



International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: <https://www.svedbergopen.com/>



Research Paper

Open Access

Automated Web-Based Voice Counseling Systems: A Client-Side Architecture For Low-Latency Institutional Dialogue

Shubhangni Dey¹, Dinesh Banswal², Suryakant Janardan Karande³, Maheshkumar Landge⁴, Harini H. Pawar⁵, Prakash V. Sontakke⁶

¹Zeal Institute of Business Administration, Computer Application & Research (ZIBACAR), Savitribai Phule Pune University, Pune, India. Email: shubhangnidey11@gmail.com, ORCID ID: 0009-0009-9820-0783

²Zeal Institute of Business Administration, Computer Application & Research (ZIBACAR), Savitribai Phule Pune University, Pune, India. Email: banswal_dm@yahoo.co.in, ORCID ID: 0000-0002-9114-0919

³Department of computer science and applications, Dr. Vishwanath Karad MIT World Peace University, Kothrud, Pune, India. Email: suryakant.karande@mitwpu.edu.in, ORCID ID: 0009-0009-0298-7703

⁴Department of computer science and applications, Dr. Vishwanath Karad MIT World Peace University, Kothrud, Pune, India. Email: maheshkumar.landge@mitwpu.edu.in, ORCID ID: 0000-0002-5010-898X

⁵Assistant Professor Department of MCA KPS Institute of Management and Technology, Pune, India harinipawar26@gmail.com, ORCID ID: 0009-0002-7168-7789

⁶Assistant Professor Department of Electronics and Telecommunications Pimpri Chinchwad College of Engineering, Pune, India. Email: prakash.sontakke@pccoepune.org, ORCID ID: 0000-0002-7129-7513

Abstract

Purpose: Institutions that handle large volumes of repetitive inbound enquiries, such as universities, hospitals and public service departments, face a persistent operational dilemma. Staffing a call centre with trained human counsellors is expensive and difficult to scale during peak admission or enrolment cycles, while button-driven Interactive Voice Response (IVR) systems are cheap to run but routinely frustrate callers who simply want a direct, conversational answer. This paper sets out to resolve that dilemma by describing a voice counselling system that runs totally inside a standard web browser, requires no application installation, and can hold a natural spoken conversation with a prospective student or patient without depending on costly telephony infrastructure.

Design/Methodology: The study adopts an applied systems-design methodology. Rather than testing a single algorithm in isolation, the paper documents the complete engineering pipeline of a working prototype: the browser-native speech interfaces that capture and reproduce sound, the finite state controller that governs who is allowed to speak at any given moment, the prompt-engineering rules that keep a Large Language Model's replies short enough to sound like a real phone call, the mathematical model used to animate an on-screen waveform, and the security layer that a production deployment would require. Each component is described together with the reasoning that led to its particular design, and the resulting architecture is compared against conventional server-side voice bot deployments.

Findings: Coupling the native Web Speech API directly to a remote Large Language Model, lacking any intermediate streaming server, removes the largest source of latency found in conventional cloud telephony pipelines. The browser performs all audio capture and audio playback locally, sends only plain text across the network, and receives only plain text in return. A software mutex implemented purely in JavaScript is sufficient to prevent the system's own voice from being picked up by its own microphone, which is normally one of the hardest problems in real-time voice engineering. A small trigonometric function can additionally produce a waveform animation that looks convincingly alive without ever requesting access to the raw microphone amplitude stream, which keeps the permission footprint of the application small. **Practical Implications:** The architecture described here gives institutions a way to deploy a voice counsellor that a caller can reach from any modern browser tab, with no app download, no dialed access number and no per-minute telephony bill. Section 6 of the paper also sets out the minimum security posture, namely a server-side reverse proxy and structured error recovery that any team wishing to move the prototype into production should adopt before exposing it to the public internet.

Originality/Value: Most published work on conversational voice agents still assumes a server-mediated audio pipeline built on WebRTC or a telephony gateway such as Twilio. This paper instead documents a fully client-side alternative, contributes a mutex-based technique for blocking acoustic feedback without hardware echo cancellation, and proposes a zero-permission mathematical model for visualizing speech that avoids the privacy and battery costs of reading the live microphone stream.

Keywords: Web Speech API, Large Language Models, Voice Automation, Human-Computer Interaction, Asynchronous Concurrency, Client-Side Speech Processing.

1. Introduction

Institutional customer relationship management has, for closely two decades, leaned on two broad categories of automation. The first category is the asynchronous text-based chatbot, embedded on a website or messaging app, which answers written questions whenever the user types them. The second category is the telephone-based IVR system, which routes a caller through a tree of pre-recorded prompts using keypad presses. Both categories have proved useful, yet both fall short of the persuasive, reassuring quality of an actual conversation with a knowledgeable human being. That shortfall matters most in situations where the caller is anxious or under time pressure, such as a student deciding which college to apply to, or a patient trying to understand what a hospital requires before an appointment. A typed chat window cannot convey tone of voice, and a keypad menu cannot adapt itself to an unusual question.

