www.ijarp.com ISSN 2456-9992 Page: 01-10

International Journal Advanced Research Publications

Research Article

Volume: 02
Issue: 04

REAL TIME VOICE ASSISTANT

***Keshav Siwach, Ishaan Kaushik, Harshit Pal, Nikhil Siwach, Sanjeev Raghav**

Department of Computer Science and Engineering, R.D. Engineering College, Duhai, Uttar Pradesh-201206.

Article Received: 16 March 2026, Article Revised: 06 April 2026, Published on: 26 April 2026

***Corresponding Author: Keshav Siwach**

Department of Computer Science and Engineering, R.D. Engineering College, Duhai, Uttar Pradesh-201206.

DOI: https://doi-doi.org/101555/ijarp.9787

ABSTRACT

This research paper presents the design and implementation of a real-time voice interaction system that enables natural conversational communication between users and an artificial intelligence model. The system is developed using OpenAI's Realtime API combined with WebRTC technology to enable low-latency audio communication.

The proposed architecture integrates a React TypeScript frontend with a FastAPI Python backend, creating a scalable and efficient framework for real-time voice-based AI interactions. The application captures user speech through the browser microphone, transmits audio streams to the AI model for processing, and returns AI-generated responses in real time.

The system emphasizes low latency, seamless communication, and an intuitive user interface, allowing users to interact with the AI through natural speech. By leveraging modern web technologies and peer-to-peer communication protocols, the system significantly reduces response delays and improves conversational efficiency. This research demonstrates the potential of real-time AI voice systems in modern human-computer interaction.

1. INTRODUCTION

Artificial Intelligence has significantly transformed the way humans interact with digital systems. Traditional interaction methods such as typing commands or clicking interfaces are gradually being replaced by voice-based communication systems. Real-time voice assistants enable more natural and efficient interactions by allowing users to communicate with machines through speech.

Recent advancements in AI models, real-time APIs, and browser technologies have made it possible to build sophisticated conversational systems capable of processing speech in real

time. Technologies such as WebRTC allow direct peer-to-peer communication with extremely low latency, making them ideal for voice-based applications.

This research focuses on developing a Real Time Voice Assistant capable of providing real-time conversational responses. The system uses OpenAI's Realtime API for speech processing and AI response generation while utilizing WebRTC technology to establish direct audio communication between the client and the AI infrastructure.

The primary objective of this research is to design a low-latency, scalable, and efficient voice interaction system that allows users to communicate with AI seamlessly through spoken language.

2. System Workflow

The system follows a structured workflow that enables seamless communication between the user and the AI model.

2.1 Conversation Flow

The conversation flow follows a systematic sequence of steps from initiation to response delivery:

Figure 1: Conversation Flow Diagram.

This workflow ensures a continuous interactive dialogue between the user and the AI system while maintaining minimal latency.

3. System Architecture

3.1 High-Level Architecture

The proposed system follows a three-tier architecture consisting of a frontend client, a backend server, and an AI processing layer.

The React frontend provides the user interface and captures microphone input. The FastAPI backend manages authentication, session creation, and communication with the AI services. After the session is established, WebRTC connections enable direct communication between the client and the AI server, significantly reducing latency.

This architecture improves performance by minimizing intermediate server processing during audio transmission.

