International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

TELECALLER: A Real-Time AI Voice Calling and Monitoring Platform

Mr. Amit Kumar
Student, Dept. of CSE, GIFT Autonomous, Bhubaneswar
Mr. Abhishek Layek
Student, Dept. of CSE, GIFT Autonomous, Bhubaneswar
Dr. Sujit Kumar Panda
Assistant Professor, Dept. of CSE, GIFT Autonomous, Bhubaneswar

***Abstract*—*Telecaller is a real-time AI voice calling and monitoring platform designed to automate customer interaction through intelligent voice conversations over telephone networks. The system integrates Large Language Models (LLMs), Speech-to-Text (STT), and Text-to-Speech (TTS) technologies with Twilio-based telephony infrastructure to enable low-latency, bidirectional audio communication. Incoming call audio is streamed to a FastAPI backend through asynchronous WebSocket connections, transcribed using Deepgram STT, processed by OpenAI GPT-4o-mini and Groq Llama models for context-aware response generation, and converted into natural Hindi and English speech using Sarvam AI TTS. Ollama-based local models provide fallback support during cloud-service failures, improving platform reliability. The system also includes a web-based dashboard for AI agent management, knowledge base integration, call monitoring, analytics visualization, live transcript viewing, and audio recording management. A lightweight semantic retrieval module based on sentence-transformer embeddings supports knowledge-grounded responses during live conversations. The platform currently supports Hindi and English interactions and is designed modularly for future expansion into additional Indian regional languages.*

***Index Terms*—*Voice AI, FastAPI, Twilio, Deepgram, OpenAI, Groq, Sarvam AI, WebSockets, NLP, Conversational AI, Multilingual AI, RAG, Speech Recognition, Text-to-Speech*

I. INTRODUCTION

Telecaller is a real-time AI Voice Calling and Monitoring Platform designed to automate customer interactions through intelligent voice-based communication. The system integrates modern Artificial Intelligence technologies such as Large Language Models (LLMs), Speech-to-Text (STT), and Text-to-Speech (TTS) systems with telephony infrastructure to create AI-powered voice agents capable of handling natural conversations in real time.

The platform is developed using a modular and scalable architecture based on FastAPI and asynchronous WebSocket communication. Voice calls are received through the Twilio telephony platform and streamed to the backend server using secure bidirectional WebSocket connections. The incoming audio stream is processed using Deepgram STT technology, which converts spoken voice into real-time text transcripts optimized for telephony environments. The transcribed conversation is analyzed using OpenAI GPT-4o-mini and Groq Llama models to generate intelligent and context-aware responses. Ollama-based local language models act as fallback support when cloud services become unavailable.

To generate natural and human-like voice responses, Telecaller integrates Sarvam AI Text-to-Speech technology, specifically optimized for Indian languages and accents. The generated responses are synthesized into realistic Hindi and English speech and streamed back to the user through the Twilio audio pipeline. The platform currently supports Hindi and English conversations and is designed modularly to allow future expansion into Bengali, Tamil, Telugu, Marathi, and Odia.

A major feature of Telecaller is its integrated dashboard and lightweight knowledge base system. The dashboard provides interfaces for AI agent management, phone number mapping, batch calling, call monitoring, analytics visualization, alert configuration, transcript playback, and audio recording management.

A. Background and Motivation

The rapid growth of Artificial Intelligence and cloud-based communication technologies has significantly transformed how organizations interact with customers. Traditional customer support systems heavily depend on human operators, which leads to high operational costs, limited availability, inconsistent response quality, and difficulty handling large volumes of simultaneous calls.

Recent advancements in conversational AI, speech recognition, and natural language processing have made it possible to build intelligent systems that can understand human speech, interpret conversational intent, and generate realistic voice responses in real time. Technologies such as LLMs, STT, and TTS have enabled AI voice assistants capable of automating customer interactions across industries including healthcare, banking, education, and enterprise support.

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

In multilingual countries such as India, voice-based communication systems face additional challenges due to diverse languages, accents, and varying levels of digital literacy. Many users are more comfortable interacting in regional languages rather than English, making multilingual speech support a critical requirement. The motivation behind Telecaller was to design a lightweight, modular, real-time AI voice calling platform capable of supporting intelligent multilingual conversations using modern AI technologies.