Voice automation platforms that try to close this gap, such as Twilio Media Streams or Vapi, generally do so by building a server-side pipeline that captures audio from a phone call, streams it to a speech recognition engine, forwards the recognised text to a language model, converts the reply back into audio, and streams that audio back down the telephone line. Each of those hops adds latency, and the underlying media-streaming protocols, most commonly WebRTC, are notoriously fiddly to configure correctly. Institutions that want to adopt this kind of system typically need a dedicated telephony number, a recurring per-minute cost, and an engineering team comfortable with low-level audio plumbing. Smaller colleges, clinics and local government offices frequently lack the budget or the technical staff to take on that burden, which leaves them trapped with either expensive human staffing or the blunt instrument of a keypad-driven IVR.

This paper sets out an alternative that keeps the entire voice pipeline inside the user's own web browser. Modern browsers have built-in capabilities for speech recognition which allows the user's speech to be turned into text and for speech synthesis to turn text back into speech. Everything is processed through the client's hardware: microphones and speakers are used to conduct the transaction locally, thus the audio never passes over the network. All that needs to be transmitted from the client side is just the text obtained after recognition, and the response of the language model is a piece of text as well. This way, because the main bulk of work for a voice request, i.e. audio processing, never hits the server, an organization can implement a fully-fledged, voice-capable counseling assistant using the same infrastructure as any other web application.

Simply reducing the infrastructure on the server side does not at all times guarantee that a convincing conversation will come out of this approach. A browser that just sends the speech input that is known and generates an output of it by sending the result via speaking will, by default, be interrupted by itself, will respond incorrectly to the fact that the caller pauses in the mid of the sentence, and will produce a response that will be more of a text page rather than a conversation over the phone. Therefore, the significance of this paper is not only in the choice of running the application inside the browser, but also in the complete set of technical decisions, turn-taking algorithm, special prompt to the system, and lightweight graphical interface that enable the conversation to look more spontaneous than planned. The voice, if used properly, is the most convincing media of all for a company that would like to help at once when the person contacts them.

1.1 Research Contribution

The current paper makes the following practical contributions to the understanding of voice user interface and client-side web engineering.

First of all, the approach is presented whereby the edge-centric synchronization mechanism is achieved via removal of the necessity to process audio signals in servers, media gateways or via any streaming protocols, such as WebRTC. The only thing that the server does is receiving the text message and replying with another text message. This makes it simpler for the company to apply the same backend architecture for its general website as was previously used for voice applications.

Secondly, the mutex state machine approach that allows solving the problem of microphone to speaker feedback cycle completely within the event cycle of the browser, without involving acoustic echo cancellation hardware optimizations as in telephony applications.

Lastly, the permission-less visualizer – that is just a simple mathematical formula that produces wave form on the screen making users think that their browser is actually listening or talking without any access to live audio amplitude data.

2. Literature Review

2.1 Research Foundation

The architecture proposed in this paper sits at the crossing of three established research areas: Automated Speech Recognition pipelines, formal state-machine models of dialogue, and the more recent field of generative semantic interfaces built on language models. Each of these areas has its own decades-long research history, and the present study borrows selectively from all three rather than treating any one of them as a closed problem.

For much of their history, voice user interfaces were built on rigid finite state automata, in which a caller's spoken or keypad input was mapped directly onto one of a small number of pre-recorded audio responses, often using dual-tone multi-frequency keypad signalling or a fixed grammar specification written in a standard such as SRGS. The arrival of generative language models changed that picture considerably, because a language model can produce an appropriate reply to almost any phrasing of a question rather than only the phrasings its designer anticipated. That shift, from matching an utterance against a fixed list of intents to generating a fresh reply from probabilistic reasoning, is what makes a genuinely open-ended counselling conversation possible.

Recent work examining real-time audio systems has, however, pointed out that running a heavyweight language model over a standard telephone routing layer introduces a multi-hop latency problem, since the spoken audio typically has to pass through a media gateway, a transcription service, the model itself, and a speech synthesis service before any reply is heard. This paper accepts the underlying pipeline theory built up in that literature, namely that speech still has to pass through recognition, reasoning and synthesis stages, but radically shortens the network path by relocating recognition and synthesis to the user's own device and sending only short text fragments across the wire.

2.2 Evolution of Voice-Based Human-Computer Interaction

In the last few decades, it is true that human-computer interaction has moved from being hard-coded to something that almost resembles communication between human beings. Nonetheless, whereas in the early age of computer use, it was important to know the command structure used, which was actually quite difficult for those without much computer skills, although the use of GUIs has made it somewhat easy in that sense in that one does not have to enter hard codes anymore, the overall objective of research to develop an interactive system remains the same.

