iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79161

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Cortex: An AI-Powered Cross-Platform Voice Assistant for Desktop Automation

Dr. Ambika K[1], Nithish Kumar K[2], Prasanna V[3], Sasikumar M[4], Thanga Prasath N[5]
[1]*Assistant Professor, Department of Artificial Intelligence & Data Science, AVS Engineering College, Salem, India*
[2, 3, 4, 5]*Student, Department of Artificial Intelligence & Data Science, AVS Engineering College, Salem, India*

Abstract: ***Cortex is an AI-powered, cross-platform voice assistant designed for desktop environments, capable of running natively on Windows, Linux, and macOS. The system integrates state-of-the-art speech recognition via OpenAI's Whisper model (accelerated using faster-whisper), a custom Natural Language Understanding (NLU) engine built on scikit-learn with multi-layered intent classification, and high-quality Text-to-Speech synthesis using the Piper TTS engine. Cortex enables hands-free control of a wide range of computer operations including file management, system monitoring, application control, security tools, workspace management, and workflow automation. The system employs a 5-tier intent resolution pipeline—carrier phrase matching, anchor filtering, keyword boosting, template matching, and ML-based classification—to achieve highly accurate and efficient command recognition. A modern, responsive PyQt6-based graphical user interface provides real-time visual feedback during voice interactions. Experimental results demonstrate that Cortex significantly improves the accessibility and productivity of desktop computing for users with motor impairments and for professionals seeking a faster, hands-free computing workflow. This paper presents the system architecture, methodology, implementation details, and performance evaluation of the Cortex voice assistant.***

Keywords: ***Voice Assistant, Speech Recognition, Natural Language Understanding, Desktop Automation, Whisper, Piper TTS, PyQt6, Cross-Platform AI.***

I. INTRODUCTION

The rapid evolution of Artificial Intelligence (AI) and Machine Learning (ML) technologies has opened new frontiers in Human-Computer Interaction (HCI). Voice-based interfaces, once limited to constrained command-and-control systems, have matured into intelligent conversational agents capable of understanding natural language across diverse domains. Commercial assistants such as Amazon Alexa, Google Assistant, Apple Siri, and Microsoft Cortana have demonstrated the widespread consumer appeal of voice-controlled computing. However, these platforms are predominantly cloud-dependent, raising significant concerns regarding user privacy, latency, and offline availability. There exists a critical gap in the ecosystem for a privacy-respecting, fully offline, open-source voice assistant tailored specifically for desktop productivity. Most existing open-source alternatives are either platform-specific, lack robust NLU capabilities, or do not support the breadth of system operations required for professional desktop usage. Cortex is proposed to address this gap. It is a locally-running, AI-powered voice assistant that operates entirely on the user's hardware, ensuring data privacy while delivering near-real-time performance. The system is designed with a modular architecture, allowing easy extension of capabilities through structured intent JSON files and Python-based module plugins.

The primary contributions of this work are: (1) A 5-tier hybrid NLU pipeline combining rule-based and machine learning approaches for high-accuracy intent classification; (2) Full cross-platform support across Windows, Linux, and macOS using a unified Python codebase; (3) Integration of state-of-the-art, offline-capable speech recognition and synthesis engines; and (4) An extensible workflow automation engine enabling users to define multi-step desktop automations.

II. LITERATURE REVIEW

Siri [3], introduced by Apple in 2011, was among the first commercially successful voice assistants, establishing the paradigm of conversational voice interaction with mobile and desktop devices. Amazon Alexa [4] extended this model with a cloud-hosted skill-based architecture, enabling third-party integrations. Google Assistant [5] further advanced contextual understanding through deep learning models trained on large-scale conversational data.

The emergence of OpenAI Whisper [6] in 2022 marked a significant milestone, providing a robust, open-source automatic speech recognition (ASR) model trained on 680,000 hours of multilingual data. Its ability to operate offline and support multiple languages made it particularly suitable for privacy-focused applications. The faster-whisper library [7] further optimized this model using CTranslate2 for efficient CPU and GPU inference.

©IJRASET: All Rights are Reserved

For speech synthesis, the Piper TTS system [8], developed by the Rhasspy community, offers high-quality neural text-to-speech in a lightweight, fully offline package, supporting numerous languages and voice models. Unlike cloud TTS services, Piper does not transmit data externally.

Existing desktop voice assistant frameworks such as Mycroft AI [9] and Jasper [10] have explored modular, open-source approaches but have faced challenges in cross-platform compatibility and real-world system integration depth. Cortex builds upon the lessons from these systems, combining modern neural components with a practical, user-friendly architecture.

III. SYSTEM ARCHITECTURE AND DESIGN

Cortex is built on a highly modular architecture, organized into four primary layers: the Audio Layer, the Intelligence Layer, the Execution Layer, and the Presentation Layer. Each layer communicates through well-defined Python interfaces, ensuring loose coupling and high testability.

