IIElevenLabs

Building AI voice agents at scale

An overview of implementations, tradeoffs, and enterprise considerations

Contents

Introduction

Enterprises are reimagining customer experiences through conversational voice agents – systems capable of natural, real-time, human-like dialogue at scale. The strategic question is no longer whether to adopt voice AI, but how to implement it in a way that balances quality, control, speed-to-market, compliance, and scalability.

The choice for most enterprises comes down to how much of the orchestration they want to own themselves, and how much they prefer to have handled by a vendor. This guide takes ElevenLabs' voice AI technology as its foundation and explores the technology stack, the three dominant implementation approaches in the market today, the trade-offs each entails, and how to make the right decision for your enterprise - illustrated with real-world case studies and supported by a decision framework you can apply immediately.

The Technology Stack Behind Conversational Voice Agents

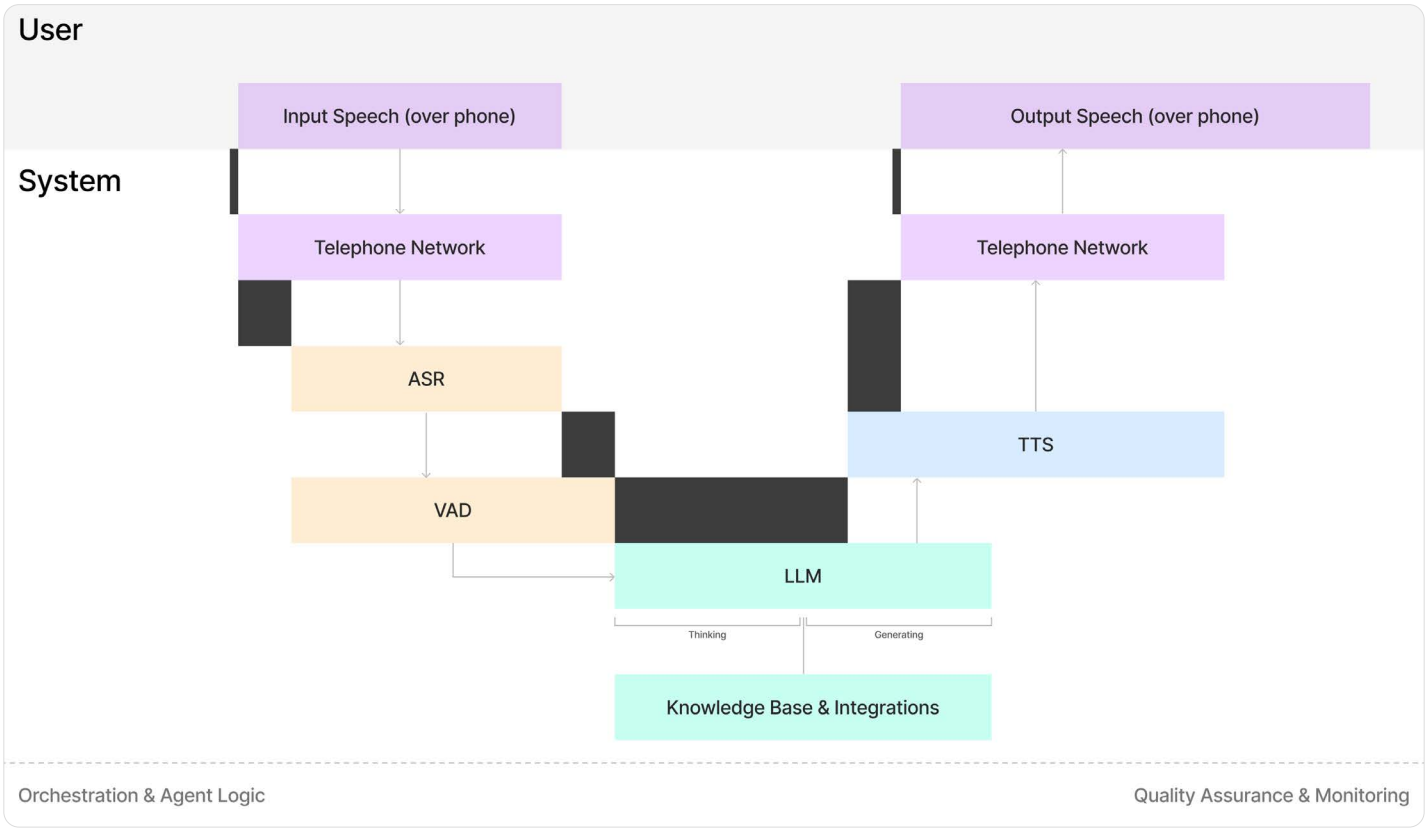
Implementing a conversational voice agent requires integrating multiple technology layers so that the system can listen, understand, think, and respond like a human and actually resolve your customers' inquiry. Below we break down the key components of a typical voice-agent stack and what each does:

