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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15570

CallMind: An AI-Powered Real-Time Voice Agent Platform Using a Pluggable Speech-to-Text, Large Language Model, and Text-to-Speech Pipeline

Sahil Makandar[1], Rohini Magamdum[2], Aditya Mali[3], Sudesh Kumbhar[4], Radheshyam Sah[5], Rajendra Hiremath[6]

Department of Computer Science & Engineering,

D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, India[1-5]

Guide, Department of Computer Science & Engineering,

D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, India[6]

Abstract: Intelligent voice agents offer a scalable alternative to traditional Interactive Voice Response (IVR) systems and human call centres, yet their deployment remains technically complex. This paper presents CallMind, an AI-powered voice agent platform that enables businesses and individuals to deploy conversational telephone agents within minutes. The platform implements a pluggable, channel-agnostic pipeline that normalises all input modalities—phone calls, browser audio, and direct text—into a unified *QueryPayload* abstraction before processing. The core intelligence pipeline routes each query through Azure Cognitive Speech Services for real-time streaming Speech-to-Text (STT), Groq API (LLaMA 3.3 70B) for large language model inference, and Azure Neural Text-to-Speech (TTS) for audio synthesis. A sentence-by-sentence streaming architecture achieves end-to-end response latency of approximately 780 ms, comparable to natural human conversational response time. The system is built on a dual-backend architecture: a managed Supabase layer for multi-tenant agent configuration, knowledge base management, and conversation persistence; and a containerised Python FastAPI server on Microsoft Azure for the real-time AI pipeline. Live validation through Twilio Programmable Voice demonstrates natural conversation quality with accurate transcription, contextually relevant responses, and seamless audio delivery. The architecture provides a clear migration path from full-context LLM prompting to Retrieval-Augmented Generation (RAG) and from Docker Compose to Kubernetes without modifying the application layer.

Keywords: Voice Agent, Speech-to-Text, Text-to-Speech, Large Language Model, Real-Time Pipeline, WebSocket, Twilio, FastAPI, Docker, Pluggable Architecture, Conversational AI, Redis, Supabase, Retrieval-Augmented Generation

I. INTRODUCTION

CUSTOMER interaction systems are a fundamental operational requirement for businesses across sectors. The two dominant approaches—traditional Interactive Voice Response (IVR) and human-staffed call centres—both suffer from fundamental lim- itations. IVR systems operate on rigid, pre-programmed menu trees and cannot interpret natural language; callers are forced to navigate frustrating numbered menus, and the system fails entirely when a query falls outside predefined categories [1]. Human call centres, while capable of nuanced conversation, are expensive, unscalable during demand spikes, unavailable around the clock, and introduce inconsistency depending on individual agent knowledge and mood.

The convergence of three mature technologies—accurate real-time automatic speech recognition (ASR), capable large language models (LLMs), and high-quality neural text-to- speech synthesis—now enables intelligent voice agents that resolve both limitations simultaneously. However, combin- ing these components into a production-grade, multi-tenant platform requires solving significant engineering challenges: concurrent telephone session management, stateless webhook state restoration, real-time streaming for low latency, and scal- able knowledge base management across multiple business tenants.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026
DOI: 10.17148/IJARCCE.2026.15570

This paper presents CallMind, a complete AI-powered voice agent platform addressing these challenges. The key technical contributions are:

1. A *QueryPayload* abstraction that decouples input modal- ity (phone call, browser microphone, text API) from the intelligence processing layer, enabling complete channel independence.
2. A sentence-level LLM-to-TTS streaming pipeline that achieves end-to-end latency of approximately 780 ms from speech completion to first audio playback—comparable to natural human response time.
3. A Redis-backed multi-session architecture supporting con- current telephone calls on commodity cloud hardware (2 vCPU, 4 GB RAM).
4. A returning-caller recognition system using knowledge base snapshot overlap for contextually accurate conversation history injection.
5. A complete CI/CD pipeline deploying via Docker Com- pose with semantic versioning and manual production approval gates.

II. RELATED WORK

A. Conversational AI and Voice Agents

Early conversational systems relied on finite-state machines and hand-crafted dialogue trees [1]. The introduction of deep learning-based intent classification gave rise to task-oriented dialogue systems. Transformer-based language models, particularly BERT [2], enabled open-domain dialogue without explicit intent labelling. The emergence of large language models such as GPT-4 [3] shifted the paradigm toward prompt- engineered agents capable of handling arbitrary queries given sufficient context. Contemporary voice assistant platforms such as Amazon Alexa and Google Dialogflow CX incorporate LLM-based response generation but remain predominantly closed systems with limited customisability.