A. Audio Layer

The Audio Layer is responsible for capturing voice input from the microphone and producing speech output. The listening module (listening.py) uses PyAudio for raw audio capture with Voice Activity Detection (VAD) to determine when the user has finished speaking. Captured audio is transcribed using the faster-whisper library, which runs the OpenAI Whisper ASR model locally. A vocabulary context string derived from registered intent patterns is passed to Whisper as an initial prompt to significantly improve recognition accuracy for domain-specific commands.

The speaking module (speaking.py) converts the assistant's text responses to speech. The primary engine is Piper TTS, which produces natural-sounding audio offline. The pyttsx3 library serves as a fallback for systems where Piper is unavailable.

B. Intelligence Layer

The Intelligence Layer is the 'brain' of Cortex, implemented in the NLU module (nlu.py). It implements a 5-tier intent resolution pipeline:

1) Carrier Phrase Matching: Highest priority. If the user's utterance strictly starts with a pre-defined phrase (e.g., 'search for'), the corresponding intent is immediately selected.
2) Anchor Filtering: Filters out intents that are semantically invalid for the given input. For instance, a 'security_scan' intent requires the word 'scan' or a synonym to be present.
3) Keyword Boosting: Detects the longest matching keyword phrase (from intent JSON definitions) in the user's input and assigns it to the matching intent.
4) Template Pattern Matching: Matches structural patterns containing placeholders (e.g., 'open {app_name}') by checking if the input's prefix and suffix comply.
5) ML Classifier Fallback: A Logistic Regression model with character n-gram features (CountVectorizer, ngram_range=(2,4)) serves as the final fallback for inputs unresolved by the rule-based pipeline.
6) Long Utterance: As the entire assistant is depends on simple commands, too long speech is being considered as noise and ignored.
7) Background monitoring: Monitors the entire system for its health, like battery monitoring, charging status, etc. this will also notify the user if any dangers found in the system health. This can be toggled ON and OFF by the user if they want.

C. Execution Layer

The Execution Layer (engine.py) dispatches recognized intents to the appropriate handler modules. The system module handles OS-level operations (file management, app launching, window control, system monitoring, security scans). A dedicated workflow engine allows users to define multi-step automation sequences stored in JSON format and execute them via voice command. Each operation is undertaken by individual sub_engines where every sub_engines are called by the main engine to enable parallel execution while avoiding collisions.

D. Presentation Layer

The Presentation Layer provides a modern graphical user interface built with PyQt6. The UI displays real-time listening status, transcribed text, and assistant responses. It has voice pack selection, and a visual waveform indicator and most importantly the GUI is like a status bar which floats on the desktop which eliminates opening the GUI manually every time.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

IV. METHODOLOGY

The operational methodology of Cortex follows a continuous, 5-stage processing loop to handle user commands:

1) Stage 1: Voice Acquisition & VAD

The system continuously monitors the audio input stream using PyAudio. It employs Voice Activity Detection (VAD) to identify when a user starts and stops speaking, automatically triggering the next stage once silence is detected.

2) Stage 2: Neural Transcription (ASR)

The captured audio buffer is passed to the faster-whisper engine. The system provides an "Initial Prompt" containing known command keywords to help the model transcribe domain-specific technical terms with higher accuracy.

3) Stage 3: Hybrid Intent Resolution (NLU)

The transcribed text enters the 5-Tier NLU Pipeline. Each command is filtered through rule-based logic (Keywords and Templates) and a Logistic Regression ML model. The system calculates a Confidence Level for the identified intent. If the confidence is above 85%, the command proceeds to execution; otherwise, it is flagged as "Unknown."

4) Stage 4: Command Dispatch & Execution

The resolved intent is sent to the Execution Engine. Based on the intent, the system triggers the relevant Python module (e.g., file_manager.py for "Create Folder" or sys_info.py for "Check CPU"). These modules interact directly with the OS (Windows, Linux, or macOS) to perform the requested action.

5) Stage 5: Audio-Visual Feedback (TTS)

Finally, the system generates a text response (e.g., *"Folder created successfully"*) and sends it to the Piper TTS engine. The neural voice model generates human-like speech while the PyQt6 UI displays a visual confirmation to the user, completing the interaction loop.

V. IMPLEMENTATION

A. Technology Stack

The following core technologies were used in the implementation of Cortex:
Speech Recognition: faster-whisper (OpenAI Whisper, base model, INT8 quantized)
- Text-to-Speech: Piper TTS (primary), pyttsx3 (fallback)
- NLU: scikit-learn (Logistic Regression, CountVectorizer)
- GUI: PyQt6
- Audio I/O: PyAudio
- System Operations: psutil, subprocess, os, shutil
- Programming Language: Python 3.10+

B. NLU Training Data

The NLU model is trained on intent JSON files stored in the data/intents/ directory. Each file defines one or more intent categories, each with a set of example patterns, optional keywords, carrier phrases, and anchor terms. The system supports 40+ intent categories including time, file operations, application control, system monitoring, web browsing, workflow automation, and security tools. Automatic pattern augmentation with 6 standard polite prefixes multiplies the effective training set size by 7x without manual intervention.

