EBOOK

STT Buyer's Guide for Voice Agents

Gladia

Table of Contents

Voice AI has rapidly evolved from early experiments to production deployment. Today, enterprises expect voice agents that **handle entire customer interactions end-to-end**, reflecting a new level of trust in the technology. Every millisecond of latency, every misheard word, and every ill-timed response can make the difference between a seamless experience and a frustrating one.

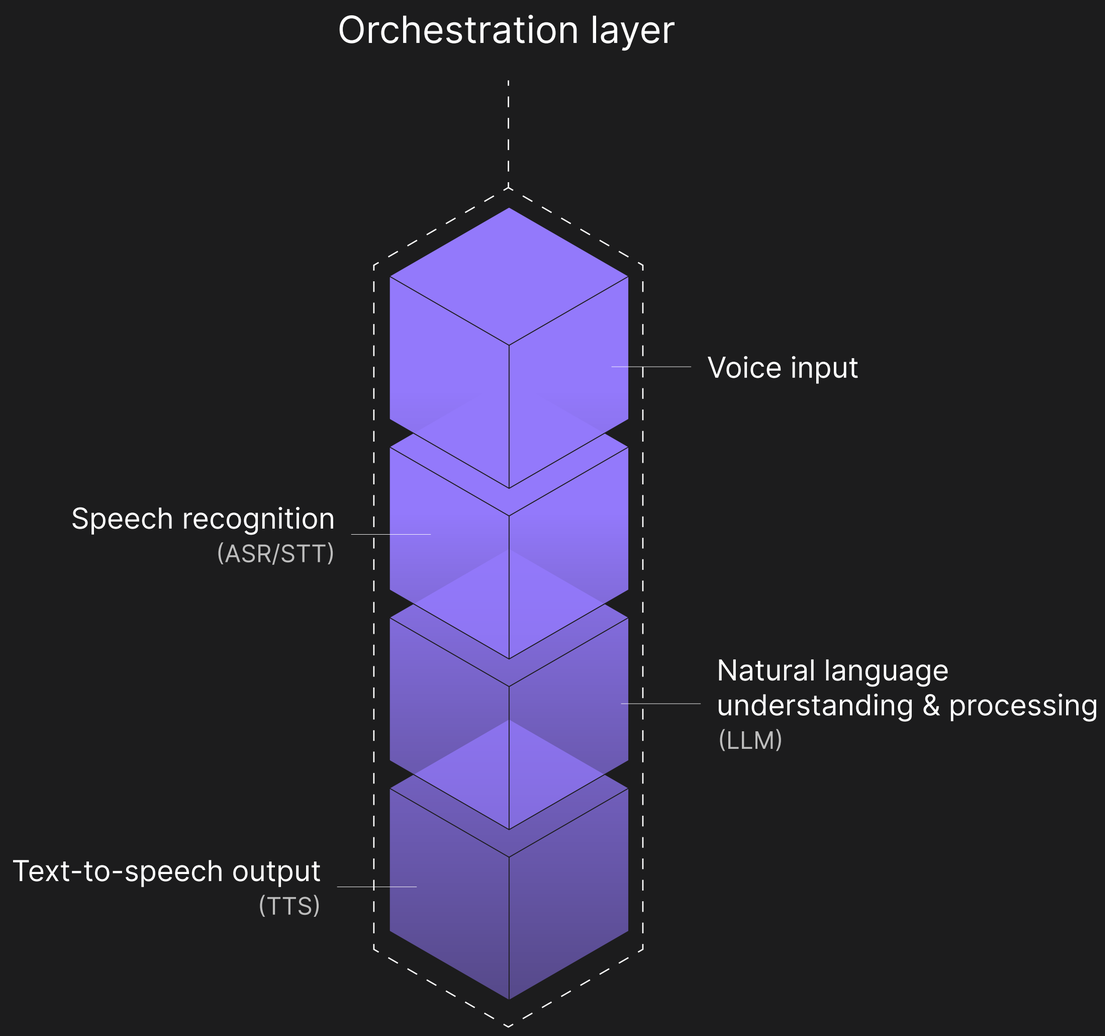
This guide is designed to **help CTOs and voice AI agent builders** make smart technology decisions for each layer of that stack – from speech-to-text (STT) and large language models (LLMs) to text-to-speech (TTS) and orchestration. We'll draw on **industry best practices** and Gladia's experience powering hundreds of audio applications to provide an actionable roadmap.