Audio Input & Telephony Interface: At the front line is the interface to capture the user's voice. This could be a microphone in a mobile app or smart device, a WebRTC connection in a browser, or a telephone network (PSTN/SIP) for call centers. In telephony scenarios, you'll handle call routing, DTMF tones, and possibly connect through contact center platforms or SIP trunking.

Voice Activity Detection (VAD) & Turn-Taking: Determines when the user starts and stops speaking. In a well-designed system, the probability of the end of a turn is determined from both the semantic meaning of the transcript and the prosodic cues - such as tone, pitch, pacing, and pauses - that indicate how the user is delivering their words.

Speech to Text (STT) / Automatic Speech Recognition (ASR): Transcribes the spoken words into text.

Natural Language Understanding & Dialogue Management (LLM or NLU engine): Interprets intent and manages conversation flow - keeping track of context (conversation state) and ensuring conversation flows logically, with knowledge base facts.



Knowledge Bases & Integrations: Retrieves data (FAQs, documents, vector database for semantic search) and connects to APIs/databases as needed.

Orchestration & Agent Logic: Manages the sequence of steps for each turn of conversation. For example, upon receiving audio: (1) get STT result, (2) feed to LLM along with context, (3) get LLM's text response, (4) send to TTS for voice output, all while handling timeouts, errors, and partial results.

Text to Speech (TTS) / Voice Synthesis: Converts the text response into natural, intelligible, brand-aligned audio. Supports multiple languages.

Quality Assurance & Monitoring: Though not a "layer", tracks performance, logs interactions, and drives continuous improvement in the real-time pipeline. It's worth noting that enterprise deployments need surrounding capabilities like logging, analytics, and human review tools.

Monitoring transcripts and audio for quality, measuring metrics like error rate, user satisfaction, and having a process to continuously improve the agent are key to success. Also, safeguards like content filters and fallback handling are part of a mature voice agent stack.

All these components must work seamlessly together. If one piece is weak, the whole customer experience suffers. For example, a brilliant dialogue model won't matter if your STT misses every third word in a noisy call center. Or a high-quality TTS voice won't impress if responses come a full 5 seconds late due to slow orchestration.

Latency, in particular, is a common bottleneck - and we've outlined practical strategies for minimizing it in conversational AI here.

> "The quality bar is high to entrust calls to an AI—and the conversational flow, plus the backend workflow on the customer's side, can quickly become complex and highly specific," notes a16z in their analysis of AI voice agents.

In other words, even with the right technology stack in place, making all the layers work together seamlessly is no small feat.

The Enterprise Decision Spectrum: Modern Implementation Approaches

The old “build vs buy” framing no longer reflects how enterprises actually approach conversational AI, because training STT, TTS, VAD, or LLM models from scratch has become increasingly complex and resource-intensive. Only a small number of hyperscalers or research-driven organisations pursue it only where voice AI development itself is considered a critical and core product differentiator.

Most enterprises instead face a more pragmatic choice of how much orchestration to own versus how much to delegate to vendors. Today, most implementations fall into one of three patterns, each representing a different balance of control, flexibility, and reliance on third-party service providers.

