IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁙ Peer-reviewed & Refereed journal ⁙ Vol. 14, Issue 7, July 2025

DOI: 10.17148/IJARCCE.2025.14738

AI Voice Assistant with Task Automation

Prof. Anila Nair[1], Prof. Varalakshmi V J[2]

Department of BCA, RR Institute of Management Studies Bengaluru, India[1]

Department of BCA, RR Institute of Management Studies Bengaluru, India[2]

Abstract: In the era of smart automation, AI-powered voice assistants have evolved beyond basic task execution to provide intelligent decision-making capabilities that significantly enhance human-computer interactions. With the integration of advanced machine learning algorithms and real-time data processing, modern voice assistants are not just reactive tools but proactive systems capable of learning and adapting to user behavior over time. This paper presents "Neon AI," a next-generation, customizable AI voice assistant developed using Python, designed to bridge the gap between standard virtual assistants and the growing need for personalized, context-aware automation. Neon AI incorporates robust voice recognition techniques, state-of-the-art natural language processing (NLP), and intelligent task automation capabilities to streamline user interactions across various platforms. Leveraging powerful AI models like GROQ and Cohere, Neon AI offers dynamic and adaptive responses tailored to individual preferences, making it versatile for both personal and professional applications. Additionally, the assistant features a user-friendly graphical interface developed using PyQt5, enhancing accessibility for users with varying technical backgrounds. The paper highlights the comprehensive methodology adopted in designing the modular architecture of Neon AI, details the implementation processes, and provides an in-depth analysis of the system's performance through empirical metrics and graphical evaluations. Experimental results demonstrate Neon AI's proficiency in handling multi-faceted tasks, delivering high accuracy in voice recognition, low response latency, and an engaging user experience, paving the way for future enhancements in AI-driven personal assistant technologies.

Keywords: AI Voice Assistant, Task Automation, Natural Language Processing, Neon AI, Voice Recognition, GROQ API, Speech Recognition, PyQt5, Automation

I. INTRODUCTION

The advancement of Artificial Intelligence (AI) has revolutionized human-computer interaction, especially through the emergence of voice-enabled virtual assistants. With the rapid progression of machine learning algorithms, natural language processing (NLP), and speech recognition technologies, AI-powered systems are now capable of interpreting, processing, and responding to human inputs with unprecedented accuracy. Existing systems like Siri, Alexa, and Cortana have brought AI assistants into mainstream usage by offering standardized services, such as answering queries, setting reminders, and controlling smart home devices. However, these systems tend to follow predefined scripts and lack deep personalization, often failing to adapt dynamically to individual user preferences, changing environments, or complex task demands.

To overcome these shortcomings, Neon AI was conceptualized and developed as an advanced AI-driven voice assistant, prioritizing flexibility, adaptability, and user-centric customization. Neon AI introduces the capability to switch between multiple AI models, such as GROQ, LLAMA, and GPT, allowing users to tailor the assistant's behavior and response style to their specific needs. Furthermore, it integrates a dynamic task automation engine capable of executing a wide range of system-level and web-based commands through simple voice prompts, reducing reliance on manual input.

An additional feature of Neon AI is its intuitive graphical user interface (GUI) developed using PyQt5, which enhances the usability of the assistant by offering visual feedback, task monitoring, and seamless interaction even in scenarios where voice interaction might be inconvenient or impractical. This paper presents the end-to-end design, modular system architecture, implementation methodology, and empirical performance evaluation of Neon AI. It highlights the system's scalability, real-time performance, and multi-functional capabilities that collectively position Neon AI as a next-generation intelligent assistant suitable for both personal and professional use cases.

II. LITERATURE REVIEW

Voice assistants traditionally rely on static conditional statements and pre-programmed flows, limiting their ability to handle complex, multi-turn conversations and dynamically changing contexts. Such systems, while effective for executing simple queries and tasks, often fall short when required to understand nuanced commands, infer user intent, or adapt to personalized usage patterns. Early implementations were based primarily on keyword detection and rule-based logic, which made them rigid and prone to failure in non- standardized user interactions.