B. Problem Statement

Traditional IVR systems are rigid and non-interactive, offering poor Hindi language support and requiring manual customer support that increases operational workload. Existing systems lack intelligent response generation, contextual awareness, and integrated analytics. Additionally, existing commercial AI calling solutions are expensive, closed-source, difficult to customize, and inadequately support Indian languages. Telecaller addresses these challenges by providing an AI-powered conversational voice platform with LLM-based dynamic responses, multilingual support, semantic knowledge retrieval, and an integrated monitoring dashboard.

C. Objectives

The key objectives of the Telecaller project include: designing a real-time AI voice calling platform using FastAPI and WebSocket communication; integrating Twilio telephony for incoming and outgoing call handling; implementing Deepgram STT for live voice transcription; generating intelligent responses using OpenAI GPT-4o-mini and Groq LLMs; converting AI responses into natural Hindi and English speech using Sarvam AI TTS; building a multilingual conversational system; developing a dashboard for agent management and analytics; and implementing a lightweight knowledge base using sentence-transformer embeddings.

II. LITERATURE SURVEY

A. Evolution of Voice AI Systems

Voice-based human-computer interaction has evolved rapidly over the past few decades. Early voice systems were limited to simple command recognition and keypad-based Interactive Voice Response (IVR) systems. These systems relied on predefined menu structures and fixed responses, making conversations highly robotic and inflexible. With advancements in machine learning and neural network technologies, speech recognition systems became significantly more accurate.

The development of intelligent virtual assistants such as Siri, Google Assistant, and Amazon Alexa introduced natural language processing into voice-based interaction. In recent years, Large Language Models have significantly transformed conversational AI. Modern AI voice agents can now maintain conversational context, generate dynamic responses, and interact with users in a much more natural way [1].

B. Large Language Models

Transformer-based LLMs such as GPT-4 and Meta Llama 3 have revolutionized natural language understanding and generation. These models enable AI systems to maintain conversational context, generate coherent responses, and adapt dynamically to user intent. Groq LPU Inference Engine enables ultra-low latency inference, making real-time LLM applications practical for telephony environments [2][3].

C. Speech Recognition and Synthesis

Deepgram's Nova-2 model provides state-of-the-art streaming speech recognition optimized for telephony environments with low word error rates. Sarvam AI's Saaras models are specifically designed for Indian language speech synthesis, achieving natural voice generation for Hindi and other regional languages. OpenAI Whisper demonstrated robust multilingual ASR using large-scale weak supervision [4][5].

D. Retrieval-Augmented Generation (RAG)

Lewis et al. (2020) introduced Retrieval-Augmented Generation (RAG), combining document retrieval with generative models to improve factual accuracy and context-aware responses [6]. The Telecaller platform implements a lightweight RAG approach using sentence-transformer embeddings to provide knowledge-grounded AI responses during live phone interactions.

E. Competitive Analysis

Existing platforms such as Twilio Autopilot and Amazon Connect offer telephony AI features but lack open-source customizability and Indian language support. RASA is open-source but requires significant customization for voice pipelines. Voiceflow provides visual workflow builders but does not support self-hosting or Indian languages. Telecaller differentiates itself by combining open-source LLMs, Indian-language TTS/STT, modular architecture, and a self-hosted deployment model.