1

Building with Blocks (Pipeline / Chained Approach)

In this approach, you assemble a conversational agent from best-in-class APIs for each major function – STT, LLM, TTS, VAD – and integrate them through your own orchestration logic. Your engineering team is the conductor, determining how each component interacts, what optimisations to apply, and how to tune the flow for your business priorities.

Some organisations use frameworks like LiveKit or Pipecat to offload complex behaviours like real-time audio streaming, barge-in, and VAD, enabling them to focus on conversation logic and integration.

Building with Blocks offers the highest degree of control: third-party service providers can be swapped at any layer, orchestration can be fine-tuned for ultra-low latency or regulatory compliance, and you decide how deeply to integrate with internal systems. The trade-off is complexity and responsibility. You own scaling, error handling, and performance tuning – and that demands strong, dedicated engineering capability.

Pipeline approach

Core idea

You build everything yourself, using APIs as raw ingredients.

How it works

Your engineering team acts as the conductor, wiring together best-in-class APIs for STT, LLM, TTS, and VAD, plus custom orchestration logic. You manage real-time audio streaming, error handling, scaling, and compliance. Frameworks like LiveKit or Pipecat can help, but responsibility still sits with your team.

Analogy

Building your own orchestra from scratch. You hire every musician (API), write the score (orchestration code), and conduct the performance. Maximum flexibility, maximum responsibility.

2

Conversational AI Products (Orchestration-First Approach)

Here, the vendor delivers a complete, pre-orchestrated speech pipeline that manages the STT > LLM >TTS loop, along with turn-taking, streaming, and performance optimization.

Within this category are orchestration-first products, like ElevenLabs' Conversational AI platform, which lets you bring your own LLM, prompts, and tools while we handle the speech layer; and speech-to-speech products, which take audio in and return audio out without exposing intermediate text.

This model enables the fastest route to market, compressing deployment from months to days. It shifts operational responsibility to the vendor, who manages scaling, uptime, monitoring, and compliance. Crucially, it also means enterprises continuously benefit from the vendor's R&D improvements: every latency reduction, quality upgrade, or compliance enhancement is delivered automatically.

Orchestration-first approach

Core idea

The vendor owns orchestration - you just configure.

How it works

You plug into a complete, pre-orchestrated pipeline that handles STT > LLM > TTS, turn-taking, latency optimization, monitoring, and compliance. ElevenLabs' Conversational AI platform is one example: you bring your LLM, prompts, or tools, and the platform handles the speech loop and scaling.

Analogy

Renting a full orchestra with a professional conductor. You only decide the playlist (prompts, integrations). Fastest to deploy, lowest lift, but limited flexibility in orchestration.

For many enterprises, particularly those with standardised use cases like customer support or voice-based ordering, this is the most efficient and lowest-risk route. The trade-off is reduced orchestration flexibility and potential vendor lock-in – but the benefits of continuous improvement and immediate time-to-market often outweigh these concerns.

3

Hybrid (Framework-Orchestrated, Model-Flexible)

Here, you offload orchestration infrastructure to a framework, while keeping freedom to choose your models.

The distinction from Building with Blocks is important: in Blocks, frameworks like LiveKit or Pipecat may just serve as helpers, while orchestration logic is still built and owned by your team.

Hybrid approach

Core idea

Use a framework to handle orchestration, while you still choose and integrate your own STT, TTS, and LLM models.

How it works

The framework manages VAD, turn-taking, and streaming so you don't have to write orchestration logic. You just plug in the APIs and connect them to your business logic.

Analogy

Like hiring an orchestra but bringing in world-class soloists. The orchestra (framework) handles timing and coordination, while you still choose the star performers (APIs) for the parts that matter most. Faster than Blocks, more flexible than Products, but some integration effort remains.

In Hybrid, those same frameworks become the orchestration backbone itself - managing low-level mechanics such as voice activity detection (VAD), turn-taking, streaming, and barge-in.

On top of that foundation, your team integrates preferred APIs for STT, TTS, and LLM. This gives you speed and reliability while retaining flexibility to swap in the best vendors for each model.