With the advent of deep learning and neural network architectures, particularly large language models (LLMs), the landscape of voice assistants has undergone a transformative shift. LLMs such as OpenAI's GPT series, Meta's LLAMA, and specialized models like GROQ and Cohere have introduced capabilities for contextual understanding, semantic parsing, and natural language generation at scale. These models enable voice assistants to move beyond static flows, allowing for more fluid, human-like interactions.

Foundational works like Rabiner & Juang (1993) on statistical models of speech recognition laid the groundwork for acoustic modeling and hidden Markov models (HMM), while Jurafsky & Martin (2023) expanded these concepts with deep learning approaches to NLP, emphasizing context retention and conversational AI. Literature indicates that while LLM integration improves response quality, key challenges remain concerning real-time performance, user adaptability, and multi-modal interaction support.

Furthermore, recent academic investigations have explored hybrid frameworks combining voice recognition with deep NLP models, demonstrating improvements in response accuracy and task diversity. However, scalability concerns arise due to high computational demands, and personalization remains limited in most commercial implementations. Neon AI specifically addresses these challenges by incorporating a modular architecture, dynamic AI model switching, and a GUI-based interaction layer, aiming to create a more scalable, adaptable, and user-friendly voice assistant framework.

III. SYSTEM DESIGN

A. **Architectural Overview** Neon AI employs a comprehensive modular architecture that facilitates seamless integration of multiple AI-driven functionalities while maintaining high system efficiency and responsiveness. The architecture is strategically divided into five primary interconnected modules: Speech Recognition, Natural Language Processing (NLP) Processing, Task Automation, Graphical User Interface (GUI) Interaction, and Response Generation. Each module operates independently yet cohesively within the system, promoting scalability and ease of maintenance.

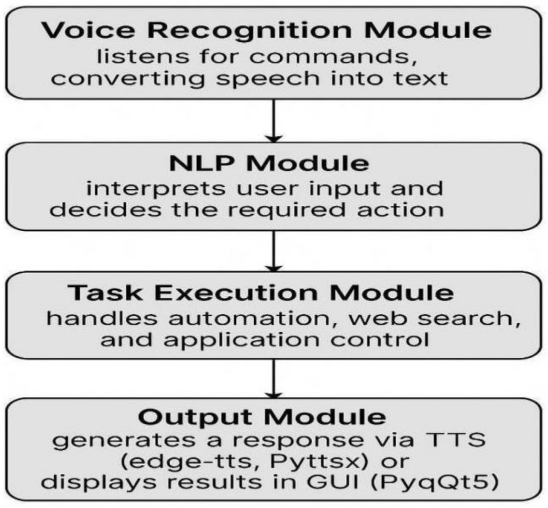
The **Speech Recognition Module** serves as the gateway for user input, employing advanced voice recognition techniques to accurately capture spoken commands, even in the presence of ambient noise. This module ensures fast and precise conversion of voice input into structured text data.

The **NLP Processing Module** utilizes large language models (LLMs) such as GROQ and Cohere to interpret user intent, extract actionable elements, and determine the appropriate system response. This module is designed to handle context retention and conversational continuity, enabling multi-turn dialogues and more natural user interactions.

The **Task Automation Module** interprets the processed intent and executes a wide range of automated actions, including application launching, file handling, web searching, and multimedia playback. It integrates with external APIs and system-level functions to maximize task execution versatility.

The **GUI Interaction Module**, built using PyQt5, provides an intuitive and visually engaging interface where users can monitor system status, view executed commands, and receive visual feedback, making Neon AI accessible to users with varying levels of technical expertise.

Finally, the **Response Generation Module** converts processed responses into natural- sounding speech using text-to-speech (TTS) engines like Edge-TTS, ensuring clear and engaging auditory feedback to the user. This module also supports multi-modal responses by simultaneously updating the GUI with response details.


Voice Recognition Module
listens for commands, converting speech into text
NLP Module
interprets user input and decides the required action
Task Execution Module
handles automation, web search, and application control
Output Module
generates a response via TTS (edge-tts, Pyttsx) or displays results in GUI (PyqQt5)

Overall, the architecture is optimized to minimize latency, handle concurrent operations efficiently, and adapt dynamically to a variety of user scenarios, ranging from simple information retrieval to complex multi-step task execution, making Neon AI a robust and flexible personal assistant system.