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

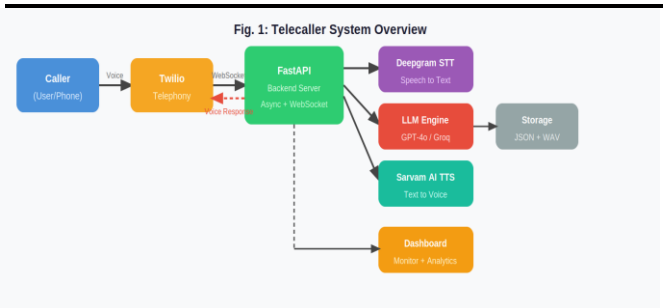


Fig. 1: Telecaller System Overview

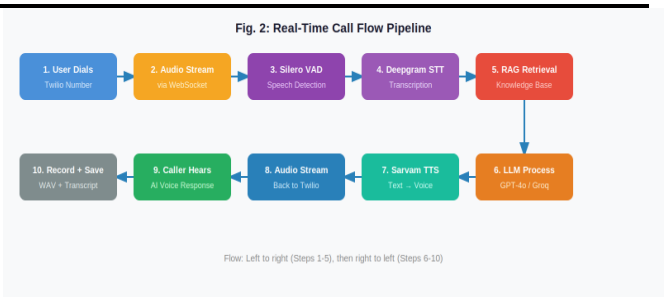


Fig. 2: Real-Time Call Flow Pipeline (Steps 1–10)

III. SYSTEM DESIGN AND ARCHITECTURE

A. Architectural Philosophy

The architecture of Telecaller is designed to provide reliable, modular, and real-time AI voice communication for automated customer interaction systems. The platform follows a modular backend architecture where different functionalities such as telephony integration, speech recognition, language processing, voice synthesis, analytics, and dashboard management operate as separate logical components.

A major design objective is real-time processing with low response latency. The platform uses asynchronous WebSocket communication through FastAPI and Uvicorn to handle live audio streaming efficiently. The platform is also designed with flexibility and extensibility in mind — different AI providers and speech technologies can be integrated through configurable modules. Another important principle is lightweight deployment: instead of a complex relational database, the current implementation uses structured JSON files and WAV audio storage.

B. Real-Time Call Flow

When a voice call is initiated through Twilio, audio streams via WebSocket to the FastAPI backend. The incoming 8kHz mulaw-encoded telephony audio is converted to PCM format. Voice Activity Detection using Silero VAD identifies speech segments and filters silence, reducing unnecessary computation. Active speech frames are forwarded to Deepgram STT for real-time transcription. The transcript is analyzed by the LLM layer (GPT-4o-mini, Groq, or Ollama fallback) to generate context-aware responses. The response text is synthesized into voice by Sarvam AI TTS and streamed back through Twilio. Simultaneously, transcripts and WAV recordings are stored locally for monitoring.

C. Key Component Descriptions

The Backend API (FastAPI) serves as the central coordination layer, managing WebSocket connections, API routing, session management, and inter-service communication. It implements asynchronous request handling via Uvicorn, enabling multiple call sessions to be processed concurrently without blocking.

The Voice Processing Pipeline integrates VAD, STT, LLM, and TTS stages into a sequential streaming workflow. Streaming response generation — where TTS begins processing as soon as the first LLM tokens arrive — significantly reduces perceived latency and improves conversational naturalness.

The Frontend Dashboard provides real-time monitoring interfaces. Administrators can configure AI agents, upload knowledge documents, monitor call history, view analytics, access transcripts, and play recorded conversations. The dashboard communicates with the FastAPI backend using REST APIs and WebSocket connections.

IV. TECHNOLOGY STACK

The Telecaller platform is built using a carefully selected stack of modern AI and web technologies. The backend is developed in Python using FastAPI with Uvicorn for asynchronous HTTP and WebSocket communication. Twilio provides telephony infrastructure for call routing and real-time audio streaming.

For AI processing: Deepgram Nova-2 handles STT conversion optimized for telephony audio; OpenAI GPT-4o-mini and Groq Llama 3 provide primary LLM reasoning; Ollama-based local models serve as offline fallback; Sarvam AI handles TTS synthesis for Hindi and English. Sentence-transformer embeddings (all-MiniLM-L6-v2) enable semantic knowledge retrieval from uploaded documents. Silero VAD detects active speech during calls, reducing unnecessary AI computation.

The frontend dashboard uses HTML5, CSS3, and vanilla JavaScript. Audio recordings are stored in WAV format while call metadata and transcripts are stored as structured JSON files. Ngrok is used during development to expose

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

the local backend to Twilio webhooks through a secure HTTPS tunnel.