Hybrid deployments appeal to teams that want to move faster than a full Building with Blocks approach, yet avoid the constraints of a single-vendor product. The trade-off is that integration work remains, and performance depends on how well your chosen components interoperate.

When weighing these three approaches, the trade-offs become clearer when viewed through the lenses of time to value, continuous monitoring/improvement, and specialisation of expertise.

Time to Value

Conversational AI Products deliver the fastest deployment, often in days, because orchestration is fully abstracted. Hybrid accelerates development compared to a full pipeline, but still requires careful integration. Building with Blocks provides the most control but carries the slowest time-to-market.

Continuous Monitoring & Improvement

Vendors continuously enhance latency, accuracy, and compliance in their managed products – improvements customers inherit instantly, without additional engineering. In pipeline builds, the burden of staying state-of-the-art falls entirely on your team. Hybrid sits in the middle: some benefits flow through from chosen vendors, but orchestration upkeep remains your responsibility.

Expertise and Specialisation

Very few enterprises build operating systems from scratch; they buy them and innovate on top. The same logic applies here: orchestration platforms and managed products encapsulate years of R&D, hard-won operational know-how, and specialist expertise. Unless conversational AI is itself your differentiator, it rarely makes sense to replicate that effort.

Broader Comparison Matrix

Dimension	**Building with Blocks (Pipeline / Chained Approach)**	**Conversational AI Products (Orchestration-First Approach)**	**Hybrid**
Cost	High upfront (talent, infra, integration); potentially efficient at scale	Low upfront, predictable OpEx; vendor manages infra & R&D	Moderate; framework + multiple vendors
Time to Value	Slowest (months)	Fastest (days–weeks)	Faster than Pipeline, slower than Products
Control & Customisation	Maximum	Limited at orchestration layer	Moderate; flexible in model choice, less in orchestration
Performance	Tunable but complex to optimise	Optimised by vendor out-of-the-box	Dependent on vendor APIs + integration quality
Latency	Full tuning possible, but hard	Vendor-optimised	Framework + vendor dependent
Team Requirements	Large, specialist engineering	Minimal, light integration	Mid-sized, strong integration engineers
Scalability	Full responsibility	Vendor handles automatically	Split responsibility
Compliance & Security	Full control, full responsibility	Vendor-certified, SLA-backed	Wider attack surface (multi-vendor)
Quality of Experience	Differentiated if executed well	High baseline, steadily improving	Mixed, integration-dependent

Real-World Case Studies: Approaches in Action

To ground this discussion, let’s briefly look at how several organizations approached their voice conversational AI deployments and the outcomes they achieved.

Telus Digital (Telecommunications) – Internal Build with External APIs

Telus, a major telecom, sought to improve its call center operations, not just for customers but for employee training. They developed an in-house **"Agent Trainer"** simulation tool that uses AI voice agents to role-play customer calls for new human agents. Rather than buy generic training software, Telus built this custom solution using ElevenLabs' voice AI to ensure ultra-realistic call scenarios.

Results

- Reduced onboarding and upskilling time for call center agents by 50% by enabling unlimited, on-demand practice conversations.
- New agents ramp up **25% faster** because the practice feels real, and they gain confidence quickly.

This is a great example of using a **bespoke approach** for a very specific internal use case – Telus kept the innovation in-house (to tailor scenarios to their exact needs), but leveraged ElevenLabs' Text to Speech via API to provide the human-like voices at scale. The success metrics show that a well-judged build with the right API partner can deliver concrete ROI.

Meesho (E-commerce Marketplace) – Custom Voice Bot via APIs

Meesho, one of India's largest online marketplaces, faced the challenge of supporting a massive, multilingual customer base without scaling up contact center costs. They opted to create a **voice customer support** bot in-house, orchestrating various components but using ElevenLabs for the speech output. Their voice agent handles common queries like order status, delays, and cancellations in both Hindi and English.