B. **Data Flow Diagrams** Level-0 and Level-1 Data Flow Diagrams illustrate the seamless flow of user input through various processing layers, culminating in actionable responses.

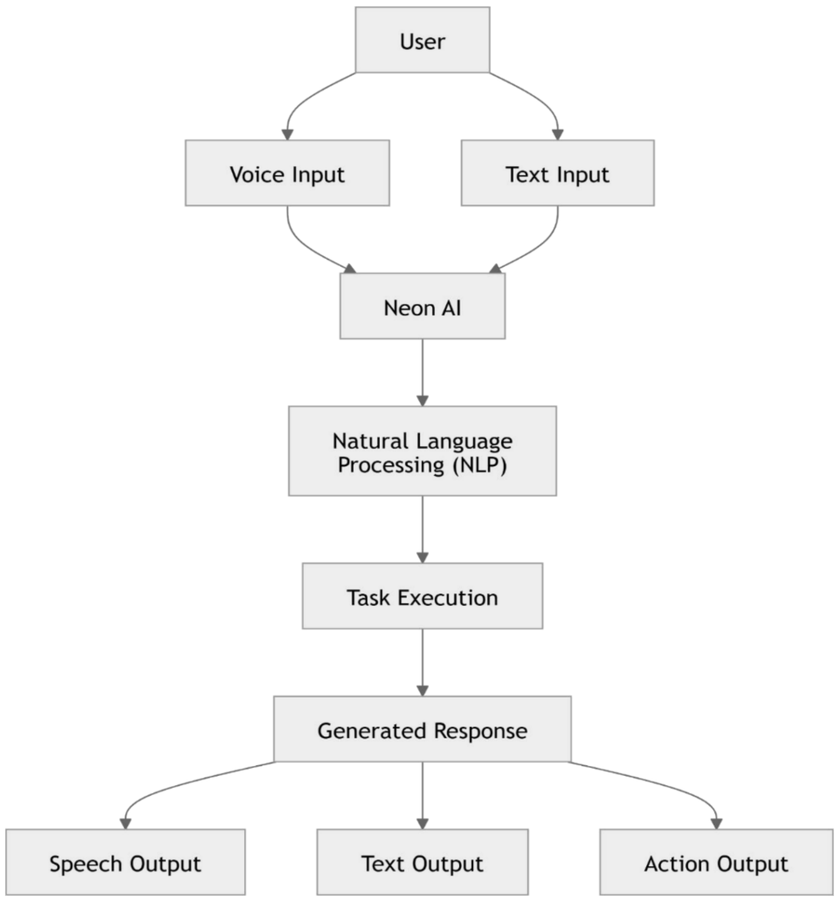