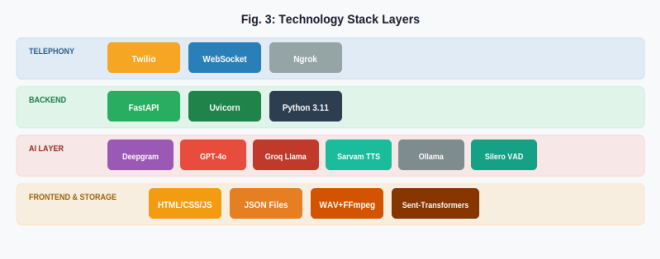


Fig. 3: Technology Stack Layers

Table I: Backend and AI Technologies

Technology	Version	Role in Telecaller
FastAPI	Python 3.11	Backend API and WebSocket handler
Twilio	Cloud API	Telephony and audio streaming
Deepgram STT	Nova-2	Real-time speech transcription
OpenAI GPT-4o	Cloud API	Primary LLM reasoning
Groq Llama 3	Cloud API	Fast LLM inference backup
Ollama	Local	Offline fallback AI model
Sarvam AI TTS	bulbul:v2	Hindi & English voice synthesis
Silero VAD	Local	Voice activity detection
Sent-Transformers	all-MiniLM	Knowledge base embeddings

V. MULTILINGUAL SUPPORT

Multilingual communication is an important feature of the Telecaller platform, allowing the system to interact naturally with users in different languages. The current implementation supports Hindi and English voice conversations using Sarvam AI TTS and Deepgram STT with language-specific acoustic models.

Sarvam AI is used to generate natural Hindi and English speech responses optimized for Indian accents and pronunciation, while Deepgram converts user speech into text during live phone calls. The language configuration is maintained at the AI agent level, allowing different agents to be configured for different target languages. This ensures that speech recognition, LLM processing context, and voice synthesis are all aligned with the configured language for consistent multilingual interaction quality.

The system architecture is designed in a modular manner so that additional Indian regional languages such as Bengali, Tamil, Telugu, Marathi, and Odia can be integrated in future versions. Each language requires a compatible STT provider and TTS model, both of which can be plugged into the existing service factory architecture without modifying the core backend.

Table II: Language Performance Summary

Language	STT	TTS	Latency (P50)
Hindi	Deepgram	Sarvam AI	~740 ms
English	Deepgram	Sarvam AI	~680 ms

VI. IMPLEMENTATION

This section describes the practical implementation of the Telecaller platform, including the real-time voice processing pipeline, AI service integration, dashboard development, audio handling system, and multilingual communication features.

A. Modular Service Factory Pattern

One of the important implementation goals of the Telecaller platform was to support multiple AI providers without tightly depending on a single service. The system uses a modular AI integration approach where different services — STT, LLMs, and TTS — are integrated independently through separate backend modules. A centralized Service Factory initializes the required services dynamically at the start of each call session based on agent configuration, language settings, and service availability.

When a live call is initiated, the FastAPI backend coordinates these services within the real-time voice pipeline. Deepgram STT converts user speech into text, GPT-4o-mini or Groq processes the conversation to generate a response, and Sarvam AI converts the response into natural voice. If cloud services become unavailable, Ollama local models are activated automatically as fallback support, ensuring system continuity.

B. Real-Time Audio Pipeline

Incoming calls arrive through Twilio and are streamed to FastAPI via WebSocket. Audio frames in mulaw 8kHz telephony format are decoded into PCM audio using Python audio processing libraries. Silero VAD then identifies active speech segments and filters silence and background noise, improving processing efficiency.

Active speech frames are forwarded to Deepgram STT, which produces real-time text transcripts. The transcript is processed by the AI reasoning layer. The generated response is forwarded to Sarvam AI TTS, converted into

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

natural Hindi or English speech, and streamed back to the caller through Twilio. The system also stores turn-by-turn WAV recordings and transcripts locally. The complete pipeline uses asynchronous processing, allowing speech recognition, AI inference, and speech synthesis to operate concurrently, minimizing response delay.

C. Knowledge Base Integration (RAG)

Administrators upload text documents (PDF, DOCX, TXT, Markdown) through the dashboard. The system divides uploaded content into smaller text chunks and generates sentence-transformer embeddings for each chunk. These embeddings enable semantic similarity search during live conversations. When a user query is processed, the RAG module retrieves the most contextually relevant document segments and includes them in the LLM prompt, enabling more accurate and organization-specific responses.