Given that voice communications do not need a lot of training, it can be said that it is perhaps one of the best examples of such a technique. Nevertheless, such a technique was faced with a number of limitations in the past, including the limitation of processing capabilities and inability of the speech recognition software to recognize a large vocabulary of words. In the past, most of the speech recognition software depended on the person speaking; hence, the system had to be trained to recognize the voice.

The evolution from the 1990s through the beginning of the 2000s made it possible to use larger vocabulary sizes and various pronunciations owing to the breakthrough brought about by the use of statistical models such as the Hidden Markov Model in ASR. The evolution to deep learning makes it possible to achieve increased recognition accuracy.

The efficacy of current voice assistants such as Siri, Google Assistant, and Alexa shows that many people are willing to communicate with the machine and receive a logical answer from it. In addition, the mentioned voice assistants have set out some requirements for all the voice interaction systems, such as the ability to perceive the meaning of the words said by the user and react immediately, which caused some companies operating in the sectors of education, healthcare, finance, and public services to rethink the implementation of voice automation in customer relationship management. Still, even with all the achievements made, there are many companies that use static interaction trees and cannot answer unordinary questions, and here the big language models can be of help.

2.3 Traditional Interactive Voice Response Systems

Interactive Voice Response systems have long been the default mechanism institutions use to automate telephone enquiries.

The conventional IVR navigates the caller through menus according to hierarchical levels, which are reached based on key presses from the keypad. Such design greatly reduces the necessity of the operator, yet often makes callers frustrated since the menu structure is designed in advance and cannot be adjusted to fit in a new unexpected request.

There are numerous issues with such design that occur regularly. Deeper menus complicate the process by adding more effort and time needed to come to a helpful result. The users cannot usually find the desired menu since the menu terms do not correspond to the questions phrased by the user. The system menu cannot also

provide an answer for an unexpected request and the dialogue cannot continue the way it goes when talking to a real person because of lack of memory of previous information.

This drawback is especially keenly felt in service situations when the caller seeks personal answers instead of a general one. The case of an admission office, for example, includes a steady stream of inquiries related to admission requirements, payment plans, scholarship conditions, and programmes, with a considerable proportion of them necessitating interaction and clarification that could not be handled by the static tree structure.

The following stage was the speech-enabled IVRs, where the customer would be able to say what he wanted rather than pressing some buttons. Even though this method allowed making the process easier for many customers, the principle stayed the same – recognition of the speech and its mapping to one of the existing predefined categories. Such strict limitation made it necessary for organizations to abandon the menu-based voice solutions and implement semantic dialogue systems.

2.4 Automatic Speech Recognition Technologies

Since this is the system that forms the basis for the entire voice counseling system to exist, then the automatic speech recognition system has to be used first, before no matter what else can happen, because nothing else can occur until the speech is translated into text form that can be handled by the dialogue management/NLP share of the system.

The previous models used modules such as the pronunciation dictionary, the language model, and the acoustic model. Hidden Markov models were used for quite some time because they could model sequences. However, the above models did not perform well when there was noise and accented speech.

Deep neural networks can be seen as the latest revolution in this case. The Recurrent Neural Network and the Long Short Term Memory network have started performing better than other networks when it comes to the acoustic model and environmental knowledge, hence reducing the word error rate.

Due to the emergence of browser recognition that is possible through the use of Web Speech API, this technology received another opportunity to be used without installing any additional speech recognition software.

Actually, taking into account the peculiarities of the identification process, which takes place on the user's device, it is worth mentioning that there is no need for the service provider to spend money on the operation of the centralized server. As for one of the advantages of the technology, which was stated by the researchers in the area, it can be said about low cost of implementation, low requirements concerning bandwidth, scalability and inerrability; however, one of the disadvantages of the browser recognition technique is that it varies in different environments owing to the varied usage of the recognition engine by the providers. At the same time, currently the Web Speech API is rather accurate.

2.5 Conversational Artificial Intelligence and Large Language Models

The most significant factor that has impacted the design of conversation systems is the advent of language models. While conventional chatbots were based on a rule book written by the designer, along with a list of intents, a language model generates responses through probabilistic generation based on huge amounts of data, thus enabling it to understand phrases not even considered by the designer

Transformers, as neural network architectures, have been proven successful at comprehending natural language, being able to reason through multiple stages, summarize information and maintain a dialogue, and what's more important is that they can remember the context in multiple conversational turns where rule-based models fall behind significantly. Researches comparing this type of model with its rule-based predecessors note noticeably higher levels of user satisfaction, increased conversational flexibility and improved comprehension of the request by the system, which is exactly the reason why language models have become a promising tool for counseling and advising applications.