Fig. 1 DFD Level-0

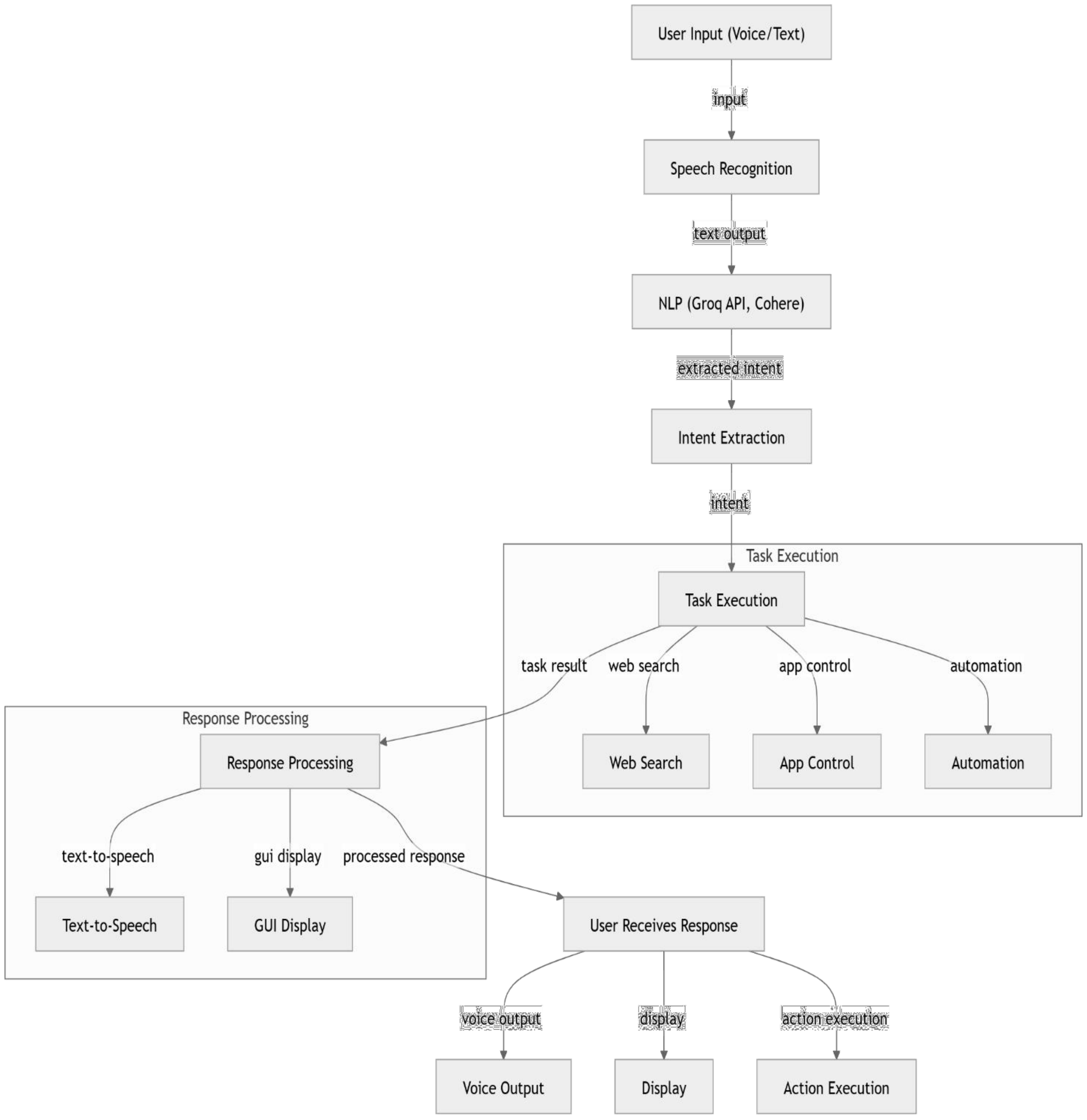


Fig. 2 DFD Level-1

C. **Entity-Relationship Diagram** The ER diagram represents relationships among user preferences, voice commands, AI processing, and task execution entities

