RESEARCH PAPER

Leveraging LiveKit for Real-Time AI-Driven Conversational Agents

Somil Shekhar[1], Saniya Ahmad[1], Prathamesh Pawar[1], Rajnish Patel[1], Dr. Hina J Chokshi[1*]

[1]Parul Institute of Computer Applications, Parul University, Vadodara, Gujarat, India

***Corresponding author: Dr. Hina J Chokshi, Parul Institute of Computer Applications, Parul University, Vadodara, Gujarat, India**

Authors: Somil Shekhar, Saniya Ahmad, Prathamesh Pawar, Rajnish Patel - Students of Parul Institute of Computer Applications

Received: 30th May, 2026; **Revised:** 8th June, 2026; **Accepted:** 10th June, 2026; **Available Online:** 12th June, 2026

Abstract:

Recent progress in real-time communication frameworks has significantly advanced the development of multimodal intelligent assistants that can interact with users in a natural, human-like manner. This paper introduces a LiveKit-based AI Assistant designed to enable seamless, low-latency human-computer interaction by integrating real-time audio streaming, large language model (LLM) reasoning, and automated system control within a unified architecture. The proposed framework utilizes LiveKit Agents for real-time connectivity, employs noise suppression plugins for robust speech processing, and leverages LangChain-based orchestration to ensure contextual understanding and reasoning [1], [2].

In addition, the assistant incorporates a versatile toolkit for web-based search, voice-command interpretation, and operating system-level automation, enabling it to function effectively across diverse platforms and environments. Experimental evaluations indicate that the system achieves high responsiveness and scalability while maintaining lightweight deployment characteristics. Compared to existing conversational agents such as traditional virtual assistants, the proposed model offers superior customizability, cross-platform adaptability, and real-time feedback performance. This research underscores the potential of LiveKit-powered multimodal agents in shaping the future of real-time interactive AI systems suitable for both academic exploration and real-world deployment [3], [4].

Keywords-

AI Assistant, LiveKit, Real-Time Communication, Multimodal Interaction, Google Realtime LLM, Natural Language Processing, Desktop Automation, Context-Aware Systems, Human–Computer Interaction

How to cite this article: Shekhar S, Ahmad S, Pawar P, Patel R, Chokshi HJ. Leveraging LiveKit for Real-Time AI-Driven Conversational Agents. Int J Drug Deliv Technol. 2026;16(58s): 1246-1250. DOI: 10.25258/ijddt.16.58s.131

Source of support: Nil.

Conflict of interest: None

Introduction:

The rapid advancement of remote collaboration technologies, automation, and intelligent human-computer interfaces has led to an increasing demand for real-time, voice-driven AI assistants capable of natural, context-aware interaction [5], [6]. While popular commercial assistants such as Google Assistant, Amazon Alexa, and Apple Siri have achieved mainstream adoption, they often face limitations such as platform dependency, restricted customization, data privacy concerns, and limited interoperability with third-party or open-source systems [7].

Moreover, existing architectures frequently struggle to maintain a balance between low-latency communication and robust multimodal performance, particularly in environments with high ambient noise or when system-level automation is required. This trade-off often results in delayed responses, reduced contextual accuracy, and suboptimal user experience [8].

To address these limitations, this paper presents a LiveKit-based AI Assistant that seamlessly integrates real-time audio streaming, advanced noise suppression, and large language model (LLM) reasoning with system-level control and automation. The framework employs LiveKit Agents for high-speed, low-latency communication, while open-source modules handle speech-to-text (STT), text-to-speech (TTS), and noise reduction tasks efficiently [9]. Additionally, the use of LangChain-based orchestration enables dynamic contextual reasoning and flexible integration with multiple tools and APIs.

Beyond traditional assistants, the proposed system supports desktop automation, web search operations, and multimodal interactions, empowering users to execute commands and perform tasks naturally in real time. Overall, this work demonstrates that combining real-time communication frameworks with adaptive AI orchestration can yield a lightweight, extensible, and scalable architecture, capable of overcoming the performance and flexibility constraints inherent in existing commercial systems.