Results

- ~90% of incoming support queries automation with voice agents, handling 60,000+ calls per day
- They also reported around 40% reduction in average handling time for calls

This case demonstrates a hybrid approach: Meesho built its own conversational logic (likely integrating STT, an LLM and hooking into their order database) – a custom build for their business process – but **they bought key components** to accelerate development. It shows how voice AI can be leveraged without a giant team, by smartly combining APIs.

Funding Societies (Financial Services) – Orchestration API route

Funding Societies, a leading Southeast Asian SME digital financing platform, sought to streamline loan application and servicing calls across multiple markets and languages. They built their own business logic and integrated with CRMs and compliance workflows, but chose ElevenLabs' Conversational AI Orchestration API for the speech layer.

This meant they could launch fully functional, **multilingual loan servicing voice agents in weeks, not months, without hiring specialised audio engineers.**

Results

- Over 1000 daily calls automated - improving reach, consistency and voice quality across markets
- Scalable, always-on engagement layer that frees human teams to focus on depth over volume.

Decision Framework

When to Build with Blocks (Pipeline / Chained Approach)

Choose this if conversational AI is a core, long-term differentiator for your business and you have the engineering resources to invest. Best for creating highly customised customer experiences, achieving ultra-low latency optimisations, or meeting compliance requirements no vendor can. Expect a multi-phase journey with ongoing iteration, and be prepared to own the integration and improvement cycle yourself.

When to Adopt a Managed Product Approach (SaaS)

Choose this path if time-to-value is your top priority. A managed product approach (SaaS) delivers ready-to-deploy conversational AI with minimal setup, allowing you to launch in days, not months. It's especially effective for proven, repeatable use cases such as call centre augmentation, customer support automation, or voice ordering systems – where reliability and speed matter more than reinventing the wheel.

The biggest advantage is that you inherit the expertise and continuous improvement of a specialised provider. Just as Fortune 500 companies don't build their own operating systems to run laptops, enterprises shouldn't have to build and maintain their own STT, TTS, or VAD engines. With a managed product, latency optimisations, compliance updates, and voice quality enhancements arrive automatically, without consuming your engineering bandwidth.

Decision Lens	**Building with Blocks**	**Hybrid**	**Conversational AI Products**
Time to Value	**Slow** – months to deploy, requires heavy integration	**Moderate** – faster than full build, slower than product	**Fastest** – days to deploy, vendor-managed
Continuous Monitoring & Improvement	**You own** – all upgrades, tuning, and compliance	**Split** – vendors improve models, you manage orchestration upkeep	**Automatic** – vendor delivers continuous improvements
Expertise & Specialisation	**High expertise** – deep in-house AI/engineering needed	**Moderate expertise** – required for integration and vendor mix	**Minimal expertise** – vendor encapsulates complexity

This option doesn't just minimise internal overhead, it future-proofs your deployment. You're tapping into best-in-class know-how that evolves daily, while your team stays focused on what truly differentiates your business: customer experience and integration with your systems. Strong SLAs ensure enterprise-grade reliability, while integration hooks give you enough flexibility without sacrificing speed or quality.

When to Go Hybrid

A balanced middle path for teams that want faster deployment than full pipeline builds but greater flexibility than a managed product. Useful when you want vendor frameworks to handle complex behaviours like turn-taking and VAD, while retaining freedom to choose TTS, STT, or LLM providers. This option gives you control where it matters most, without forcing you to reinvent low-level orchestration.

Conclusion

In closing, building voice-based conversational agents is a journey - but not one you need to start from scratch. With the right mix of internal innovation and trusted external expertise, enterprises can quickly launch solutions that delight customers, streamline operations, and explore new possibilities.

At ElevenLabs, we've seen success from teams who build full platforms to those who create powerful prototypes with just a few API calls. The key is knowing what to own and what to leverage.

Start small and learn by doing - whether it's a quick voice bot prototype or exploring vendor solutions. Voice technology has matured to the point where even a quick demo can feel magical – and that magic, backed by a solid strategy, can become a very real competitive advantage for those who seize it today.

Here's to making your enterprise heard in the age of conversational AI!