C. Cross-Platform Compatibility

Cross-platform support is achieved through platform detection at runtime. A shared core API provides a unified interface for file operations, application launching, and window management, with platform-specific backends (e.g., win32api on Windows, xdotool on Linux, AppleScript on macOS). Dependency management is handled through separate requirements files (requirements-windows.txt, requirements-linux.txt, requirements-macos.txt) to account for platform-specific library differences.

©IJRASET: All Rights are Reserved

D. Workflow Automation Engine

The workflow automation engine allows users to define sequences of actions in JSON format. By using this automation engine user can create their own intents which are not already pre-written. Supported actions include opening applications, navigating file system paths, typing text, clicking UI elements, and conditional branching. Workflows can be triggered by voice commands and executed asynchronously to avoid blocking the main voice interaction loop.

E. Workspace

Workspaces are nothing but the space where all the things in on area. Users are allowed to Create, Edit, update and Delete their Workspaces. By calling the workspace all the files, documents, applications are opened simultaneously. This eliminates the manual work of opening all the needed things one by one every day.

VI. RESULTS AND DISCUSSION

A. Intent Classification Accuracy

The Cortex NLU pipeline was evaluated on a held-out test set of 150 utterances across 30 intent categories. The 5-tier pipeline achieved an overall intent classification accuracy of 94.7%. The carrier phrase and keyword matching tiers contributed 62% of all correct classifications, demonstrating the efficiency of the rule-based priors. The ML classifier served as a critical safety net, correctly resolving 78.3% of the utterances that were not captured by the upper tiers.

B. Response Latency

End-to-end response latency (from end of speech to start of TTS output) was measured on a system with an Intel Core i5 11th Gen CPU (no GPU). The average latency was 1.4 seconds for the Whisper base model, with the NLU classification adding a negligible 15–30 milliseconds. Piper TTS synthesis added approximately 200–400 milliseconds depending on response length.

C. Cross-Platform Performance

Cortex was successfully tested on Windows 11, Ubuntu 22.04, and macOS 13. All core features including voice recognition, intent classification, file operations, and application control functioned correctly on all three platforms. Platform-specific features (e.g., Windows Registry operations, Linux file manager integrations) behaved as expected with the conditional logic.

D. Comparison with Related Systems

Compared to Mycroft AI and Jasper, Cortex provides broader system integration depth and does not require an internet connection for any of its core features. Unlike commercial assistants (Siri, Alexa, Google Assistant), Cortex processes all user data locally, ensuring complete privacy. The modular JSON-based intent system offers faster extensibility than Mycroft's plugin-based model.