The voice agent tech stack

Building a voice agent is fundamentally an integration challenge. A voice AI system isn't a single model or program, but a pipeline of specialized components working in unison. This typically involves:

- **Speech-to-text (STT)** – converting the user's audio into text transcripts for the AI to understand.
- **Large Language Model (LLM)** – interpreting the transcribed text and formulating a response
- **Text-to-speech (TTS)** – converting the AI's response text back into spoken audio.

Surrounding these is the orchestration layer. It manages the sequence and timing of each step (including voice input and output), handles turn-taking, routes data to external APIs or databases, and enforces any business rules.



The orchestration layer makes sure the STT, LLM, and TTS engines all interoperate in concert, *in real time*.

Real-time vs cascading architecture

“Concurrency and speech-to-speech are where voice agents are heading, and **speech-to-speech (S2S)** is incredible for ultra-real-time, expressive conversations. But in the field, I still recommend the **classic STT-LLM-TTS stack** for most business use cases. S2S today often means **limited voice and feature control**, **proprietary lock-in**, and **higher costs at scale**, plus fewer **mature open or on-prem options.**

The modular pipeline is slower by a fraction of a second, but it’s **proven, swappable, easier to customize** or **clone voices**, and **simpler to deploy privately** for compliance. For clients, that flexibility and control usually wins.”

Nour Siakir Oglou

Voice AI developer & CEO

SANAVA

Speech-to-text (STT)

The STT engine is the *foundation* of a voice agent platform.

Its accuracy, latency, and streaming capabilities set the ceiling for your agent's overall performance and user experience. There are a few core dimensions to focus on for STT in voice AI:

- Concurrency,
- Latency optimization,
- Real-time processing and partials,
- Accuracy & robustness,
- Multilingual support,
- Features.

Concurrency

To minimize latency, make sure your STT, LLM, and TTS components (and any intermediate logic) can all **run in parallel** and that your orchestration properly coordinates those parallel tasks.

Achieving this requires **careful software engineering** (threading, async I/O, message queues, etc.) and often a **microservices architecture** where STT, LLM, and TTS components operate independently and communicate through streaming APIs or websockets.

Concurrent STT also means managing throughput carefully: inquire about provider **rate limits on concurrent websocket streams** and requests per second to avoid throttling or dropped sessions.

Design your orchestration to **batch, queue, or shed load** when you're close to those limits, and to prioritize active conversations over background jobs. Align these patterns with your STT provider's documented concurrency quotas and upgrade paths as traffic scales.

Tactical checklist

- [ ] Ensure that the vendor's API or SDK supports **streaming and concurrent processing.**
- [ ] Design your internal workflow to **avoid any single-threaded bottlenecks**. Break the problem into pieces that can overlap.
- [ ] **Embrace asynchronous thinking**: use async I/O and handle events ("user finished speaking" or "LLM produced first token") to trigger next actions.

Latency

In voice AI, **latency is defined as the time from the end of the user's utterance to the start of the agent's response**. It's the sum of all the processing delays in the pipeline. To make a voice agent feel truly real-time, a common benchmark is to get that round-trip latency under 500 ms or better. For that, you need to optimize above all STT latency.

You should always **measure time-to-first-byte/token (TTFB**, a.k.a. TTFT). TTFB is how long it takes from input to the **first partial output**. An STT that begins showing text in 200 ms versus one that takes 800 ms makes a big difference in perceived responsiveness.

Infrastructure: If you self-host, beware of cold starts. Spinning up a GPU on-demand can take many seconds, which is unacceptable in a live call. Keep models "warm" or use autoscaling.

Tactical checklist

- [ ] **Enable and act on partial transcripts**: see the next page for more details.
- [ ] **Optimize each hop.** If you're calling third-party APIs (e.g., to get account info), cache or pre-fetch frequently needed data.
- [ ] **Measure in realistic scenarios.** On actual audio from calls, incl. network transmission time to/from the STT service.
- [ ] **Use logs** to capture timestamps at each stage and identify bottlenecks.