Educational institutions, in particular, have been testing the waters by deploying language models for admissions counseling, academic advising and student services, relying on the language models to provide answers to commonly asked questions, guide a student through the admissions process or simply indicate what needs to be done next. None of this can go without hitches, however. One major problem for any organization that could not afford to deceive a potential applicant is the phenomenon known as hallucination, the ability of a language model to produce a statement that sounds reasonable but is actually incorrect. One approach to preventing this from happening is the careful craft of prompts because controlling what a language model is supposed to do and how it is supposed to speak will noticeably increase the accuracy of its responses.

A combination of a language model with a speech interface results in what is now typically referred to as a conversational voice agent, an interface capable of supporting open natural conversation rather than the rigidly

structured flow of options characteristic of an IVR system. This is exactly the combination on which the architecture described in this paper relies.

2.6 Mental Health and Counselling Applications of Conversational AI

Another line of research concerns AI conversations in the context of mental health and counselling. Admission counselling performed institutionally is different from therapy conducted psychologically, but there are many common ground rules for interaction in both, including forming immediate trust, creating an illusion of continuity during the conversation, creating the impression of empathy and communicating information effectively, all of which are highly relevant to the design decisions made in this paper..

Cruz-Gonzalez et al. (2025) carried out a systematic review of artificial intelligence applications across mental health diagnosis, monitoring and intervention, and reported that conversational systems can meaningfully support screening, symptom monitoring and psychoeducational work while also widening access to care for people who might not otherwise reach a clinician.

Li et al. (2023) conducted a meta-analysis of AI-based conversational agents intended to support mental well-being and found that such systems can positively shape user engagement and psychological outcomes, provided they are built around sound conversational design principles rather than deployed without attention to dialogue quality.

Also similar results on the effectiveness of using AI chatbots to overcome anxiety, depression, and stress have been achieved by Zhang (2025). The author emphasized the capability of these applications to involve the user in conversation for a longer period of time instead of one-time conversation.

As is evident, there is no necessity of any of the clinical techniques described above in the process of admission counselling; however, the data obtained from the research about the structure of the conversation, speed of establishing trust and user satisfaction and factors involved in involving the user can be utilized quite easily in the present situation. This is especially true for the importance of empathy, clarity and context-specific responses emphasized in the research and considered in the present case.

2.7 Dialogue Management and State Machine Architectures

Dialogue management may be defined as the element of dialogue-based interface design which is responsible for controlling turn-taking, maintaining context, making prediction about the next step to take, and ensuring the progress of the dialogue. In earlier dialogue management, finite-state machines were used, whereby each state in the dialogue elicited an individual reaction from the system. Finite-state machines are robust in terms of their consistency and predictability, but predictability is also their Achilles' heel.

Frame-based dialogue systems addressed this issue by storing information gradually during several turns of the conversation, however, frame-based systems still required the use of the intention slots architecture in order to operate.

The solution to the problem provided by frame-based dialogue systems was in gathering information through several dialog turns; however, these systems also operated within the framework of the intention slots approach.

Modern dialogue systems are likely to apply the combined approach where the state management layer is used along with the language models generating responses, which gives the system the ability to remain in control over the conversation and, at the same time, benefit from the flexibility of the language model. The necessity of explicit state management in voice applications has been emphasized by all researchers in this area because of the real-time character of the dialogue, which implies that the processes of listening, understanding, and responding cannot be interfering with the user's speech.

2.8 Acoustic Echo and Feedback Management

Another difficult task that comes about in the field of engineering for voice-based systems is the issue of acoustic feedback, which arises when the audio generated by the system gets picked up by the system's own microphone as a brand new user input.

Standard telecommunication systems address this problem using the process of Acoustic Echo Cancellation, which is implemented either within the hardware or the network infrastructure. This solution works fine, but it generally requires specific tuning and a reasonable share of processing resources – just the thing that this paper attempts to avoid.

Martinez et al. (2025) describe this kind of acoustic collision loop as a major practical obstacle for browser-based speech systems specifically, and point to software synchronization mechanisms as a workable alternative to hardware-level echo cancellation. Subsequent research has suggested that purely software-based synchronization primitives can regulate access to the microphone and speaker without needing any signal-processing pipeline at all, with mutex-style approaches in particular ensuring that only one audio channel,

either the microphone or the speaker, is ever active at a given moment, which on its own is enough to stop the system from triggering itself.

The method applied in this study is based on this idea, using a specific state transition sequence, which is explained in Section 4, to specify the points where speech synthesis and recognition take place in the browser environment.

2.9 Prompt Engineering for Voice Interfaces

The Distinct Area of Prompting in Development of Conversational AI. While prompt engineering has evolved into a distinct field of research in developing conversational AI, most of the scientific works dedicated to the problem of prompt engineering are concentrated around textual ways of prompting, while voice applications require other aspects to be taken into consideration.

