OpenAI

Voice Solutions 201

Accelerator

July 2025

How to use this Accelerator

This accelerator offers a robust starting point for anyone looking to quickly ramp up to an intermediate (201-level) understanding of the tools, methodologies, and outcomes involved. You'll gain practical insights from proven reference architectures. To help you take the next step, we've curated a selection of additional resources tailored to deepen your expertise.

- ✓ Showcasing the options you have to build this solution
- ✓ Giving you a high-level approach to know where to start
- ✓ Showing you how to approach next steps

Overview

Voice Solutions enable users to interact with LLMs through speech, building upon the voice interface API capabilities such as converting text into natural-sounding speech, transcribing spoken language into text, and direct speech-to-speech modality for real-time spoken language interactions.

Example use cases

- Customer Service Agent
- Multimodal User Interface

What we'll cover

- Value Proposition
- Building:
  - Process flow
  - High-level architecture
  - Features & limitations
  - Key challenges & pitfalls
  - Solution Deep Dive

Value Proposition

Value Proposition

Personalization at Scale

LLM-powered voice solutions can adapt based on context, user preferences, tone, and prior interactions. This enables emotionally aware, context-sensitive responses that improve customer satisfaction, build brand loyalty, and offer differentiated service.

Operational Efficiency

Voice Solutions can handle routine inquiries and high-volume interactions autonomously, freeing up human teams to focus on complex or high-value tasks. Organizations can optimize staffing levels while maintaining or improving service quality.