Partial results

Achieving true real-time transcription means your STT should start producing text within a few hundred milliseconds of the user speaking and keep updating as more speech comes in. In practice, this is done via **partial transcripts**: interim results that update word-by-word (or even character-by-character) before the final transcript is confirmed.

Used well, partial transcripts drastically **reduce perceived latency** and make the agent seem natural and attentive. Initial partial results often arrive within **100–200 ms** of speech onset – essentially in real time. Those early words can be used to start database lookups or feed an LLM so that by the time the user finishes a sentence, the agent is already halfway done thinking of the answer.

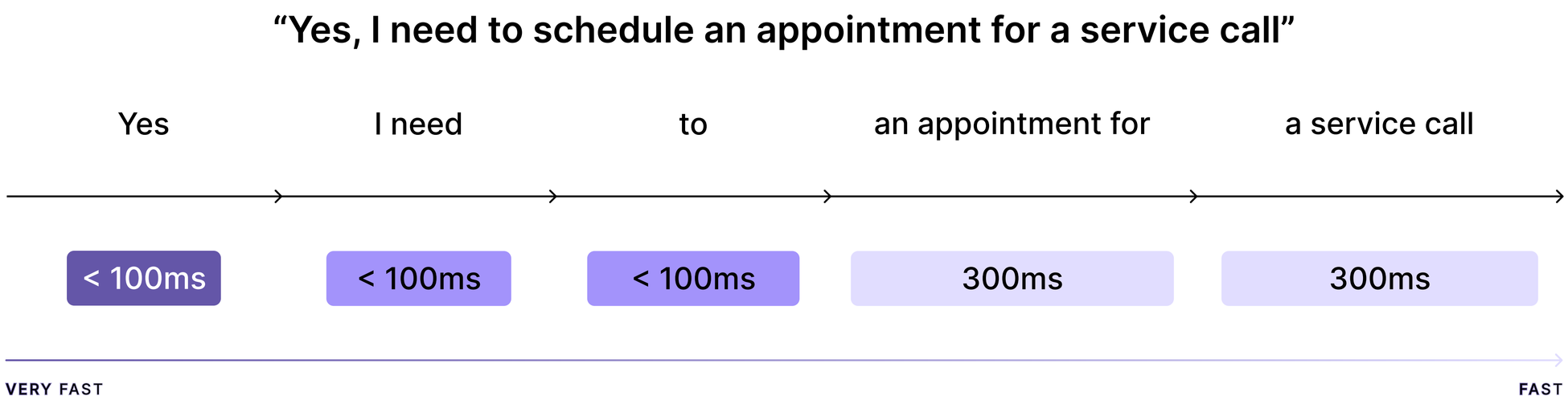
Studies have found that users begin forming an impression of system responsiveness within about **300 ms** of finishing their turn, so hitting that sub-300ms “time to first word” threshold is a good target.

Tactical checklist

- [ ] Look at both the **latency** of partials and the **quality** (are the partials reasonably accurate, or so unstable that they’re useless?)
- [ ] **Use confidence thresholds**. Only forward partials that exceed a chosen certainty score: typically 0.7-0.8.
- [ ] **Experiment with “speculative” responses.** Start formulating an answer before the user finishes; be ready to discard/adjust if intent changes.

What makes Gladia stand out?

Gladia's real-time STT delivers initial partials in **~100 ms** (up to 2x faster than the leading alternative) and a time-to-first-word in **<300 ms**, and streams on the fly. That level of performance enables voice agents to start "thinking" almost as soon as the user begins speaking, with limitless parallel streams to support your growth.



Accuracy

Accuracy is the other side of the STT coin. Speed means nothing if the transcribed words are wrong. **Transcription errors can directly lead to AI errors** – the LLM might completely misunderstand the user's request if a key word is misheard.

When we talk about STT accuracy, it's more than just overall Word Error Rate (WER). **Domain-specific accuracy** matters a lot. This includes accents or dialects, industry jargon or product names, and background noise conditions (VoIP noise, street noise, etc.). Many general models are **trained on clean, scripted English** (e.g. audiobooks), and their accuracy can drop sharply with heavy accents, cross-talk, or niche vocabulary.