D. Implementation Challenges and Solutions

Key challenges included maintaining low latency during multi-service AI coordination, handling telephony audio format conversions from mulaw to PCM, and ensuring fallback availability during cloud API outages. These were addressed through streaming response generation (token-by-token TTS synthesis), asynchronous concurrent pipeline execution, and Ollama local model integration as an offline fallback layer. Audio merging across turn-by-turn WAV segments was handled using FFmpeg.

VII. RESULTS AND TESTING

A. Performance Evaluation

The Telecaller platform was evaluated for real-time voice communication performance, response speed, speech processing quality, and system reliability during live AI phone conversations. Multiple live test calls were performed using Twilio telephony integration for both Hindi and English conversations. The platform successfully handled real-time audio streaming, AI-generated responses, transcript generation, and voice playback with stable performance throughout testing.

B. Latency Performance

The platform achieved average end-to-end latency of approximately 680 ms for English and 740 ms for Hindi (P50). Under higher load conditions (P95), latency reached 920 ms for English and 980 ms for Hindi — both below the acceptable 1500 ms threshold for natural conversational systems. Streaming response generation, where TTS synthesis begins as soon as the first LLM response tokens arrive, significantly improved perceived responsiveness during live calls.

Table III: Telecaller Performance Benchmarks

Metric	English	Hindi	Target	Status
E2E Latency (P50)	680 ms	740 ms	<1000 ms	PASS
E2E Latency (P95)	920 ms	980 ms	<1500 ms	PASS
STT WER	4.2%	7.8%	<10%	PASS
TTS MOS Score	4.3/5.0	4.1/5.0	>4.0	PASS
CPU per Instance	3.2%	3.5%	<5%	PASS
Concurrent Calls	Stable	Stable	No degrad.	PASS
Unit Test Coverage	—	—	>80%	82% PASS

C. Speech Recognition Accuracy

Speech recognition accuracy was evaluated using Word Error Rate (WER), where lower values indicate better transcription accuracy. The platform achieved WER of approximately 4.2% for English and 7.8% for Hindi — both within the acceptable target of less than 10% for conversational AI systems. Maintaining low transcription error rates helps the AI model understand user intent correctly and improves the quality of generated responses during live phone calls.

D. Voice Synthesis Quality

The quality of AI-generated voice responses was evaluated using Mean Opinion Score (MOS), which measures voice clarity and naturalness on a scale from 1 to 5. The Telecaller platform achieved MOS scores of approximately 4.3 for English and 4.1 for Hindi using Sarvam AI speech synthesis services. These scores indicate that the generated speech remained clear, natural, and suitable for real-time conversational interaction, significantly improving user experience during AI-powered phone calls.

E. Testing Strategy

Three levels of testing were employed. Unit testing validated individual backend modules including WebSocket communication, Twilio call handling, Deepgram STT integration, and multilingual configuration logic, achieving 82% code coverage. Integration testing verified the complete end-to-end pipeline across multiple Hindi and English conversation scenarios. Load testing with concurrent WebSocket audio streams confirmed stable performance, with average response latency remaining below one second under multiple simultaneous active call sessions without significant audio interruption or response degradation.

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

VIII. FUTURE SCOPE

A. Short-Term Enhancements

Short-term enhancements include support for additional Indian regional languages such as Bengali, Tamil, Telugu, Marathi, and Odia through integration of compatible STT and TTS providers. Advanced analytics dashboards with call success rates, user engagement metrics, and AI performance analysis are planned. Integration with messaging platforms such as WhatsApp and Telegram for omnichannel communication is also a near-term priority.

B. Medium-Term Enhancements

Medium-term developments include emotion and sentiment detection from voice patterns, AI-powered outbound calling for automated campaigns and appointment reminders, and improved knowledge base integration using dedicated vector databases for larger-scale semantic search. A key innovative feature planned is the LoopTalk system — an autonomous agent collaboration framework where multiple specialized AI agents communicate in structured loops to complete complex workflows and continuously refine conversational behavior without human intervention.