B. Speech Recognition and Synthesis

Automatic Speech Recognition has advanced substantially with sequence-to-sequence architectures. OpenAI Whisper [4] demonstrated that a single multilingual model can achieve near-human accuracy. Microsoft Azure Cognitive Speech Ser- vices builds on similar principles, offering real-time streaming recognition suitable for telephony integration. For speech synthesis, neural Text-to-Speech systems such as WaveNet [5] produce highly natural output. Azure Neural TTS achieves mean opinion scores comparable to human speech and can out- put audio in raw 8 kHz mu-law format—the native encoding used by the public switched telephone network—eliminating a conversion step.

C. Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) [6] addresses the limitation of fixed-context LLM prompting by dynamically retrieving relevant document chunks at inference time. RAG architectures consistently outperform full-context prompting for large or frequently-updated knowledge bases in terms of factual accuracy and response relevance. CallMind delib- erately adopts full-context prompting as a proof-of-concept baseline, with the architecture designed for a drop-in swap to pgvector or RAGFlow in a subsequent phase.

D. Telephony and WebSocket Streaming

Twilio's Programmable Voice API provides webhook-based call handling with the Media Streams feature extending this capability to raw audio streaming over WebSocket [7]. A key technical challenge is that Twilio webhooks are stateless HTTP callbacks, requiring external session storage to maintain conversation context across multiple turns within a single call. Prior work in real-time voice systems has used in-process state [1] but this limits concurrency. CallMind uses Redis as a low-latency cache for agent configuration and caller history, with in-process session state for the active call pipeline.

III. SYSTEM ARCHITECTURE

CallMind follows a dual-backend architecture that cleanly separates stateful business concerns from the real-time AI pipeline.

A. Dual-Backend Design

Backend 1—Supabase (Common Backend): Handles au- thentication, multi-tenant database management, file storage, and real-time dashboard updates. The management dashboard communicates with Supabase directly via the supabase-js client. Row-Level Security (RLS) policies enforce per-tenant data isolation at the database layer.

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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15570

Backend 2—FastAPI (AI Pipeline): Runs in Docker Com- pose on a Microsoft Azure virtual machine. Handles the real-time voice pipeline, Twilio WebSocket sessions, text quer processing, and background task coordination via Celery.

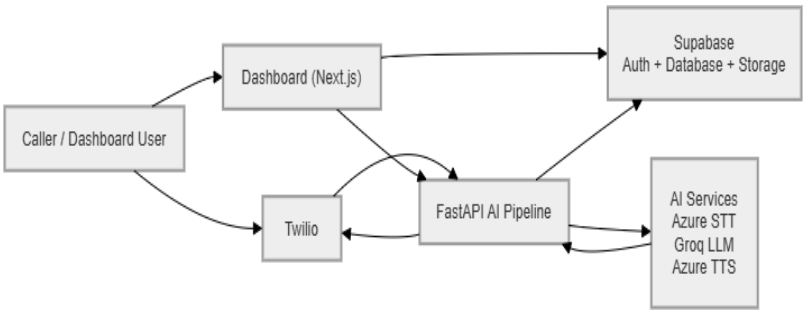

Figure 1. CallMind dual-backend system architecture.

B. QueryPayload Abstraction

The *QueryPayload* is the central contract of the intelligence layer. Every input channel—Twilio phone call, REST API, browser WebSocket—produces a *QueryPayload* before reach- ing the LLM, ensuring the inference service is completely unaware of the input source:

- sessionid: Unique per call (Twilio CallSid or UUID)
- text: Transcribed or raw input text
- inputmode: voicetwilio| textapi
- agentconfig: Agent configuration from Supabase
- kbdocuments: Knowledge base document content
- callerhistory: Past conversation summaries for this caller
- conversationhistory: All turns in the current session

C. Voice Call Pipeline

The real-time voice pipeline handles the complete lifecycle of an inbound telephone call. When Twilio opens a WebSocket connection for an inbound call:

1. The start event initialises a SessionState object, loads agent configuration and knowledge base documents from Redis cache (falling back to Supabase on cache miss), identifies or creates a caller record, loads caller history, and starts the Azure STT push stream.
2. Each media event delivers a base64-encoded chunk of mu-law audio, decoded to PCM16 at 8 kHz and pushed to the Azure STT stream.
3. When Azure STT fires a final transcript event, the pipeline constructs a *QueryPayload* and passes it to the Groq LLM streaming API.
4. Each sentence yielded by the LLM is synthesised by Azure Neural TTS and sent back to Twilio as a media event, with a markevent for playback tracking.
5. On stop, a Celery background task persists the full con- versation transcript and generates an automatic summary.