Another aspect to be mindful of is **robustness to disfluencies** (ums, ahs, repetitions). Some STT engines have options to exclude filler words. For a voice agent, you want to ignore the "ums" but catch self-corrections (e.g. "I need a ticket to Boston – I mean to Austin").

Tactical checklist

- [ ] **Use recordings from your own users/ environment.** Don't just rely on a provider's quoted WER on standard datasets.
- [ ] Evaluate STT accuracy on your **target languages, accents, and jargon**, ideally using *real sample audio*
- [ ] Test whether **key domain terms** relevant for your use case (e.g. phones, emails) are recognized accurately and consistently.

Multilingual support

If your voice agent will serve users in multiple languages (or even just languages other than English), you need to bake in multilingual support at the core of your stack. This means both **the STT and the LLM** (and possibly the TTS) must handle different languages – ideally within the same conversation (code-switching).

Beyond just language, consider **cultural context**. The LLM might need different prompt tuning for different languages to maintain the same persona or politeness level. For example, how a support agent speaks in Japanese (very politely) versus in American English (more informal) will differ. If using one LLM for both, you might adjust the system prompt based on language context ("In Japanese, be very polite and use -masu/-desu form." etc.)

Tactical checklist

- [ ] Decide early **whether multilingual capabilities matter** for your use case. If yes, make sure your architecture supports it.
- [ ] Ensure either your chosen **LLM is multilingual** or plan for a **translation layer or multi-model setup.**
- [ ] **Evaluate STT in all target languages/ dialects**. Don't assume "supports X language" equals "good at X language." Test it.

What makes Gladia stand out?

Gladia offers the broadest and most accurate language support of any STT provider on the market—powering real-time transcription, code-switching, and translation across 100 languages. Whether you're enabling a global workforce or supporting multilingual customers, Gladia makes it possible to serve users fluently, no matter where they are.

True global language support

Gladia supports 100 languages, including 42 underserved languages that aren’t available from other leading STT providers—covering high-population markets like Bangladesh, India, and the Philippines.

Fine-tuning by language(s)

For even greater performance, developers can pre-set one or multiple expected languages in a call or conversation, reducing misclassification and speeding up transcription—ideal for contact centers and multilingual products.

Robust code-switching capabilities

Gladia can handle real-time language mixing within a single conversation—essential for any real-time voice product built for the chaos of multilingual conversations.

Accuracy across dialects & accents

Unlike models that falter outside of English or clean audio, Gladia was trained and evaluated to perform reliably across regional variations and accent-heavy speech.

Features

Modern STT platforms often come with a suite of additional features beyond plain transcription. When comparing options, look at:

Voice activity detection (VAD)

VAD detects when a user starts and stops speaking, while endpointing defines how many seconds of silence are required before the STT closes an utterance.

Named entity recognition (NER)

Keyword spotting features identify and categorize specific information—like names, locations, organizations, dates, emails, and phone numbers.

Real-time audio insights

Some platforms can do things like emotion or sentiment detection from the voice, or detect keywords that were spoken (independent of the transcript).

Custom vocabulary

The ability to teach the model new words (e.g. product names, industry-specific terms) can be critical for achieving high accuracy in niche domains.

Tactical checklist

- [ ] Tune **VAD + endpointing**: Benchmark VAD and endpointing across real calls to minimize hallucinations while avoiding premature cutoffs on long pauses.
- [ ] Configure **NER** or **keyword spotting** to tag entities that trigger routing, alerts, or downstream workflows in your stack.
- [ ] Integrate **real-time sentiment** or **emotion scores** into orchestration to dynamically adjust dialog paths, escalation rules, and incentives.
- [ ] Maintain and regularly update your **custom vocabulary**; track errors to decide when deeper fine-tuning is justified.

What makes Gladia stand out?

Gladia’s next-gen STT model doesn’t just transcribe—it powers voice agents with advanced real-time audio intelligence features and native integrations with voice AI toolkits and frameworks. We enable teams to ship richer voice experiences without stitching together multiple vendors.