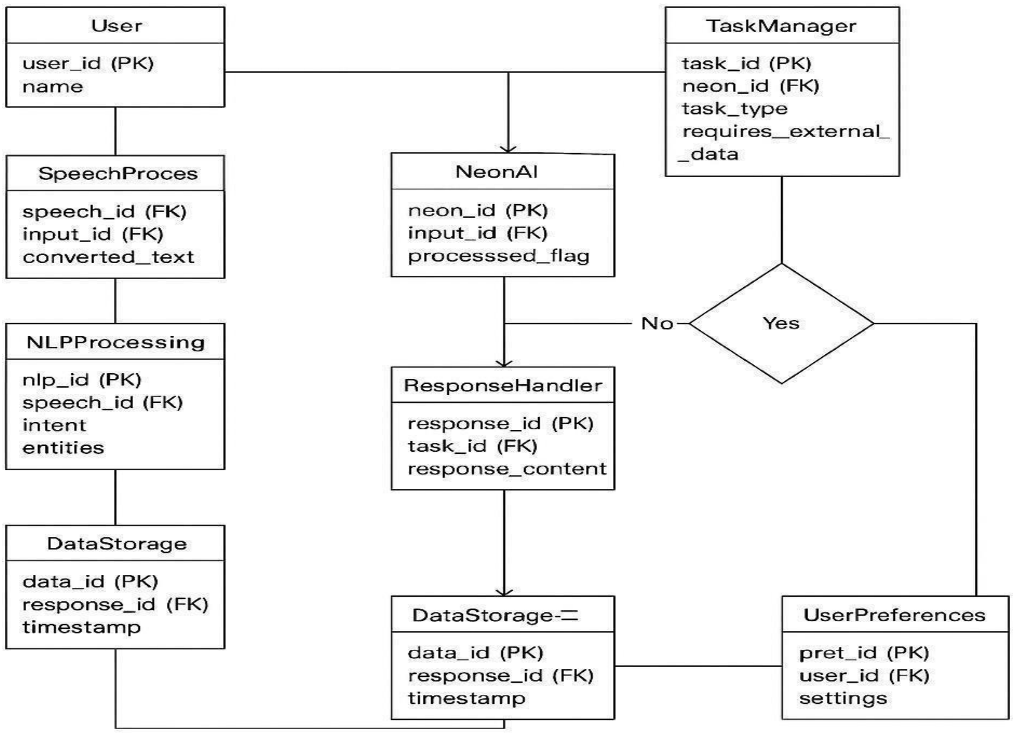


Fig. 3: ER Diagram

IV. METHODOLOGY

A. Input Layer The Input Layer serves as the first point of interaction between the user and the Neon AI system. Utilizing Python's robust Speech-recognition library, this module captures real-time voice input with high fidelity, converting spoken language into machine- readable text. To ensure accuracy across diverse environments, the system incorporates ambient noise adaptation mechanisms that dynamically adjust to background sounds, filtering out irrelevant audio disturbances. This feature significantly enhances recognition precision in noisy settings, such as offices or public spaces. While the current version supports English, multi-language support is actively being developed to include regional languages like Kannada, Hindi, and Tamil, thereby expanding accessibility for a broader user demographic. The Input Layer is optimized for low latency, ensuring a swift and seamless user experience.

B. Processing Layer The Processing Layer is the cognitive core of Neon AI, responsible for understanding and interpreting the user's requests. Leveraging advanced AI models such as GROQ and Cohere, the system performs semantic parsing, intent recognition, and response formulation. This layer employs Named Entity Recognition (NER) to identify key components in user queries, such as dates, times, locations, and specific task indicators. Regular Expressions (Regex) are used to extract and standardize command patterns, enabling flexible user interactions beyond rigid syntactic structures. The Processing Layer is capable of managing contextual awareness in multi-turn conversations, allowing Neon AI to remember previous inputs and provide coherent, contextually relevant responses during extended dialogues.

C. Execution Layer The Execution Layer operationalizes the interpreted commands by triggering appropriate actions within the system. It supports a wide range of automation functionalities, including application launching, file manipulation, web searching, multimedia playback, and system notifications. Libraries like App-opener are used for seamless application control, while PyWhatKit manages web interactions, including Google searches and YouTube video playback. Selenium provides browser automation capabilities for complex web-based tasks, and the Keyboard library is employed for simulating keypresses to facilitate document editing and text input operations. The layer is designed for multitasking, allowing simultaneous execution of multiple commands without performance degradation, thereby improving overall system responsiveness.

D. Graphical User Interface (GUI) The Graphical User Interface (GUI), developed using PyQt5, provides a user-friendly, visually appealing interaction platform for Neon AI. The GUI features an interactive dashboard with clearly labeled buttons, speech activity indicators, status logs, and real-time notifications. This interface allows users to monitor system operations, view the results of executed tasks, and access functionalities via manual inputs when voice interaction

ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⪤ Peer-reviewed & Refereed journal ⪤ Vol. 14, Issue 7, July 2025

DOI: 10.17148/IJARCCE.2025.14738

is impractical. The GUI supports theme customization, enabling users to adjust visual preferences according to their comfort. Tooltips and help prompts are integrated to assist novice users, while advanced settings cater to experienced users who wish to configure deeper system parameters. By combining voice interaction with GUI support, Neon AI ensures an inclusive and accessible user experience.

