart voice assistant and recognizing academic words," International Robotics &Automation Journal, vol. 8, no. 1, pp. 27–32, 2022. https://doi.org/10.15406/iratj.2022.08.00240
4. A.Agarwal,R.Rai, and P. K. Chaubey, "BRAIN – THE A.I.(Personal Voice Assistant)," Int. Res. J. Eng. Technol., vol.7, no. 4, pp. 1234–1238, 2020.
5. M. Anusha, D. K. Harshitha, M. Kavya, K. Neha, T. T.Neelakantappa, B. E. Ramesh, and T. V. Aravinda, "Voice-based virtual assistant using AI and ML," Int. J. Innov. Sci.Res.Technol., vol. 1, 2025.
6. S.Pakhmode,V. Poojary, P.Bhore, K.Thakur, andV. Dethe," NLP based AI Voice Assistant, "Int. J. Sci. Res. Eng. Manag., vol. 7, no. 3, pp. 1–6, Mar. 2023
7. A.Doshi, R. Shah,D. Bhimani, B. Patel,and S. Mali, "Donna A web based AI Personal Assistant," International Journal of Computer Applications, vol. 175, no. 8, pp. 7–12, Oct. 2017,doi:https://doi.org/10.5120/ijca2017915610.
8. N. Kumaran, K. E V, and M. S, "Personal Assistant with Emotion Recognition Based on Artificial Intelligence," International Journal of Innovative Technology and Exploring Engineering, vol. 11, no. 4, pp. 67–70, Mar. 2022, doi:https://doi.org/10.35940/ijitee.c9812.0311422.
9. N. M. T. R, "Personal A.I. Desktop Assistant," International Journal of Information Technology Research and Applications, vol. 2, no. 2, pp. 54–60, Jun. 2023, https://doi.org/10.59461/ijitra.v2i2.58
10. R.Sajja,Y.Sermet,M.Cikmaz,D.M.Cwiertny,andİ.Demir," Artificial Intelligence-Enabled Intelligent Assistant for Personalized and Adaptive Learning in Higher Education, "arXiv (Cornell University), Sep.2023, doi: https://doi.org/10.48550/arxiv.2309.10892.