Voice Activity Detection (VAD) control

Gladia’s speech_threshold (0–1) fine-tunes noise filtering. A 0.999 setting keeps only direct mic input—ideal for noisy call centers. Lower values allow more ambient speech. VAD control adapts to your environment.

Stress-tested across diverse datasets

We benchmark on datasets like Mozilla Common Voice & Google FLEURS for accent and audio diversity. Unlike vendors tied to one benchmark, we test across versions—including real-world data—for true robustness.

Adaptability to specific use cases

While you get high accuracy out of the box, you can easily fine-tune Gladia to your terminology, audio environments & user speech patterns. This is especially useful for improving recognition of repeated or unique phrases, without the overhead of building custom models from scratch.

Named Entity Extraction (NER)

We go beyond transcription by extracting structured data like names, orgs, and locations. Powered by proprietary ASR systems, this enables fast, accurate capture for CRMs, automation, and insights from any audio.

Integrates natively with

VAPI Pipecat LiveKit

Building production-grade AI voice agents

Thoughtly, a US-based startup, is pioneering enterprise-grade voice agents across phone, SMS, and more. Some lessons from their journey:

- **Evals and benchmarking were crucial**. They built internal eval tools to test each STT/LLM/TTS combination using real call replays and clear metrics. This let them compare vendors objectively and improve the stack based on data, not hype.
- **Latency was a top priority from day one**. Thoughtly knew that even a one-second delay breaks a conversation, and optimized every stage of the pipeline, using techniques like speculative generation and vendor race conditions to cut milliseconds wherever possible.

They emphasized that the rise of third-party evaluation tools, like Coval for voice AI, signals a more mature ecosystem. Takeaway: build a solid **evaluation layer** for your voice agents – combine offline benchmarks with live metrics like containment rate.

On **scaling**, Thoughtly recommends relying on regional infrastructure to reduce latency, aggressive caching, and lightweight websocket streaming. They also learned that over-provisioning hides latency problems but drives costs up.

For **compliance** (HIPAA, SOC2), they noted that chaining too many third-party services makes audits significantly more complex. Their advice: keep the vendor stack as lean as possible.

THEIR BUILD-VS-BUY INSIGHT

AI models improve so quickly that self-hosting rarely provides a long-term edge. Maintaining GPU infrastructure and model updates becomes a distraction. Focus effort on your own value layer – the workflow, the orchestration, and the UX – and let specialized APIs do the heavy lifting.

The hidden downsides of self-hosting

“This time last year, a **speech-to-text API** typically took 400–700 ms to respond. A **self-hosted model** may have been needed then in order to hit <1 minute latency. Now benchmarks are at **50–150 ms**. At that point, the benefit of self-hosting disappears — especially when you factor in **infra cost**, **server hops,** and **cold-start delays**.

There’s no standard eval in voice AI (yet). Every company has to build its own. All the more important to build a robust, reproducible, and **reliable evaluation layer**. There are a lot of vendors in the market, and they’re not all created equal. Having great evals and benchmarking saves you from wasting months chasing the wrong tool.”

Alex Castella

Co-founder & CTO

thoughtly

Large Language Model (LLM)

If STT is the ears of the voice agent, the LLM is the brain.

This is the component that actually “understands” the transcribed text and decides how to respond. A voice agent’s conversational intelligence – how helpful, coherent, and on-brand it is – hinges on the LLM’s capabilities.

In practice, these needs mean evaluating the following LLM qualities:

- Latency and streaming quality,
- Dialogue memory and interactivity,
- Output control, style, and error handling,
- Practical fit, integration, and cost.

LLM vendor considerations

Latency and streaming quality

Voice agents depend on immediacy. The best LLMs return their first token (TTFT) in <400 ms and stream outputs at a smooth, conversational pace. Consistent token pacing helps downstream TTS systems generate fluid, natural-sounding speech. Avoid models that produce output in unpredictable bursts or long pauses as it disrupts turn-taking and erodes realism.