Literature Review

The evolution of intelligent voice assistants has been a focal area of research, primarily driven by rapid progress in speech recognition, natural language processing (NLP), and human–computer interaction (HCI). Commercial platforms such as Google Assistant, Amazon Alexa, and Apple Siri have made voice-based interfaces widely accessible; however, their operation within closed ecosystems imposes significant limitations on extensibility, interoperability, and the ability to integrate with external tools or perform desktop-level automation [10].

Several research efforts have explored multimodal assistants capable of processing audio, visual, and textual inputs concurrently. Frameworks built on WebRTC-based communication have demonstrated promising results in enabling low-latency, real-time collaboration [11]. Despite their efficiency in handling synchronous communication, these solutions often lack coupling with task orchestration frameworks, thereby constraining their capacity to execute multi-step commands or manage complex workflows.

Recent advancements have shifted towards agent-based architectures for real-time AI assistants. LiveKit Agents, for instance, offer a scalable foundation for managing audio–video sessions while supporting the integration of custom plugins for speech recognition, text-to-speech synthesis, and noise suppression [12]. This modular design grants developers flexibility in extending functionalities to include API integrations, system-level automation, and context-driven reasoning.

Parallelly, studies on LangChain and similar orchestration frameworks have introduced structured approaches for coordinating multiple tools and APIs to achieve context-aware decision-making and task execution [13]. Such orchestration mechanisms enhance the assistant's ability to process sequential instructions, thereby improving reliability in real-world applications.

Nevertheless, a research gap persists: only a few existing frameworks successfully combine real-time voice communication, adaptive LLM reasoning, noise suppression, and system-level automation within a unified deployable system. The proposed LiveKit-based AI Assistant aims to bridge this gap by integrating:

- Real-time audio and video communication through LiveKit
- Advanced noise suppression and speech-to-text processing for improved command interpretation
- LLM-driven contextual reasoning for intelligent response generation
- Comprehensive system-level automation tools for managing desktop operations, applications, and files

This synthesis enables a scalable, customizable, and interactive assistant capable of bridging the gap between traditional voice assistants and programmable, task-oriented AI agents.

Related Work:

Over the past decade, the domain of real-time AI assistants has experienced rapid evolution, primarily fueled by advances in natural language processing (NLP), speech recognition, and human–computer interaction (HCI). Widely adopted commercial systems such as Google Assistant, Amazon Alexa, and Apple Siri have showcased effective voice-command capabilities; however, their operation within closed-source ecosystems restricts extensibility, external integration, and task-level customization [10].

A number of studies have investigated multimodal assistants that simultaneously process audio, visual, and textual inputs to enhance interaction quality. Frameworks built upon WebRTC-based communication models have proven effective in supporting low-latency, real-time collaboration [14]. Nevertheless, these architectures often lack seamless integration with task orchestration frameworks or system-level automation layers, which limits their applicability in scenarios requiring desktop control or custom workflow execution.

Recent research in agent-based AI architectures has underscored the potential of frameworks such as LiveKit in delivering scalable, low-latency voice and video communication solutions [12]. LiveKit Agents, in particular, enable developers to attach modular plugins and tools for dynamic task execution, bridging the gap between conventional conversational assistants and programmable, adaptive AI agents. Furthermore, works exploring LangChain and similar orchestration systems have demonstrated effective coordination between external APIs, reasoning modules, and contextual processing pipelines, thereby improving an assistant's capacity for multi-step reasoning and context-aware decision-making [13].

Despite these advancements, a significant gap remains in designing a unified, deployable framework that integrates real-time communication, adaptive language reasoning, noise suppression, and system-level automation. The proposed approach addresses this challenge by combining LiveKit Agents with open-source speech processing modules and automation tools, resulting in a scalable, extensible, and context-aware AI assistant capable of performing diverse system and conversational tasks in real time.

System

The proposed AI assistant, Arshii, is designed as a real-time, multimodal voice-based assistant using LiveKit Agents. The system integrates speech processing, natural language understanding, real-time communication, and system-level automation tools in a unified architecture.

A. Architecture Overview

The system comprises the following components:

1. LiveKit Agents & Sessions

- AgentSession: Manages real-time voice and video communication between user and assistant.
- RoomInputOptions: Provides noise cancellation and video feed handling, ensuring high-quality input for processing.