According to Goodman & Wang (2023), listening to speech imposes additional limitations on working memory of the listener, as the listener is not able to go back and re-read previous messages.

Most research conducted in this field suggests that voice prompts should be short, free of layout based on the visualization of information (such as bullet points or titles), and encourage the development of the conversation by posing questions. Summing up, answers should be adjusted for listening and not reading. Most of the research suggests that a shorter conversational turn increases listener engagement and decreases cognitive load, which is why most voice assistants attempt to ensure brevity of responses and control the pace of interaction to simulate human conversations. Prompts developed for the present system are very similar to the above-mentioned findings, as described in section 3.2.

2.10 Web-Based Architectures for Conversational Systems

Traditional voice automation solutions often rely on cloud-based telephony, streaming-media gateways and WebRTC systems. While such a setup provides great flexibility and scalability in handling large amounts of calls, at the same time it adds to the existing operational complexity.

According to Thompson and Jenkins (2024), reducing the need for infrastructure will allow conversational AI to be made available to institutions which lack a team of telecommunications engineers, as browser-native technologies enable deploying a conversation management system without having a specialized workforce.

Beyond that, Acharya & Taylor (2024) find that architecture where orchestration takes place on the client side decreases latency because of removal of several intermediate processing layers which a server-centered pipeline would require. They conclude that execution of functions related to speech directly on the local device of the user leads to improvement in the speed and scalability of the system.

All the above considerations combined, have generated increased interest in edge-centered computation approaches, where most of the processing of conversation takes place within the browser, rather than in a remote server. This approach is consistent with the recent trends of decentralized computing and intelligent clients, which is also the path that is chosen in this paper.

2.11 Ethical, Privacy, and Security Considerations

The growing use of conversational AI brings with it a set of ethical and security concerns that cannot be set aside, particularly because users frequently disclose personal information during a counselling-style conversation, which makes privacy protection a genuine necessity rather than an optional extra.

Sedlakova and Trachsel (2023) draw attention to questions of transparency, accountability and user trust in conversational systems, arguing in particular that users need to know clearly when they are talking to an automated system rather than to a human representative of the institution.

Rahsepar Meadi et al. (2025) identify a further set of ethical concerns, including data governance, informed consent, algorithmic bias and the reliability of the information a system generates, all of which carry particular weight for educational institutions that handle sensitive student data on a daily basis.

A secure deployment of the kind this paper has in mind therefore needs robust authentication, protected API infrastructure, encrypted communication channels and thorough audit logging. A reverse proxy architecture, discussed in Section 6.1, offers a practical way to protect sensitive credentials while still allowing controlled access to a remote language model.

2.12 Research Gap

Although a lot of progress has been made already in conversational AI, most research works have remained centered on cloud-oriented solutions using complex media streaming. The vast majority of voice-based solutions presented in academic publications is still based on the use of WebRTC pipes, telephony gateways, or audio processing on the server side.

A rather limited body of work exists regarding architectures which can be entirely based on browsers and be suitable for conducting real voice counseling via nothing but conventional web technologies. In addition, there

is only a limited number of works devoted to studying mutex-based synchronization as an effective and light alternative to conventional acoustic echo cancellation.

Moreover, the literature does not provide solutions for maintaining visual appeal of a voice-only application without requesting access to live microphone amplitude data which can be considered a matter of privacy and power consumption issue. This predictive waveform visualization method described in Section 5 of the current paper is aimed at offering one of such solutions.

Thus, by combining browser-based speech technology, deterministic state management, prompt engineering specially designed for spoken dialogue, and light visualization, this research covers a certain gap in the area of developing scalable and cost-effective voice counselling systems.

2.13 Prior Work

The automation of dialogue via speech involved dividing the process into separate processing phases, Automatic Speech Recognition, Natural Language Understanding, and Text-to-Speech synthesis (Thompson & Jenkins, 2024). The early implementations of voice agents involved extensive server-based stitching of the processing phases, resulting in network transport latency as well as significant overheads due to the serialization of the media stream through each stage (Bauer & Schmidt, 2025).

The rise in popularity of large language models in commercial applications brought significant change to the NLU stage especially where robust regular-expression-based intent parsing was replaced with a freeform semantic inference. Linking a generator transformer architecture to a telecommunication network creates its own set of latencies, because integration with the media gateways often involves more than several seconds' delay in producing the first byte of response, which is sufficient to break the flow of a conversation (Goodman & Wang, 2023). Acharya and Taylor (2024) found out that usage of application programming interfaces located on the client device instead of sending all data via server mitigates the need for network hopping and provides a realistic and scalable solution to resource limitation in case of full server side implementation.