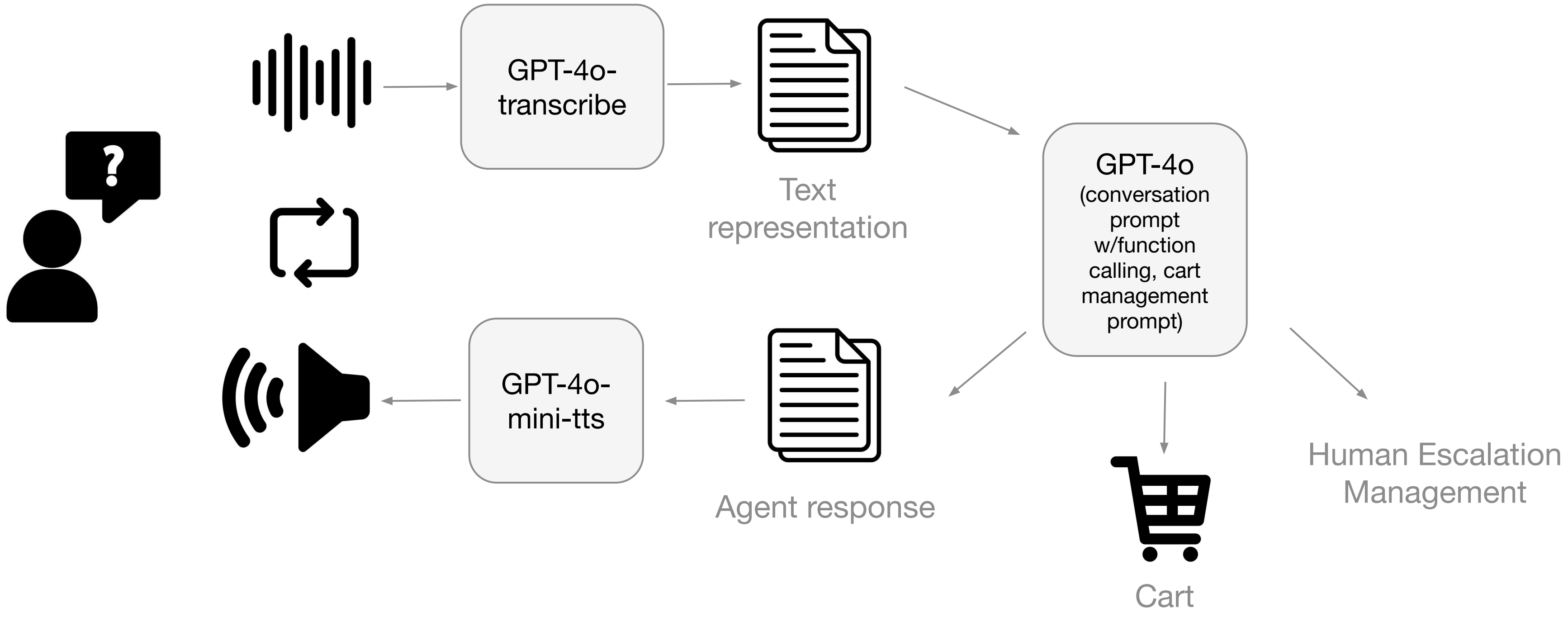
Seamless Interactions

Voice interfaces reduce the cognitive and physical effort required to interact with systems, increasing accessibility. Users can speak instead of typing or navigating menus—ideal for accessibility, hands-free environments (e.g., driving), and real-time interaction scenarios.

Building

Process flow

High Level Architecture - Voice Agents (Chained Approach)


GPT-4o-transcribe
Text representation
GPT-4o (conversation prompt w/function calling, cart management prompt)
Agent response
GPT-4o-mini-tts
Cart
Human Escalation Management

Features and limitations (Chained Approach)

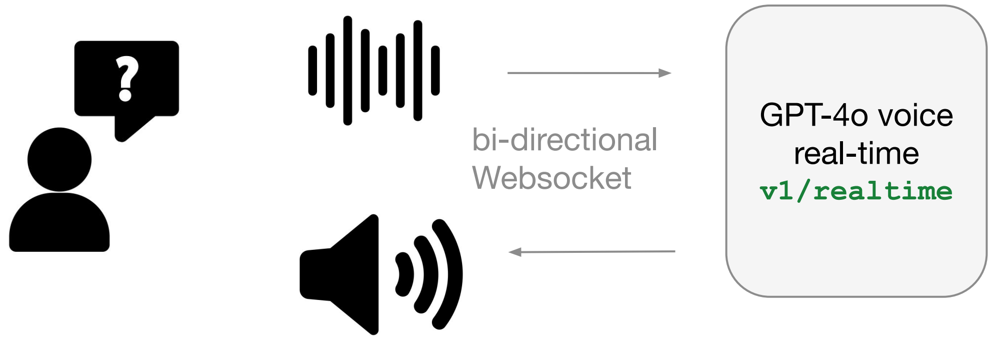
Good for ✅

- **High control & transparency:** All intermediate outputs are accessible (e.g., transcripts), making it easier to debug, audit, and fine-tune logic.
- **Structured interaction support:** Works well with workflows that require function calling, scripting, or logging.
- **Easy to repurpose existing LLM apps:** Especially valuable if you're starting with a text-based app or API flow.
- **Easier to run evals on**: Store and anonymizing transcripts and generating synthetic data is faster and cheaper.

Not good for ❌

- **Higher latency:** Due to the sequential processing of transcribe → generate → synthesize.
- **Loss of vocal nuance:** Emotions, tone, or hesitations in user speech are flattened during transcription.

High Level Architecture: Realtime Voice Approach



Low-latency interaction with streaming audio input and output

Supports function calling (similar to Chat Completions API)

Features and limitations (Realtime Voice)

Good for ✅

- **Low-latency, fluid conversations:** Designed for natural turn-taking and responsive dialogue.
- **Multimodal understanding:** The model "hears" tone, inflection, and nuance, making interactions feel more human.
- **No need for transcripts:** Reduces complexity if logs and auditability aren't top priorities.

Not good for ❌

- **Less predictable outputs:** May be harder to log, validate, or review individual stages.
- **Fewer guardrails and lower modularity:** It's more difficult to insert logic or hand off control between steps.
- **Cost:** Realtime API is more expensive.

Key Challenges & Pitfalls

CHALLENGES

- Determining which to go with: Realtime or chained approach.
- Evaluating performance can be harder with voice, as automated evals are harder to implement.
- Maintaining low latency with high quality. This may have tradeoffs that need to be managed.