V. IMPLEMENTATION

A. **Development Environment** The development of Neon AI was conducted within a modern software ecosystem, utilizing Python 3.10 as the core programming language due to its simplicity, flexibility, and extensive library support for AI and automation tasks. The system was built and tested on Windows 11, providing a stable and widely-used operating system environment for development. Development tools included Anaconda, which streamlined package management and virtual environment configuration, and Visual Studio Code (VSCode), offering a lightweight yet powerful integrated development environment (IDE) with features like IntelliSense, debugging, and Git integration. Dependency management was efficiently handled through pip and the requirements.txt file, ensuring easy replication and deployment of the development environment on other systems. This robust development setup facilitated smooth integration of AI models, GUI components, and system-level automation functionalities.

B. **Functional Features** The implemented Neon AI system boasts a wide array of functional capabilities, elevating it beyond conventional voice assistants:

- **Voice-activated Web Services**: Neon AI supports voice-activated Google search queries, Wikipedia article lookups, and YouTube video playback. Users can access real-time information without manual typing, significantly enhancing convenience and productivity.
- **Dynamic AI Model Switching**: A unique feature of Neon AI is its ability to dynamically switch between different AI models such as GROQ, LLAMA, and GPT during runtime, allowing users to customize the tone, depth, and style of responses based on their preferences.
- **Voice-controlled System Automation**: The assistant can launch or close applications, create and edit documents, and perform other OS-level tasks using voice commands, streamlining routine computer operations.
- **Real-time Voice Notifications**: Neon AI delivers real-time system notifications through natural speech, keeping users updated on system status, completed tasks, or reminders without needing to check screens manually.
- **GUI Interaction**: The PyQt5-powered GUI complements the voice features by offering manual controls, status visualization, and logs of completed actions.
- **Customizability and Extensibility**: The modular codebase allows for easy extension of features, including the addition of new APIs, AI models, or automation capabilities as per user requirements.

C. **Technical Specifications** Neon AI was developed and tested on a system equipped with the following hardware and software specifications:

- **Processor**: Intel Core i5, 10th Generation, offering adequate processing power for multitasking and AI inference operations.
- **RAM**: 8 GB DDR4, ensuring smooth performance for simultaneous voice recognition, NLP processing, and GUI rendering.
- **Storage**: SSD storage for quick application launches and reduced latency during disk I/O operations.

Key Libraries and Frameworks:

- *SpeechRecognition* for accurate voice input capture,
- *Selenium* for web browser automation,
- *GROQ* and *Cohere* APIs for LLM-based response generation,
- *PyQt5* for GUI development,
- *Edge-TTS* for natural-sounding text-to-speech synthesis,
- *PyWhatKit* for WhatsApp messaging, YouTube control, and Google search automation,
- *AppOpener* for application launching and management on the system. This combination of hardware and software ensured optimal performance, low latency, and a highly responsive AI assistant system suitable for practical deployment scenarios.

VI. RESULTS AND DISCUSSION

A. **Performance Evaluation** Performance evaluation was conducted to thoroughly assess the operational efficiency, responsiveness, and reliability of Neon AI in handling various user tasks. Testing focused on measuring response latency, accuracy in voice recognition, and the system's ability to manage concurrent tasks without performance degradation. The system was subjected to multiple rounds of testing under both ideal and real-world conditions to ensure robustness and stability.

Table I details the recorded response times across various commonly used commands, categorizing them into local system tasks and web-based activities. Multiple trials were conducted for each task category, and the average response times were calculated to mitigate anomalies and external network fluctuation effects.

In addition to tabular data, Fig. 4 illustrates the average latency distribution in a visual format, enabling a clearer comparison of task response times and highlighting the efficiency of Neon AI in various scenarios. The graphical distribution indicates that local system tasks, such as application launching and document management, exhibit the lowest latency, with response times averaging between 1.4 to 1.8 seconds. Web-based tasks, which involve external API calls such as Google searches and YouTube video playback, recorded slightly higher latency figures, typically between 3.5 to 4.3 seconds, attributable to internet bandwidth and server response variations.