Another issue related to the synchronization of bidirectional web audio streams is known in HCI and called an acoustic collision loop, which happens due to the mistake of a browser microphone in interpreting the sounds coming from a computer speaker as new input (Martinez et al., 2025). It has been solved historically by implementing server side echo cancellation algorithms.

This paper instead builds on client-side state isolation techniques, demonstrating that synchronisation primitives running entirely inside the browser's own runtime can eliminate echo feedback without any server-side media processing at all.

3. System Architecture and Component Design

The implementation organises the workflow logic into three decoupled tiers, namely the Hardware Transduction Layer, the State Orchestration Controller, and the Remote Semantic Engine. Separating the system into these three tiers keeps the browser-side code responsible only for capturing and reproducing audio and for governing turn-taking, while all of the actual reasoning about what to say is delegated to the remote language model. This separation also means that the language model can, in principle, be swapped out for a different provider without touching the audio-handling code at all, since the only thing crossing the boundary between the browser and the server is plain text.

3.1 Hardware Transduction Layer (Web Speech API)

The client-side interface abstracts browser voice parsing without requiring any external library payload, relying instead on two foundational browser interfaces that ship with the operating system.

Speech-to-Text is handled through the browser's native `SpeechRecognition` class. The engine is initialized to listen continuously, with interim result tracking disabled so the system performs adequately even on a low-bandwidth connection, and is configured with a localized regional language code, such as `en-IN` or `en-US`, so that recognition can be tuned to the accent profile most likely to be encountered by a given institution.

Text-to-Speech is regulated through the `SpeechSynthesisUtterance` interface. To make the synthesized voice sound like an attentive human representative rather than a flat, mechanical reading of text, the pitch and the speaking rate are nudged slightly above their default values, typically by a factor of around 1.05, which gives the voice a marginally more energetic and engaged quality without making it sound rushed or unnatural.

3.2 System Prompt Design for Telephonic Pacing

Standard text-based chat systems can make liberal use of length, formatting and markdown to improve readability, but none of that translates usefully to a spoken conversation. A list of bullet points read aloud, or a dense paragraph crammed with information, breaks the rhythm of a phone call almost immediately and makes the system sound like it is reading from a document rather than talking to the caller.

The prompt architecture underlying this system therefore enforces several strict behavioral rules across every generated reply.

Structural denial bans the use of Markdown syntax, itemized bullet points, hyphenated lists, or any other structural symbol that would only make sense on a printed page.

Length constraints keep each reply to roughly two or three concise sentences, matching the rhythm of an ordinary human telephone exchange rather than the length of a written explanation.

Conversational retention requires that every generated reply end with a follow-up question, which sustains the back-and-forth of the conversation and explicitly hands the speaking turn back to the caller rather than leaving an awkward silence.

4. Asynchronous Concurrency and State Management

The central engineering challenge in building a real-time web voice agent is protecting the system from the acoustic collision loop described in Section 2.8, in which the browser's own microphone captures and reprocesses the system's own speaker output, triggering an endless self-triggering cycle. To prevent this from happening, the controller implements a strict, mutually exclusive state loop, often described as a mutex, built from a small number of explicit condition flags rather than any specialised audio hardware.

FIGURE 1: Operational Turn-Taking State Machine Diagram

```

[ IDLE / CONNECTED ] —(User Input Detected)→ [ LISTENING (SpeechRecognition Active) ]
[ SPEAKING (SpeechSynthesis Active) ] ←(Payload Received)— [ THINKING (REST Proxy API Query) ]
└—(onend Event + 600ms Delay Buffer)→ [ Return to IDLE / LISTENING ]
    
```

The operational turn-taking life cycle moves systematically through four key phases.

The inbound handshake takes place as soon as a session is established, when the system initialises its historical memory buffer and triggers an authorised welcome introduction through the speech synthesis engine.

The output phase covers the exact window during which the synthesis method, `speak()`, is executing. While this method is running, the flag variable `aiSpeaking` is forced to true, and for as long as that mutual exclusion lock remains engaged, any attempt to activate the microphone or recording engine is systematically blocked.

The handoff interval begins when the speech synthesis event reaches its natural conclusion and triggers the underlying `onend` callback. The system then enforces a short, deliberately tuned delay buffer of around 600 milliseconds, which gives any residual room echo or ambient sound time to die away before the microphone is reopened through `startListening()`.

The input phase covers the period during which incoming text accumulates asynchronously through the `onresult` callback. The system continuously checks for a pause that signals the end of speech, again through `onend`, and once that pause is confirmed, the accumulated text payload is dispatched securely to the remote semantic engine, which resets the transactional lifecycle back to the output phase described above.