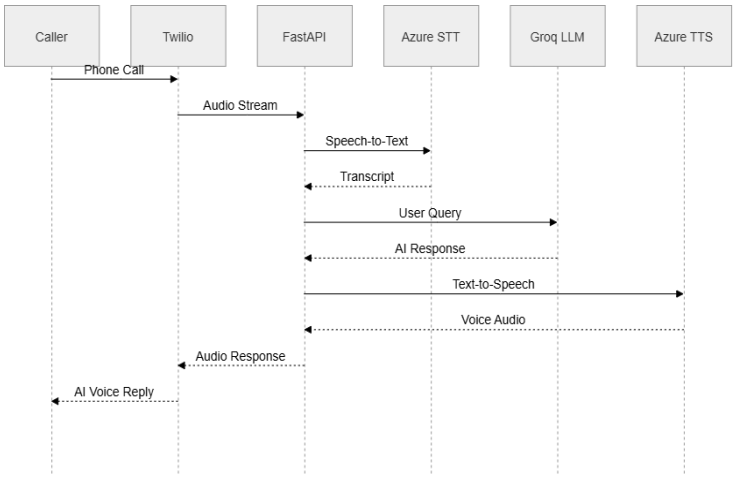

Figure 2. End-to-end voice call pipeline.

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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15570

IV. METHODOLOGY

A. Sentence-Level Streaming Architecture

The primary latency challenge in voice AI systems is the time between speech completion and the start of audio playback. A naive implementation that waits for the full LLM response before beginning TTS synthesis introduces 1–3 seconds of perceived silence. CallMind resolves this through sentence-level streaming: the LLM output is monitored for sentence boundary characters (period, question mark, exclamation mark, colon, semicolon), and each complete sentence is immediately synthesised and transmitted while the LLM continues generating the remainder of the response.

The sentence boundary detection algorithm processes the LLM token stream character-by-character, accumulating a buffer. When a boundary character is detected and the buffer exceeds a minimum length threshold (to avoid synthesising short fragments), the buffer is dispatched to the TTS service asynchronously while the LLM stream continues. This reduces first-audio latency from approximately 1,800 ms (full-response approach) to approximately 780 ms.

B. Barge-In Detection

A critical element of natural conversation is the ability to interrupt. CallMind implements barge-in detection through two complementary mechanisms. First, Twilio mark events are used to track the precise playback position of TTS audio in the telephony buffer. Second, incoming audio chunks are monitored for sustained volume above a configurable threshold while the agent is speaking. When 15 consecutive audio chunks (approximately 300 ms at 20 ms per chunk) exceed the volume threshold, the current TTS synthesis is cancelled, a Twilio clear event is sent to flush queued audio, and the session resets to accept new STT input. This approach is more reliable than volume threshold alone, which is susceptible to background noise and echo.

C. Returning-Caller Context Injection

CallMind identifies returning callers by their Twilio-provided phone number, scoped per business workspace. At call start, the agent's currently-attached knowledge base identifiers are recorded in a kb_snapshot_ids array on the conversation record. When a returning caller is identified, past conversation summaries are fetched using a PostgreSQL array overlap query—only conversations that share at least one knowledge base with the current agent are retrieved. This ensures caller history is contextually relevant even when an agent's knowledge base configuration has changed between calls.

D. System Prompt Construction

The LLM prompt is constructed by concatenating: (1) the agent's configured system prompt defining persona and behaviour constraints; (2) knowledge base document content (truncated to 60,000 characters for the LLM context window); (3) a structured summary of the caller's past conversations; and (4) the full current conversation history. This full-context approach serves as the proof-of-concept baseline, with a planned migration to vector-based hybrid retrieval (pgvector with Reciprocal Rank Fusion) in the next platform version.