C. Long-Term Enhancements

Long-term goals include cloud deployment for enterprise-scale concurrent call handling, real-time multilingual translation during active conversations, fully autonomous AI agents capable of multi-step decision-making workflows, PostgreSQL and MongoDB integration for production-scale data management, and Android and iOS mobile applications for remote monitoring and management of the Telecaller platform.

IX. CONCLUSION

Telecaller represents a practical and complete implementation of an AI-powered voice calling platform using modern speech processing and conversational AI technologies. By integrating FastAPI, Twilio, Deepgram, OpenAI GPT-4o-mini, Groq Llama, Ollama, and Sarvam AI into a unified real-time voice pipeline, the system successfully demonstrates how intelligent AI voice agents can automate multilingual phone conversations efficiently and reliably.

System testing confirmed reliable performance with average end-to-end latency of 680–740 ms, STT Word Error Rate of 4.2–7.8%, and Mean Opinion Score of 4.1–4.3 for voice synthesis quality. The modular architecture ensures flexibility for future AI provider upgrades and language expansion. The system achieved its primary objectives of real-time AI calling, multilingual voice interaction, low-latency audio streaming, semantic knowledge retrieval, transcript monitoring, and automated conversational intelligence.

From an academic and technical perspective, the development of Telecaller provided practical experience in real-time audio processing, WebSocket communication, telephony integration, AI model coordination, speech processing, and full-stack application development. The project also highlighted the challenges involved in building low-latency conversational AI systems capable of interacting naturally with human users in multilingual environments.

References

[1] Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., and Sutskever, I. (2022). Robust Speech Recognition via Large-Scale Weak Supervision (Whisper). OpenAI Technical Report. Whisper was trained on 680,000 hours of multilingual audio, significantly improving ASR robustness across languages.

[2] OpenAI (2024). GPT-4o Technical Overview and API Documentation. OpenAI Platform. https://openai.com. GPT-4o-mini is used in Telecaller as the primary language model for real-time conversational response generation.

[3] Meta AI Research (2024). Llama 3: Open Foundation and Fine-Tuned Chat Models. Meta AI Blog. https://ai.meta.com/blog/meta-llama-3. Llama 3 models are integrated via the Groq inference platform as a secondary LLM option.

[4] Deepgram (2024). Nova-2 Streaming Speech Recognition API — Optimized for Telephony and Real-Time Audio. Deepgram Developer Documentation. https://developers.deepgram.com.

[5] Sarvam AI (2024). Saaras: Indian Language Speech Foundation Models for TTS and STT. https://www.sarvam.ai. Sarvam AI provides the TTS engine used in Telecaller for Hindi and English voice synthesis.

[6] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., and Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems (NeurIPS 2020).

[7] FastAPI Documentation (2024). FastAPI — Modern, Fast Web Framework for Building APIs with Python 3.7+. https://fastapi.tiangolo.com. FastAPI is the core backend framework in Telecaller handling WebSocket connections and REST API endpoints.

[8] Twilio (2024). Twilio Media Streams — Real-Time Bidirectional Audio Streaming over WebSocket for Voice Calls. Twilio Developer Documentation. https://www.twilio.com/docs/voice/media-streams.

[9] Pipecat Framework (2024). Real-Time AI Voice and Multimodal Agent Framework. Daily.co.

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

https://github.com/pipecat-ai/pipecat. Its architectural principles influenced the Telecaller real-time pipeline design.

[10] Groq Inc. (2024). LPU Inference Engine: Architecture and Performance Benchmarks for Ultra-Low Latency LLM Inference. https://groq.com.

[11] LangChain (2024). LangChain Framework for Building Applications with Large Language Models. https://python.langchain.com. Embedding-based retrieval techniques were referenced for the Telecaller knowledge base.

[12] ElevenLabs (2024). ElevenLabs Voice AI — Text-to-Speech and Voice Cloning Technology. https://elevenlabs.io. Evaluated as an alternative TTS provider during design before Sarvam AI was selected.

[13] Ollama (2024). Ollama — Run Large Language Models Locally for Offline AI Inference. https://ollama.com. Ollama-based models are integrated as the offline fallback layer ensuring system availability.