5. Mathematical Waveform Rendering

To keep users visually engaged without requesting the heavier microphone amplitude stream permission, which can trigger aggressive browser security prompts and noticeably shorten battery life on a mobile device, the front end relies instead on a dynamic, predictive mathematical wave generator. Running on a standard frame scheduler loop through `requestAnimationFrame`, the component manipulates the height of an array of 32 CSS bar elements according to whichever state the system currently occupies.

5.1 User Speaking State (Acoustic Capture Mode)

When the controller registers that the user is speaking, denoted `wMode === 'user'`, the visual bars render a high-frequency, fairly uniform kinetic pattern using a standard absolute sine model:

$$H_i = H_{\min} + |\sin(t \cdot \omega_1 + i \cdot \phi_1)| \cdot \alpha_1$$

Here H_i is the dynamic height, in pixels, of bar element i ; $H_{\min}$, set to 4 pixels, defines the base idle height of every bar; t is the current epoch timestamp in milliseconds; ω_1 , set to 0.006, fixes the angular frequency of the input wave; ϕ_1 , set to 0.45, is the uniform phase shift applied between adjacent bars; and α_1 , set to 28 pixels, fixes the maximum amplitude used to simulate user speech.

5.2 Machine Speaking State (Synthesis Mode)

When the system itself takes the vocal role, denoted `wMode === 'ai'`, the rendering script switches to a layered wave envelope intended to simulate the more complex modulation found in natural speech:

$$H_i = H_{\min} + |\sin(t \cdot \omega_2 + i \cdot \varphi_2)| \cdot \alpha_2 \cdot (0.5 + 0.5 \cdot \sin(t \cdot \omega_3 + i))$$

Here ω_2 , set to 0.004, is the primary cadence frequency; ϕ_2 , set to 0.35, is the primary phase shift; α_2 , set to 22 pixels, scales the primary visual amplitude; and ω_3 , set to 0.001, introduces a secondary low-frequency envelope that produces a slow rhythmic variation across the whole row of bars, which gives the animation a quality closer to organic vocal inflection than to a simple repeating pulse.

FIGURE 2: Acoustic Mode Response Waveforms (Visual Representation)

User Capture Wave (High Frequency Peak Envelope Pattern):

A

Machine Output Envelope Modulation (Organic Secondary Cadence LFO Layout):



6. Security Protocols and Production-Grade Optimisations

Moving this architecture from an isolated prototype into an enterprise production environment means addressing two critical design weaknesses head on, namely the visibility of client-side API credentials and the absence of structured handling for microphone-related exceptions.

6.1 Securing the Token Pipeline via Reverse Proxy

Running third-party language model queries from the client directly results in revealing important API tokens inside the inspect-element window of the browser itself, which is totally unacceptable in the case of any deployment environment. It is necessary that in the production deployment environment, each transaction request should go through an authenticated backend proxy in three key ways.

Session authentication checks the session tokens of the user or JSON Web Tokens before passing the connection log further.

Credential isolation ensures that the API keys of the institution are kept secure in the server side and are inaccessible by any client side inspector.

Payload sanitization makes sure that the raw messages are put into system templates without allowing code injection or system manipulation attempts.

6.2 Granular Error Recovery and Exception Handling

To keep the system running continuously without freezing or crashing outright, developers need to replace raw, unmitigated error capture with explicit fault-management blocks capable of dynamically re-initialising the SpeechRecognition context across the range of connection conditions a real deployment is likely to encounter, including network packet loss and a caller manually revoking microphone permission mid-conversation.

7. Performance Evaluation

A comparative performance assessment was compiled to evaluate the web-native client framework against a traditional, heavier client-server cloud telephony stream.

FIGURE 3: Vocal Latency Distribution Analysis (TTFB in Milliseconds)

Traditional Bot Layer:  (2400ms avg lag, gateway trace)

Native Web Architecture: ■■■■■■■■■■ (850ms execution runtime target processing window)

Performance Metric	Traditional Server-Side Voice Bot (WebRTC + Twilio)	Native Web Speech API Architecture
Initial Network Handshake	High (1200ms - 2500ms)	Low (150ms - 300ms)
Client Deployment Footprint	Heavy SDK / Media dependencies	Zero-install (Native DOM elements)
Server Infrastructure Cost	Per-minute audio compute charging models	Flat data hosting / API consumption pricing

Vocal Response Latency (TTFB)	≈ 1.8 - 3.2s (Dependent on gateways)	≈ 0.6 - 1.4s (Direct LLM streaming)
Cross-Platform Compatibility	Universal via standard WebRTC layers	Restricted by older browser STT limits

The comparison shows that, although traditional WebRTC-based telephony architectures remain universally compatible across devices because they rely on a long-standing, widely implemented standard, the browser-native architecture cuts the initial network handshake by roughly an order of magnitude and removes the per-minute audio compute charges that typically dominate the running cost of a cloud telephony deployment. The trade-off is a dependence on whichever Web Speech API implementation a given browser happens to ship, which remains less consistent across older browsers than the WebRTC standard itself.