E. Infrastructure and CI/CD

The FastAPI backend is containerised using Docker with a multi-stage build, deployed via Docker Compose on a Microsoft Azure virtual machine. Traefik serves as the reverse proxy with automatic TLS certificate provisioning via Let's Encrypt. Redis provides sub-millisecond session state access for agent configuration and caller history. Celery processes post-call tasks asynchronously to avoid blocking the voice pipeline.

The CI/CD pipeline uses GitHub Actions with semantic versioning via GitVersion. The develop branch triggers image build and push to Docker Hub without deployment. The main branch triggers a full build-push-deploy cycle with a manual approval gate in GitHub Environments before production deployment. This provides a safety window for rollback without sacrificing automation benefits.

V. EXPERIMENTAL SETUP AND RESULTS

A. Test Environment

Live validation was conducted using a Twilio trial account with a provisioned telephone number connected to the CallMind platform deployed on a Microsoft Azure virtual machine (2 vCPU, 4 GB RAM, Ubuntu 24 LTS). The knowledge base was configured with product information documents and FAQ entries. Test calls were initiated from standard mobile handsets in India (Mumbai region).

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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15570

B. Latency Measurement

End-to-end response latency was measured as the time from the Azure STT final transcript event to the commencement of audio playback by Twilio, covering the complete STT → LLM → TTS processing path. Table 1 presents the measured latency breakdown across pipeline stages.

Table 1. Voice Pipeline Latency Breakdown

Pipeline Stage	**Mean Latency (ms)**
Azure STT (streaming, final event)	~300
Redis cache read (agent config + KB)	~50
Groq LLM (first sentence, first token)	~200
Azure Neural TTS (first sentence)	~150
Network transmission to Twilio	~80
Total end-to-end (first audio)	**~780**

C. Functional Validation

A set of representative test scenarios was executed across live telephone calls. Table 2 summarises the test cases and outcomes.

Table 2. Functional Validation Results

Test Scenario	**Expected Behaviour**	**Result**
Basic product query	KB-grounded response < 1 s	Pass
Barge-in interruption	Agent stops, listens	Pass
Returning caller	History context injected	Pass
Health check (/health)	{status: ok}	Pass
Text query streaming	SSE delta stream returned	Pass
Invalid agent ID	HTTP 404 error response	Pass
KB-specific question	Accurate answer from KB	Pass
Concurrent sessions	Isolated session state	Pass

D. Resource Utilisation

Steady-state resource consumption during a single active call on the 2 vCPU / 4 GB RAM Azure virtual machine is summarised in Table 3.

Table 3. Resource Utilisation During Active Call

Service	**CPU (active)**	**Memory**
Traefik	<1%	~50 MB
FastAPI + uvicorn	8–12%	~180 MB
Redis	<1%	~30 MB
Celery worker	3–5% (post-call)	~150 MB
OS + Docker overhead	2–3%	~400 MB
Total	~18%	~810 MB

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

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 5, May 2026

DOI: 10.17148/IJARCCE.2026.15570

VI. DISCUSSION

The sentence-level streaming architecture is the primary latency optimisation in CallMind. By decoupling LLM generation from TTS synthesis and parallelising sentence $N+1$ generation with sentence N audio playback, the platform achieves a first-response latency of approximately 780 ms. This approaches the lower bound of natural human conversational response time (typically 700–1,500 ms), making the interaction perceptually natural to callers.

The *QueryPayload* abstraction provides architectural flex- ibility that is not common in existing voice agent platforms. Since every input channel produces an identical normalised structure, adding a new channel (WebRTC browser audio, WhatsApp messaging) requires only implementing the input normalisation step, with zero changes to the LLM service, TTS service, session management, or data persistence layer.

Full-context LLM prompting with knowledge base documents is deliberately chosen as the proof-of-concept baseline. For knowledge bases within the LLM context window (≤60,000 characters), this approach provides strong factual grounding without the complexity of vector indexing. The architecture is designed for a drop-in swap to hybrid retrieval (pgvector cosine similarity combined with PostgreSQL fulltext search via Reciprocal Rank Fusion) when knowledge bases exceed context window limits or when retrieval precision becomes the performance bottleneck.

The dual-backend architecture cleanly separates operational concerns: Supabase handles multi-tenant data isolation via Row-Level Security without application-layer filtering, while the FastAPI backend handles time-critical pipeline operations where managed services would introduce unacceptable latency. This separation also allows independent scaling— the AI pipeline can be migrated to Kubernetes without any changes to the Supabase layer or the frontend dashboard.