2. Natural Language Understanding & LLMs

- Google Realtime Model: Processes user queries, interprets context, and generates intelligent responses.
- Behavior Prompts & Reply Prompts: Define the assistant's persona, tone, and conversation style, ensuring friendly yet professional interaction.

3. Tool Integration Layer

- Search Tools: Google Custom Search and DuckDuckGo API for information retrieval.
- Weather Tool: Fetches real-time weather data using OpenWeather API.
- File and System Automation: Opens/closes applications, manages files/folders, plays media, and interacts with desktop environments.
- Keyboard and Mouse Control: Enables cursor movement, clicks, scroll, typing, hotkeys, swipe gestures, and volume control through PyAutoGUI and Pynput.

4. Noise Cancellation & Audio Processing

- Silero VAD & Noise Cancellation plugins remove background noise, enhancing speech recognition accuracy.

5. Task Orchestration

Integrates LangChain or similar orchestration frameworks to coordinate multiple tool executions and maintain context-aware task handling.

Methods

The development of the real-time AI assistant involves several sequential processes, ranging from data capture to response generation and task execution. The overall methodology emphasizes real-time interaction, context-awareness, and seamless integration of multiple functional modules.

A. Data Capture & Preprocessing

1) Real-Time Audio/Video Input:
User interactions are conducted through a LiveKit Room, which continuously streams audio and video data. The system employs the livekit-plugins-noise-cancellation module to reduce background noise and enhance speech clarity for accurate processing.

2) Speech-to-Text Conversion:
The audio stream is processed using the Google Realtime LLM API, which provides high-accuracy voice transcription. The transcribed text undergoes preprocessing, including normalization, punctuation correction, and removal of artifacts, to prepare it for intent recognition.

B. Intent Recognition & Feature Extraction

1) Command Parsing:
The transcribed input is analyzed to extract key linguistic features such as action verbs, entities, and command structures. A custom parsing mechanism maps these elements to corresponding system functions (e.g., "open Chrome," "get weather," "play video").

2) Tool Mapping:
Each recognized intent is dynamically linked to a specific function_tool defined within the LiveKit framework. Key mappings include:

- Search & Information Retrieval: google_search, get_current_datetime, get_weather
- File & Application Management: open, close, folder_file, play_file
- Keyboard/Mouse Control: move_cursor_tool, mouse_click_tool, type_text_tool, press_key_tool, etc
  .

3) Context Management:
The system maintains conversational state and session memory to support contextual understanding. It integrates predefined behavioral and response templates that help sustain a consistent tone, personality, and interaction style throughout the session.

C. Task Execution

1) Asynchronous Tool Invocation:
All tool executions are handled asynchronously using the asyncio event loop to ensure low-latency, non-blocking operations. This design enables simultaneous command execution while preserving system responsiveness.

2) Desktop Automation:
Modules responsible for system-level operations—such as file management, application control, and input automation—are executed securely with proper logging mechanisms. The SafeController class ensures that all automation tasks are authorized, monitored, and automatically deactivated after completion.

3) Error Handling & Feedback:
Each action generates structured feedback, including success or failure messages. The results are immediately delivered to the user via the LiveKit interface, ensuring transparency and traceability of system operations.

D. Response Generation

1)LLM Integration:
The Google Realtime Model is employed to generate natural, coherent, and contextually relevant responses. These outputs are guided by predefined conversational prompts to maintain a human-like, adaptive dialogue style.

2) Result Formatting:

Task Type	Tasks Tested	TSR (%)	Avg. Response Time (s)	Comments
Information Retrieval	50	96	1.2	Fast, accurate results from Google and weather APIs
System Automation	40	92	1.5	Minor delays with large folder indexing
Cursor/Keyboard Control	30	90	1.8	All gestures executed; occasional misalignment in swipe
Conversational Interaction	20	95	1.1	Persona maintained, context-aware responses

Outputs from APIs, local executions, or other modules are formatted for clarity and coherence before being relayed to the user. The system ensures that each response is concise, informative, and conversationally natural.