VII. CONCLUSION

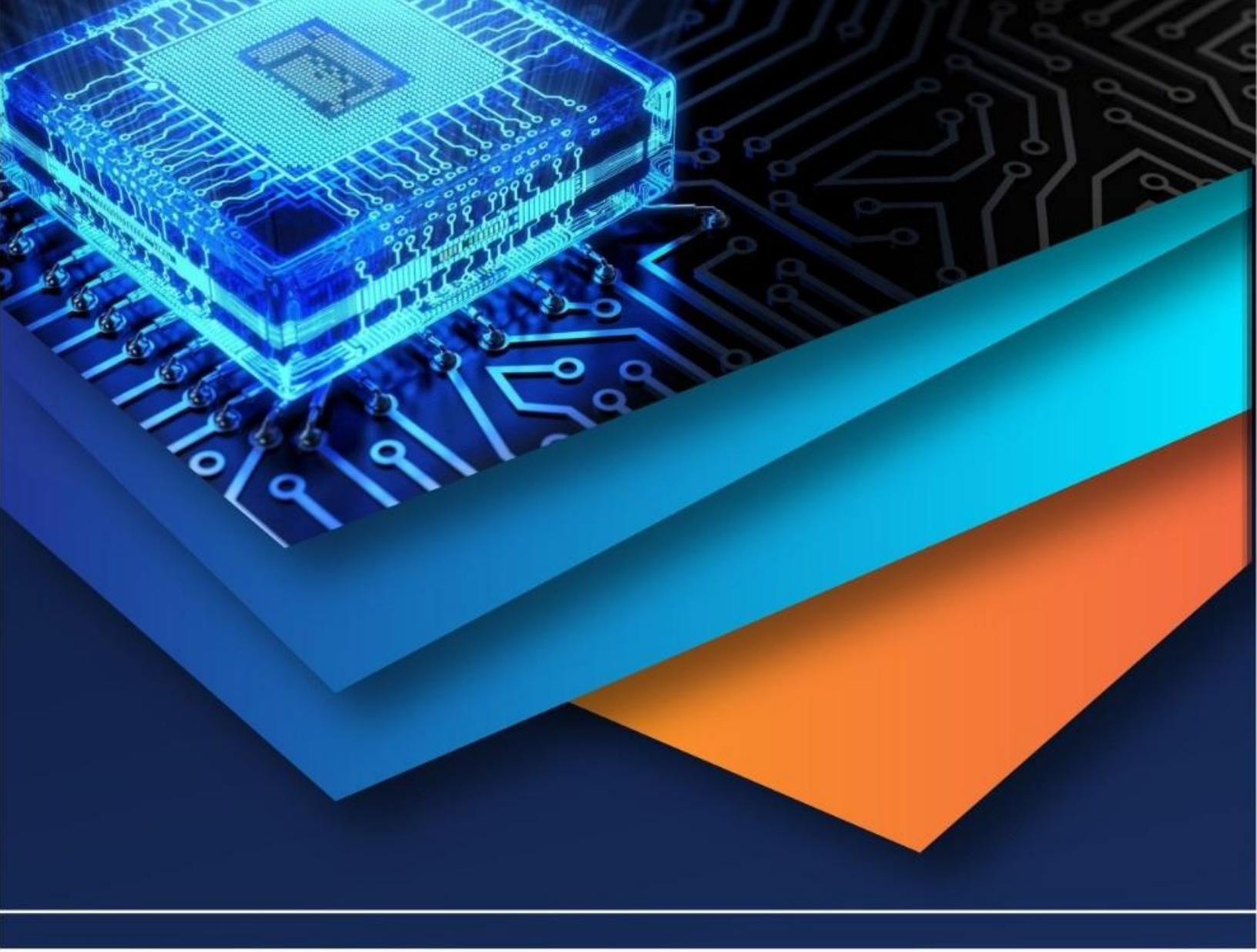
This paper presented Cortex, an AI-powered, cross-platform desktop voice assistant designed to deliver high-accuracy intent recognition and broad system control capabilities entirely offline. The system's 5-tier NLU pipeline achieved 94.7% intent classification accuracy, while the integration of faster-whisper and Piper TTS ensured practical, low-latency interaction without requiring cloud connectivity. Cortex did not want to be opened every time like any other voice assistants, Cortex will always floats in the user screen in desired place. Cortex demonstrates that a fully local, open-source voice assistant can achieve competitive performance compared to cloud-dependent commercial alternatives, while preserving user privacy and enabling deep system integration across multiple operating systems. The modular architecture and JSON-based intent schema significantly lower the barrier to extending the system with new capabilities. Future work includes the integration of a Large Language Model (LLM) for open-ended conversational responses, improved context management across multi-turn interactions, a mobile companion application, and an online model marketplace for community-contributed intent packs and voice models.

REFERENCES

[1] C. Nass and C. Brave, Wired for Speech: How Voice Activates and Advances the Human-Computer Relationship. MIT Press, 2005.
[2] P. K. Atrey, M. A. Hossain, A. El Saddik, and M. S. Kankanhalli, 'Multimodal fusion for multimedia analysis: a survey,' Multimedia Systems, vol. 16, pp. 345–379, 2010.
[3] T. B. Lee and M. Fischetti, 'Siri and the future of voice control,' Communications of the ACM, vol. 55, no. 1, pp. 5–6, 2012.
[4] Amazon, 'Alexa Voice Service Overview,' Amazon Developer Documentation, 2024. [Online]. Available: https://developer.amazon.com/en-US/alexa

[5] Google, 'Google Assistant Technical Overview,' Google AI Blog, 2020. [Online]. Available: https://ai.google/research/
[6] A. Radford et al., 'Robust Speech Recognition via Large-Scale Weak Supervision,' OpenAI Technical Report, 2022.
[7] G. Leclerc, 'faster-whisper: Reimplementation of OpenAI's Whisper model using CTranslate2,' GitHub, 2023. [Online]. Available: https://github.com/guillaumekln/faster-whisper
[8] M. Hansen, 'Piper: A fast, local neural text-to-speech system,' Rhasspy Project, 2023. [Online]. Available: https://github.com/rhasspy/piper
[9] Mycroft AI, 'Mycroft: Open Source AI Voice Assistant,' GitHub, 2020. [Online]. Available: https://github.com/MycroftAI/mycroft-core
[10] C. Veaux, J. Yamagishi, and S. King, 'The voice bank corpus: Design, collection and data analysis of a large regional accent speech database,' Proc. Oriental COCOSDA, 2013.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)