Output control, style & error handling

A strong voice LLM maintains context over multiple turns, e.g. tracking references like “that change” or “the second one.” Look for a model with a large context window and good long-range memory use. Additionally, voice agents must support mid-sentence interruptions and recovery. Some models handle this smoothly; others require orchestration to restart generation.

Dialogue memory & interactivity

LLMs should produce brief, on-brand, and friendly responses. Some default to verbose or robotic styles; mitigate this with careful prompting or prompt templates. Avoid hallucinations with grounding: RAG, system prompts, or external fact injection. Also, ensure graceful error handling: LLMs should say “I don’t know” instead of guessing.

Practical fit, integration & cost

Evaluate if the model supports external tool use (e.g., APIs, function calls), RAG integration, and fallback behavior. Balance performance with cost: consider using high-quality models for complex queries and faster, cheaper ones for routine interactions. Hosting your own model may reduce variable costs but introduces engineering and infrastructure tradeoffs.

Tactical checklist

- [ ] Ensure **low time-to-first-token** and smooth streaming for natural-sounding voice playback.
- [ ] Choose LLMs with **multi-turn memory** and pause/resume behavior for real-time interactivity.
- [ ] Prompt or fine-tune for **brevity, friendly tone, and hallucination resistance** under pressure.
- [ ] Evaluate **tool-use support** and match model cost/performance to expected traffic patterns.

Text-to-speech (TTS)

The text-to-speech layer is the voice of your agent.

And in many ways, TTS is the unsung hero of voice agents – a user will often judge the agent's quality by how natural and pleasant the voice sounds, regardless of the intelligence behind it.

Some of the key TTS considerations are:

- Real-time streaming and natural turn-taking,
- Voice quality and persona adaptability,
- Accuracy and dynamic handling on key domain entities, barge-in and interruptibility,
- Scalability and cost,
- Multilingual & advanced feature support.

TTS vendor considerations

Real-time for natural turn-taking

Modern TTS should generate speech at the word or phoneme level. This allows playback to begin milliseconds after the LLM starts producing output. Ideally, users never notice a delay between speaking and hearing the system reply: this real-time overlap is essential for natural turn-taking.

Voice quality and persona alignment

Speed alone isn’t enough. The selected voice must reflect the brand's personality—whether friendly, calm, formal, or energetic. High-quality TTS also manages prosody effectively: rhythm, intonation, and stress patterns ensure the voice sounds engaging, not flat or robotic.

Handling unpredictable content

Voice agents must pronounce names, numbers, dates, addresses, and domain-specific phrases on the fly. A production-ready TTS engine needs robust text normalization and support for custom pronunciations, acronyms, and alphanumeric sequences.

Barge-in and interruption handling

Spoken conversations are interactive. TTS systems must support barge-in – stopping speech instantly when the user interrupts, to avoid frustrating overlap. This feature keeps exchanges fluid and helps the agent feel more attentive and conversational.

Multilingual support

If your use case requires it, ensure the TTS supports multiple languages or can seamlessly switch voices when the language changes.

Infrastructure and scaling

Cloud-hosted TTS often handles concurrency well, but if you're self-hosting or anticipating heavy load, assess cold starts, throughput, and cost.

Tactical checklist

- [ ] If using multiple voices/languages, choose ones that **match tones** and support **expressive prosody**.
- [ ] Ensure your TTS provider offers support for **custom pronunciations, numbers, and multilingual** input
- [ ] Test extensively vendors’ **barge-in capabilities** (on real audio!) for clean, interruption-ready conversations.
- [ ] Always evaluate scaling, load behavior, and estimated cost under **real-time usage conditions**.