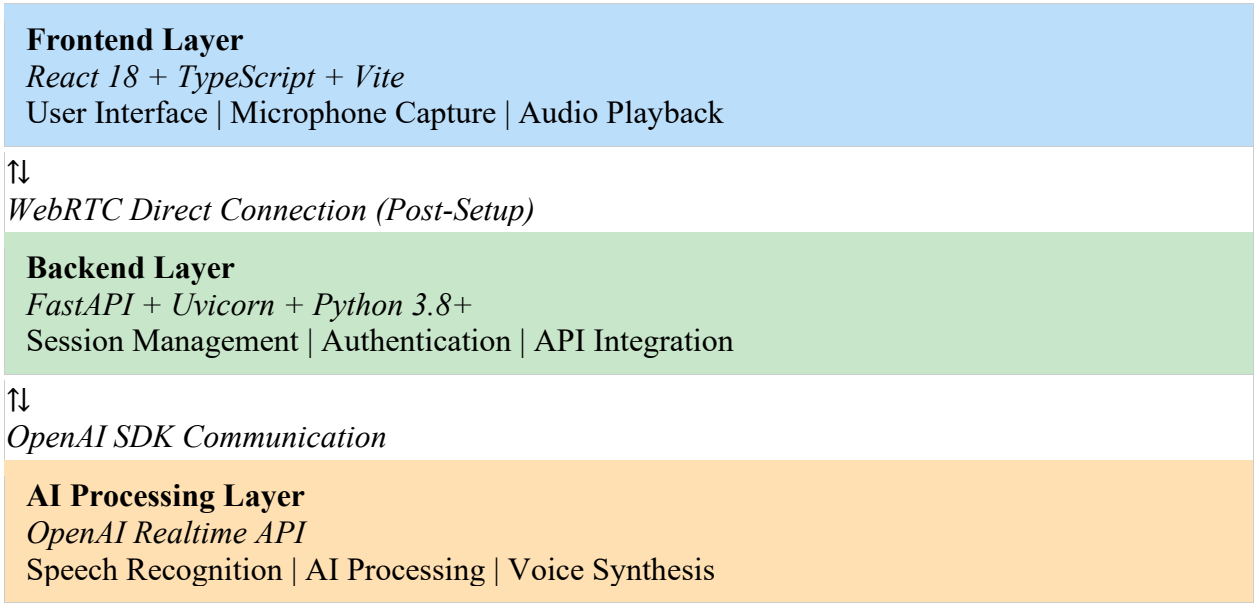

Figure 2: Three-Tier System Architecture.

Direct WebRTC connection minimizes latency after initial session setup

3.2 Communication Protocol

The communication process consists of several stages:

Initial Setup

The frontend application sends a request to the backend server to create a temporary session for AI communication.

Authentication

The backend validates the API credentials and generates an ephemeral session key that allows temporary access to the AI services.

WebRTC Connection

A WebRTC handshake is initiated to establish a secure peer-to-peer communication channel between the client and the AI infrastructure.

Real-time Communication

Once the connection is established, audio streams are transmitted in both directions with latency typically below 100 milliseconds, enabling natural conversational interactions.

4. Technology Stack

4.1 Backend Infrastructure

The backend is implemented using Python 3.8+ with the FastAPI framework, which is known for its high performance and asynchronous processing capabilities. FastAPI also provides automatic API documentation, improving development efficiency.

The backend server runs on Uvicorn, an ASGI server capable of handling concurrent connections efficiently. The integration with OpenAI services is achieved through the OpenAI Python SDK, which simplifies API communication and response handling.

Additionally, Pydantic is used for data validation and schema enforcement, ensuring reliable communication between the frontend and backend systems.

4.2 Frontend Technology

The frontend interface is developed using React 18 combined with TypeScript, enabling type-safe component development and improved code maintainability.

The application uses Vite as the build tool, providing fast development server performance and optimized production builds. The user interface is styled using Tailwind CSS, which enables rapid UI development using a utility-first design approach.

For visual elements and icons, the system utilizes Lucide React, providing a modern and consistent user interface design.

4.3 Real-Time Communication Layer

The real-time communication layer is built using the WebRTC API, which allows peer-to-peer audio streaming directly within the browser. WebRTC ensures minimal latency and includes advanced features such as:

- Connection state monitoring
- Automatic reconnection handling

- Efficient audio streaming
- Network interruption recovery

This technology enables the system to deliver smooth and uninterrupted voice interactions.

5. System Workflow and Data Flow

5.1 Session Lifecycle Management

The application follows a structured session lifecycle beginning with user interaction. When the user initiates a session, the backend generates temporary authentication keys with configurable expiration times.

These ephemeral sessions enhance security by limiting the duration of access to the AI services. The session configuration also includes parameters such as:

- Voice configuration
- Speech detection thresholds
- Conversation settings

These parameters can be dynamically adjusted during runtime.

5.2 Audio Processing Pipeline

The system processes audio using a structured pipeline.

First, the user's speech is captured through the browser using the getUserMedia API. The captured audio is encoded in PCM16 format, ensuring high-quality audio transmission.

The audio stream is then transmitted to the AI system through the WebRTC connection, where it is processed in real time. The AI generates a response, which is transmitted back to the user through the same communication channel.

This pipeline enables natural conversational interactions with minimal delay.

Table 1: Complete Technology Stack.