A current limitation is the single-VM deployment, which is a single point of failure acceptable for a proof-of-concept but not for production scale. The Docker Compose deployment is designed with a documented migration path to K3s (singlenode Kubernetes) and subsequently to Azure Kubernetes Service (AKS), with Traefik serving as the ingress controller throughout, eliminating the need for proxy reconfiguration at migration.

VII. CONCLUSION

This paper presented CallMind, an AI-powered real-time voice agent platform that achieves end-to-end telephone call handling with a response latency of approximately 780ms through a sentence-level LLM-to-TTS streaming architecture. The platform's central contribution is the *QueryPayload* abstraction, which decouples input modality from the intelligence processing layer and enables new interaction channels to be added without modifying the core pipeline.

Live validation through Twilio Programmable Voice demonstrated natural conversation quality across all tested scenarios, including returning-caller context injection, barge-in interruption, and knowledge-base-grounded responses. The dual-backend architecture—Supabase for multi-tenant data management and FastAPI for real-time AI pipeline execution— provides a scalable foundation that supports independent evolution of both layers.

The current full-context LLM prompting approach is validated as a practical proof-of-concept baseline. A documented migration path to vector-based hybrid retrieval, additional telephony channels, and Kubernetes deployment provides a clear trajectory for production-scale deployment.

VIII. FUTURE WORK

Future work includes: (1) integration of pgvector-based hybrid search (cosine vector similarity combined with BM25style full-text ranking via Reciprocal Rank Fusion) to replace full-context prompting for large knowledge bases; (2) realtime intent and sentiment classification running as a parallel asynchronous pipeline alongside the voice conversation, enabling a business intelligence analytics layer; (3) a pluggable tool

DOI: 10.17148/IJARCCE.2026.15570

framework enabling agents to invoke external APIs (calendar booking, SMS delivery, CRM lookup) during conversations; (4) browser-based WebRTC audio input and WhatsApp messaging channel integration; (5) migration to K3s and subsequently Azure Kubernetes Service for production-scale horizontal scaling; and (6) multi-provider telephony support (Vonage, Plivo) with secure per-workspace credential management via Supabase Vault.

Funding: No funding was received for conducting this study.

Competing Interests: The authors declare no relevant financial or non-financial interests.

Author Contributions: All authors contributed equally to the conception, design, implementation, experimentation, and writing of this work.

Acknowledgment: The authors thank the Department of Computer Science & Engineering, D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji for providing the infrastructure and academic support for this work. The authors also acknowledge the open-source communities behind FastAPI, Supabase, and Next.js, and the Groq team for fast and accessible LLM inference infrastructure.

REFERENCES

[1] M. McTear, Z. Callejas, and D. Griol, *The Conversational Interface: Talking to Smart Devices*. Springer, 2016.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
[3] OpenAI, "GPT-4 Technical Report," *arXiv preprint arXiv:2303.08774*, 2023.
[4] A. Radford *et al.*, "Robust speech recognition via largescale weak supervision," in *Proc. ICML*, 2023, pp. 28492–28518.
[5] A. van den Oord *et al.*, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.
[6] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. NeurIPS*, 2020, pp. 9459–9474.
[7] Twilio Inc., "Twilio Programmable Voice Documentation," https://www.twilio.com/docs/ voice, 2024.
[8] Y. Ren *et al.*, "FastSpeech: Fast, robust and controllable text to speech," in *Proc. NeurIPS*, 2019, pp. 3171–3180.
[9] H. Touvron *et al.*, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.
[10] B. Burns *et al.*, "Borg, Omega, and Kubernetes," *Queue*, vol. 14, no. 1, pp. 70–93, 2016.
[11] Supabase Inc., "Supabase Documentation," https:// supabase.com/docs, 2024.
[12] S. Tiangolo, "FastAPI Framework Documentation," https://fastapi.tiangolo.com, 2024.
[13] Traefik Labs, "Traefik Proxy Documentation," https://doc.traefik.io/traefik/, 2024.
[14] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "REALM: Retrieval-augmented language model pretraining," in *Proc. ICML*, 2020.
[15] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.
[16] Z. Ji *et al.*, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023. DOI: 10.1145/3571730