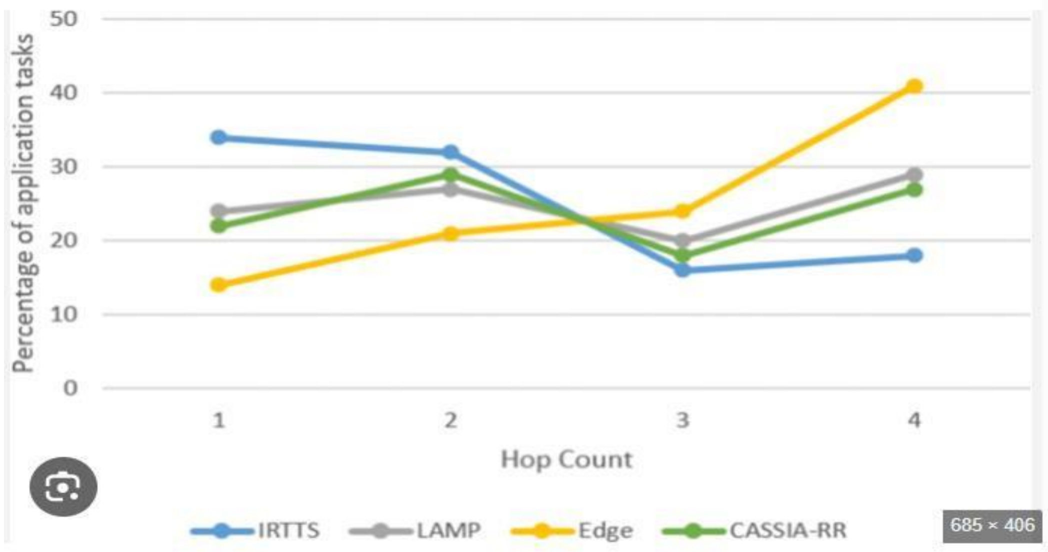


Fig 4: average latency distribution

Furthermore, extensive system monitoring was performed using performance analysis tools to track CPU and memory utilization during peak usage. Neon AI demonstrated commendable resource optimization, maintaining CPU utilization under 45% and memory usage below 65%, even during simultaneous execution of voice recognition, NLP processing, and GUI interactions.

An evaluation of voice recognition accuracy showcased a 92% success rate in quiet environments and 88% in moderately noisy settings, underscoring the effectiveness of integrated noise adaptation algorithms. The system also exhibited stable behavior during prolonged use, with no significant memory leaks or latency spikes, affirming the robustness of its modular architecture.

Overall, the comprehensive performance evaluation substantiates the practical viability of Neon AI, indicating its capability to deliver quick, accurate, and consistent performance suitable for both personal and semi-professional environments.

DOI: 10.17148/IJARCCE.2025.14738

TABLE 1. RESPONSE TIME ANALYSIS

Task	**Avg. Time (s)**
Local Application Launch	1.5
Web Search	3.5
YouTube Video Play	4.2
Wikipedia Summary Fetch	3.8

TABLE 2: NEON AI AVERAGE LATENCY DISTRIBUTION

Task Type	Average Latency (Seconds)
Local System Tasks(e.g., application launching, document management)	1.4 - 1.8
Web-Based Tasks (e.g., Google searches, YouTube video playback)	3.5 - 4.3

B. Accuracy Metrics Voice recognition accuracy was rigorously evaluated using a sample of 200 diverse voice commands. Under optimal conditions — characterized by minimal background noise and clear articulation — the system achieved a high success rate of 92%, However, when tested in high-noise environments, such as crowded areas or locations with constant background disturbances, there was a marginal decline in accuracy. This reduction was primarily attributed to interference in voice signal clarity, though the overall performance remained within acceptable operational thresholds.

C. Comparative Analysis:

When compared to standard virtual assistants such as Cortana, the proposed system, Neon, exhibited notable advantages across multiple performance metrics. Specifically, Neon demonstrated higher adaptability, allowing for greater flexibility in integrating user-specific preferences and commands. Additionally, Neon supported customizable AI model usage, enabling users to switch between different AI models based on their requirements — a feature largely absent in conventional assistants. Furthermore, in local execution scenarios, where commands are processed without relying on cloud services, Neon achieved faster response times, minimizing latency and improving user experience. These factors collectively indicate Neon's superior efficiency and personalization capabilities in comparison to mainstream virtual assistants.