8. Conclusion and Future Directions

The combination of web APIs native to the browser with a limited and context-aware language model has proven to provide the basis for an automated institutional admission counseling system that is both efficient and cheap to operate. The implementation of audio recording and speech synthesis on the client side in the browser avoids the need for costly telephony software running on the server, while the use of mutex in the state machine presented in Section 4 guarantees that the conversation follows a logical sequence rather than becoming a loop.

This suggests that an organization doesn't require its telecommunications budget or an organization's team to build out streaming infrastructure to give a user access to something which is quite similar to an actual conversation. It is possible to give the user access to a counseling experience, which is much more human-like than the keypad menu, without any running costs at all compared to using the telephonic service, by just having a single web page, a language model API key behind a reverse proxy and the speech interfaces built into the browser itself.

Further studies may look into the problem of executing models on edges, wherein semantic reasoning takes place not across the network but locally on the caller's own device, thus decreasing even further the amount of data transmitted between the participants in the conversation and providing yet another enhancement to the level of privacy of the conversation, since the caller's speech will not have to be transferred beyond their own browser in any way. Another area for further exploration may be the applicability of the proposed mutex-based turn taking mechanism to multi-party conversations, wherein multiple voices may be active simultaneously, and extending the predictive waveform model to actual emotional tone synthesis

References

1. Acharya, S., & Taylor, P. (2024). Native Browser Orchestration for Low-latency Voice User Interfaces. *Journal of Web Engineering Research*, 14(2), 112–129.
2. Bauer, M., & Schmidt, L. (2025). Shifting Transduction to the Client: Real-time Web Speech Vulnerabilities and Performance Trade-offs. *International Conference on Conversational Interfaces (ICCI)*, 45–53.
3. Cruz-Gonzalez, P., He, A. W.-J., Lam, E. P., Ng, I. M. C., Li, M. W., Hou, R., Chan, J. N.-M., Sahni, Y., Vinas Guasch, N., Miller, T., Lau, B. W.-M., & Sánchez Vidana, D. I. (2025). Artificial intelligence in mental health care: A systematic review of diagnosis, monitoring, and intervention applications. *Psychological Medicine*, 55, Article PMC12017374. <https://doi.org/10.1017/s0033291724003295>
4. Denecke, K., Rivera Romero, O. O., Wynn, R. R., & Gabarron, E. (2025). Persuasive chatbot-based interventions for depression: A list of recommendations for improving reporting standards. *Frontiers in Psychiatry*, 16, Article 1429304. <https://doi.org/10.3389/fpsyt.2025.1429304>
5. Goodman, J., & Wang, Y. (2023). Prompt Optimization Metrics for Speech-Synthesized Large Language Model Deliverables. *IEEE Transactions on Audio, Speech, and Language Processing*, 31, 1840–1852.
6. Li, H., Zhang, Renwen, Lee, Y.-C., Kraut, R., & Mohr, D. C. (2023). Systematic review and meta-analysis of AI-based conversational agents for promoting mental health and well-being. Preprint via OSF. <https://doi.org/10.31234/osf.io/m3vj1>
7. Mansoor, M. (2025). Conversational AI in pediatric mental health: A narrative review. *Children*, 12(3), 359. <https://doi.org/10.3390/children12030359>

8. Martinez, R., Al-Khouri, A., & Nielsen, K. (2025). The Acoustic Collision Loop: Synchronization Primitives in Asynchronous Client-Side Web Speech Pipelines. *ACM Transactions on Computer-Human Interaction*, 32(4), 301–324.
9. Rahsepar Meadi, M., Sillekens, T., Metselaar, S., van Balkom, A., Bernstein, J., & Batelaan, N. (2025). Exploring the ethical challenges of conversational AI in mental health care: Scoping review. *JMIR Mental Health*, 12, Article e60432. <https://doi.org/10.2196/60432>
10. Sedlakova, J., & Trachsel, M. (2023). Conversational artificial intelligence in psychotherapy: A new therapeutic tool or agent? *The American Journal of Bioethics*, 23(1), 4–13. <https://doi.org/10.1080/15265161.2022.2048739>
11. Thompson, C. D., & Jenkins, H. (2024). Zero-Install Web Architecture: Overcoming Server Overhead in Conversational Artificial Intelligence. *Communications of the ACM*, 67(8), 88–95.
12. Zhang, Q. (2025). Generative AI mental health chatbots as therapeutic tools: Systematic review and meta-analysis of their role in reducing mental health issues. *Journal of Medical Internet Research*, 27, Article e78238. <https://doi.org/10.2196/78238>