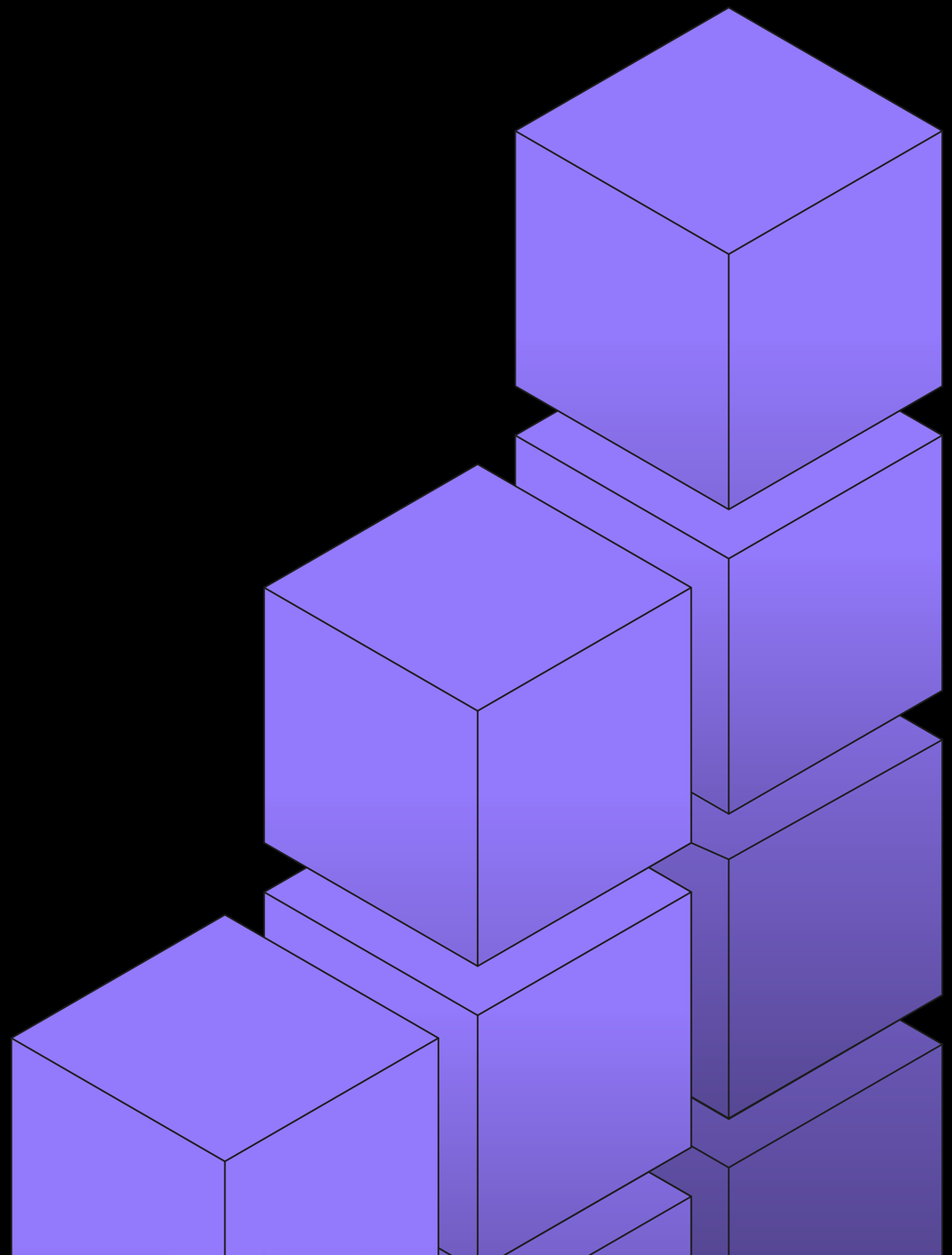
Final thoughts

A voice agent isn’t just a sum of STT+LLM+TTS.

It’s the cohesive experience of a helpful, fast, and trustworthy conversation. Achieving that means optimizing each layer, but also continually testing the end-to-end experience as users will see (or hear) it.

Gladia’s approach – and what we’ve tried to convey in this guide – is to use best-in-class technologies for each layer and fine-tune how they interact.

By leveraging Gladia’s real-time STT, pairing it with an appropriate LLM strategy through a robust orchestration, and delivering responses via high-quality TTS, you can build a voice agent that feels cutting-edge and reliable without reinventing the wheel at every layer.



Gladia

The speech-to-text backbone for voice platforms

Everything starts with reliable transcription.
Learn more at gladia.io.

Talk to an expert

Trusted by 500+ AI agents and contact center platforms

JELLYSMACK · method · recall · VEED
alan · mojo · Spoke · Selectra
Brut. · Circleback. · aircall · Attention