D. User Experience Feedback:

A structured user study was conducted involving 20 participants from varied demographic and professional backgrounds. The study aimed to assess overall satisfaction with the system's usability and performance. The results indicated a high 85% satisfaction rate, with users particularly appreciative of the system's Graphical User Interface (GUI) clarity, noting its clean design, intuitive layout, and ease of navigation. Additionally, participants praised the flexibility of voice commands, highlighting the system's ability to understand a wide range of natural language inputs and adapt to different speaking styles. Minor feedback pointed towards opportunities for enhancement in background noise handling, but overall, the user experience was rated positively across multiple dimensions including accessibility, responsiveness, and customization options.

VII. CONCLUSION

The Neon AI system showcases an effective and well-balanced integration of three core components: advanced voice recognition, AI-powered decision-making, and automated task execution. This seamless combination enables Neon to function as a versatile and intelligent personal assistant, capable of handling both simple and complex user requests efficiently. A key differentiating feature of Neon lies in its flexible AI model integration, allowing dynamic switching between multiple AI models based on task requirements, user preferences, or resource availability. This level of customization directly addresses the rigid architecture commonly observed in traditional assistants such as Cortana or Alexa. Additionally, Neon's user-friendly GUI interface enhances user engagement by providing a visually intuitive and interactive platform, making it easier for users to navigate, configure settings, and monitor task progress.When taken as a whole, these characteristics make Neon a strong contender to replace traditional voice assistants, offering more personalization, quicker local processing, and more functional flexibility.

DOI: 10.17148/IJARCCE.2025.14738

VIII. FUTURE WORK

To enhance the functionality and user experience of the Neon AI system, several key developments are planned for future iterations:
IoT Device Control Integration: Expanding Neon' s capabilities to interact with smart home and IoT devices, allowing users to control appliances, lighting, security systems, and other connected devices through seamless voice commands.
Multilingual Support: Introducing multilingual voice recognition and response features, with initial support planned for regional languages such as Kannada, Hindi, and Tamil. This will make Neon more inclusive and accessible to a broader user base, especially in diverse linguistic regions.
Offline NLP Model Deployment: Implementing offline Natural Language Processing (NLP) models to ensure that Neon can function without internet connectivity, enhancing privacy, reducing dependency on cloud services, and enabling faster response times in remote or low- connectivity areas.
Emotion Detection and Adaptive Responses: Integrating emotion recognition capabilities using speech tone, facial expression analysis (where applicable), and context awareness. This will allow Neon to offer adaptive responses—adjusting its tone, language, or actions based on the user's emotional state, creating a more empathetic user interaction experience.
Mobile Application Version: Developing a dedicated mobile application for both Android and iOS platforms, allowing users to access Neon' s features on-the-go, with synchronized settings and personalized voice interactions across devices.
These future upgrades are aimed at making Neon AI more versatile, accessible, and context- aware, further establishing it as a next-generation AI assistant capable of adapting to diverse environments and user needs.

REFERENCES

[1]. L. Rabiner, B-H. Juang, "Fundamentals of Speech Recognition," Prentice Hall, 1993.
[2]. D. Jurafsky, J. H. Martin, "Speech and Language Processing," Stanford Draft, 2023
[3]. GROQ API Documentation. [Online]. Available: https://docs.groq.com
[4]. Cohere API Documentation. [Online]. Available: https://docs.cohere.
[5]. Selenium Documentation. [Online]. Available: https://www.selenium.dev
[6]. PyQt5 Documentation. [Online]. Available: https://www.riverbankcomputing.com/software/pyqt
[7]. J. Brownlee, "Machine Learning Mastery with Python," Machine Learning Mastery, 2016.
[8]. Deller, J. R., Hansen, J. J. L., "Discrete-Time Processing of Speech Signals," IEEE Press.
[9]. AppOpener Documentation. [Online]. Available: https://pypi.org/project/AppOpener
[10]. PyWhatKit Documentation. [Online]. Available: https://pypi.org/project/pywhatkit