Layer	Technologies
Frontend	React 18, TypeScript, Vite, Tailwind CSS, Lucide React
Backend	Python 3.8+, FastAPI, Uvicorn, Pydantic, OpenAI SDK
Communication	WebRTC API, getUserMedia API, PCM16 Audio Encoding
AI Processing	OpenAI Realtime API

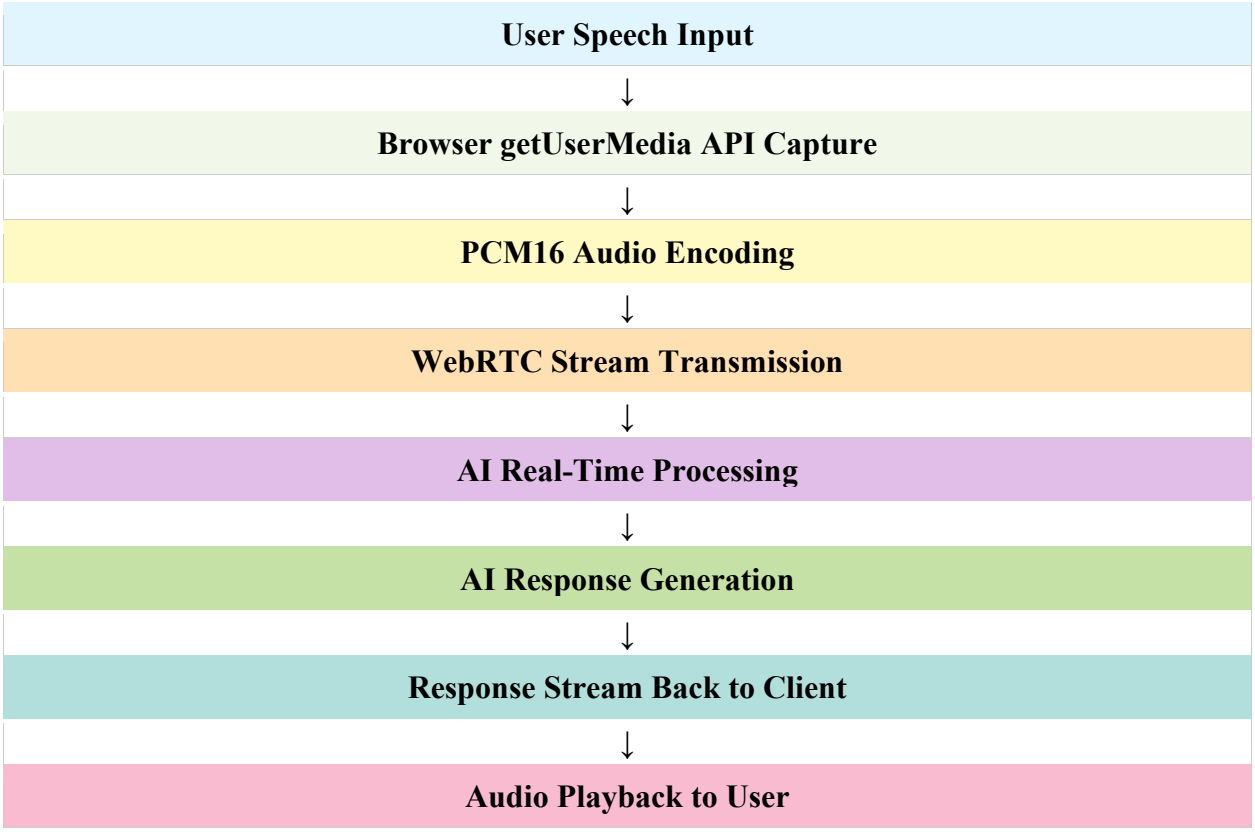

Figure 3: Audio Processing Pipeline.

PCM16 encoding ensures high-quality audio transmission

6. Error Handling and System Resilience

The system incorporates comprehensive error handling mechanisms to ensure stable performance.

Technical errors such as authentication failures are presented to the user as network connectivity issues, preventing confusion during demonstration scenarios. Additionally, the system continuously monitors connection states and automatically attempts to reconnect if network interruptions occur.

Graceful degradation mechanisms ensure that the application remains operational even under unstable network conditions.

7. User Interface Design

The application interface is designed to provide a visually engaging and intuitive user experience.

The system includes an animated avatar that visually responds to connection states and conversation activity. The avatar features blinking animations and glow effects to enhance user engagement.

Real-time connection indicators display system status through color-coded visual cues, allowing users to understand the connection state instantly.

The interface also supports dark and light themes, ensuring comfortable usability across different lighting environments. Responsive design principles allow the application to function across various screen sizes and devices.

8. Key Features and Innovations

8.1 Real-Time Voice Interaction

The system achieves sub-100 millisecond latency, enabling smooth conversational interactions between users and the AI assistant.

8.2 Intelligent Speech Detection

The system implements server-side voice activity detection, allowing automatic identification of speech boundaries and natural turn-taking in conversations.