Experiments

A. Experimental Setup

1. Hardware & Software Environment
   - Hardware: Intel i7 CPU, 16GB RAM, Windows 10
   - Software: Python 3.11, LiveKit, Google Realtime LLM, PyAutoGUI, Pynput
   - APIs: Google Custom Search, OpenWeather, DuckDuckGo
2. Network Configuration
   - Experiments performed over stable Wi-Fi and LAN environments to simulate real-time communication.
   - LiveKit handles voice/video streaming with noise cancellation enabled.
3. Test Scenarios
   - Information Retrieval: Queries like "What's the weather in Delhi?" or "Latest news about AI"
   - System Automation: Commands such as "Open Chrome", "Play music",
   - "Create folder ArshiiTest"
   - Cursor & Keyboard Control: Mouse movement, clicks, typing, hotkeys, swipe gestures
   - Conversational Interaction: Casual dialogues to evaluate persona adherence and contextual understanding

B. Evaluation Metrics

1. Task Success Rate (TSR):

$$TSR = \frac{\text{Number of successfully executed tasks}}{\text{Total tasks attempted}} \times 100$$

2. Response Time:
   - Measured from voice input capture to response delivery (in seconds).
3. User Satisfaction Score (USS):
   - Qualitative assessment based on clarity, tone, and helpfulness of responses. Scale: 1–5.
4. Accuracy of Intent Recognition:
   - Percentage of commands correctly mapped to appropriate tools/actions.

C. Results

Conclusion:

This paper presents the development of an advanced AI-powered desktop assistant capable of real-time multimodal interaction through speech, video, and automated system control. By leveraging the LiveKit framework for low-latency communication, Google Realtime LLM for natural language understanding, and an integrated suite of automation tools, the system demonstrates the ability to understand context, execute desktop-level tasks, and engage in intelligent conversation simultaneously.

The modular architecture ensures scalability and adaptability, allowing the integration of additional tools and APIs without disrupting the system's core functionality. Experimental results validate that the assistant performs tasks efficiently while maintaining conversational fluency and contextual relevance.

Future enhancements may focus on improved emotion recognition, adaptive learning for personalized responses, and broader platform compatibility. Overall, this work contributes to the growing domain of intelligent human–computer interaction by demonstrating a real-time, context-aware assistant capable of bridging the gap between conversation and action.

References:

[1] T. Chen, "Real-Time AI Agents for Multimodal Interaction," *IEEE Access*, vol. 12, pp. 23340–23352, 2024.

[2] H. Zhang et al., "LangChain Framework for

Contextual Reasoning in LLMs," *Proc. of AAAI Conference on Artificial Intelligence*, 2024.
[3] A. Gupta and J. Li, "Low-Latency Communication in LiveKit Architectures," *IEEE Transactions on Network Systems*, vol. 11, no. 3,2023.

[4] S. Patel, "Cross-Platform AI Assistants Using Live Audio Streams," *International Conference on Intelligent Computing and Systems*, 2024.
[5] M. Hasan et al., "Advances in Multimodal AI Assistants for Real-Time Interaction," *IEEE Access*, vol. 12, pp. 34122–34135, 2024.

[6] R. Li and S. Bhatia, "Human-Computer Interaction through Real-Time Voice AI Systems," *Proc. of the IEEE International Conference on Emerging Computing Technologies*, 2023.

[7] J. Torres et al., "Privacy and Integration Challenges in Commercial Voice Assistants," *ACM Computing Surveys*, vol. 55, no. 6, pp. 1–24, 2023.

[8] K. Nair, "Optimizing Latency and Noise Suppression in Multimodal AI," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2101–2115, 2023.

[9] A. Gupta and L. Chen, "LangChain-Orchestrated AI Agents for Real-Time Automation," *International Conference on Artificial Intelligence Systems (ICAIS)*, 2024.
[10] Google LLC, "Google Assistant Overview," 2023.

[11] A. Smith et al., "Low-latency WebRTC Frameworks for Collaborative AI Applications," *IEEE Access*, vol. 11, pp. 12145–12158, 2023.

[12] LiveKit, "LiveKit Agents Documentation," 2024.

[13] LangChain, "LangChain Framework for Tool Orchestration," 2024.

[14] A. Smith et al., "WebRTC-based Low-Latency Frameworks for Real-Time Collaboration," *IEEE Transactions on Multimedia*, vol. 25, pp. 1452–1463, 2023