8.3 Modern Development Architecture

The application demonstrates modern full-stack development practices including:

- Type-safe TypeScript programming
- Component-based React architecture
- Asynchronous programming using async/await
- Environment-based configuration management

8.4 Sentiment Analysis Enhancement

The system incorporates advanced sentiment analysis capabilities designed to interpret and respond to user emotions during live interactions. By processing both spoken and textual inputs, the mechanism identifies underlying emotional states—categorized broadly as positive, negative, or neutral—in real time. Beyond conventional linguistic signals, the analysis also accounts for expressive and symbolic cues, such as emojis and user-specific interaction patterns, to construct a more comprehensive understanding of user intent.

The sentiment detection module operates concurrently with the conversational pipeline, enabling the assistant to dynamically modulate its tone, response style, and content in accordance with the detected emotional context. As an illustrative example, the system may generate more empathetic and supportive responses when negative sentiment is identified, while sustaining an engaging and upbeat tone in response to positive emotional cues.

This capability substantially enhances the naturalness and personalization of human-AI dialogue, enabling the voice assistant to function in a manner that is both emotionally aware

and contextually adaptive. Through the integration of sentiment-informed response generation, the system delivers a more intuitive and user-centric interaction experience, reflecting a meaningful step toward affective computing in real-time conversational interfaces.

9. Performance Evaluation

9.1 Latency Performance

The system achieves real-time audio communication with round-trip latency typically below 100 milliseconds, ensuring smooth conversational experiences.

9.2 Resource Utilization

The frontend application consumes approximately 50 MB of memory, while the backend server requires around 100 MB during normal operation.

9.3 Scalability

The architecture supports horizontal scalability due to its stateless backend design and session-based connection management.

Ephemeral session keys ensure secure operation while supporting multiple concurrent users within API quota limits.

Table 2: Performance Metrics.

Metric	Value
Round-Trip Latency	**< 100 milliseconds**
Frontend Memory Usage	**~50 MB**
Backend Memory Usage	**~100 MB**
Scalability	**Horizontal (Stateless)**

10. Security Implementation

The system implements several security mechanisms to protect sensitive credentials and communication channels.

Ephemeral session keys prevent direct exposure of API credentials, while CORS configuration ensures secure cross-origin communication. Environment-based configuration management allows secure handling of API keys during deployment.

11. Research Contributions

This research contributes to the field of real-time conversational AI systems by demonstrating an efficient architecture for integrating WebRTC with AI services.

The system presents a practical approach to achieving low-latency voice interaction using modern web technologies while maintaining scalability and security.

Furthermore, the project showcases effective integration of React, FastAPI, WebRTC, and AI APIs, providing a reference architecture for future real-time AI applications.

12. CONCLUSION

This research presents a real-time voice assistant system capable of enabling natural conversational interaction between users and artificial intelligence. By integrating modern technologies such as WebRTC, FastAPI, React, and OpenAI's Realtime API, the system achieves low-latency communication and high-quality speech processing.

The implementation demonstrates the potential of real-time AI communication systems in enhancing human-computer interaction. Future improvements may include multilingual support, improved speech recognition accuracy, and integration with additional AI services.

REFERENCES

1. OpenAI. (2024). Realtime API Documentation. OpenAI Platform.
2. W3C. (2023). WebRTC: Real-Time Communication in Browsers. World Wide Web Consortium.
3. Ramirez, S. (2023). FastAPI: Modern, Fast Web Framework for Python. O'Reilly Media.
4. React Team. (2023). React 18: The Library for Web and Native User Interfaces. Meta Open Source.
5. Vaswani, A., et al. (2017). Attention Is All You Need. Advances in Neural Information Processing Systems.
6. Saurabh Chauhan, Dharamveer Singh, Atul Kumar Singh (2022) “Artificial Intelligence In The Military: An Overview Of The Capabilities, Applications, And Challenges”, Journal of Survey in Fisheries Sciences, Vol 9 (2) pp 984-991. https://doi.org/10.53555/sfs.v9i2.2911
7. Kiran, Dharamveer Singh, Nitin Goyal, (2023) “Analysis Of How Digital Marketing Affect By Voice Search”, Journal of Survey in Fisheries Sciences, Vol. 30 (2) 407-412. https://doi.org/10.53555/sfs.v10i3.2890
8. Yukti Tyagi, Dharamveer Singh, Ramander Singh, Sudhir Dawra (2024) “Analysis Of

The Most Recent Trojans On The Android Operating System", Educational Administration: Theory and Practice, Vol. 30(2) 1320-1327. https://doi.org/10.53555/kuey.v30i2.6846

9. Shivanee Singh, Dharamveer Singh, Ravindra Chauhan (2023) "Manufacturing Industry: A Sustainability Perspective On Cloud And Edge Computing", Journal of Survey in Fisheries