PITFALLS & MITIGATIONS

- **Unclear Goals**
  - Ensure that clear evaluations are established ahead of time.
- **Architecture Selection**
  - Ensure that requirements are clearly defined from the onset to make sure the right method is chosen.

Solution Deep Dive

OpenAI API Offerings

Text-to-speech

Transforming written text into natural-sounding speech, enabling applications like audiobooks, virtual assistants, and announcement systems.

- GPT-4o-mini-tts

Speech-to-text

Converting spoken language into written text, facilitating functionalities such as voice commands, transcriptions, and voice-controlled data entry.

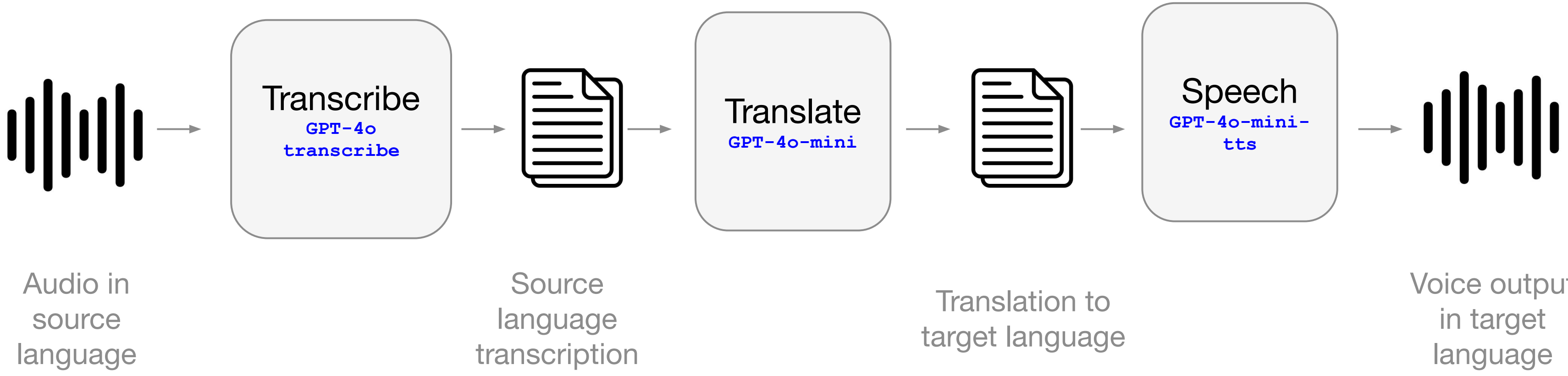
- GPT-4o-mini-transcribe
- GPT-4o-transcribe

Speech-to-speech

Directly translating spoken input into synthesized speech output, allowing for real-time language translation, voice cloning and customer support applications.

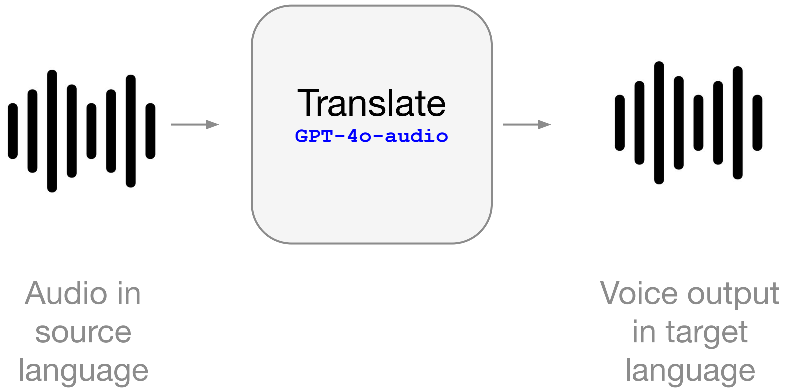
- GPT-4o-realtime
- GPT-4o-mini-realtime

Voice Translation of an Audio (chained)



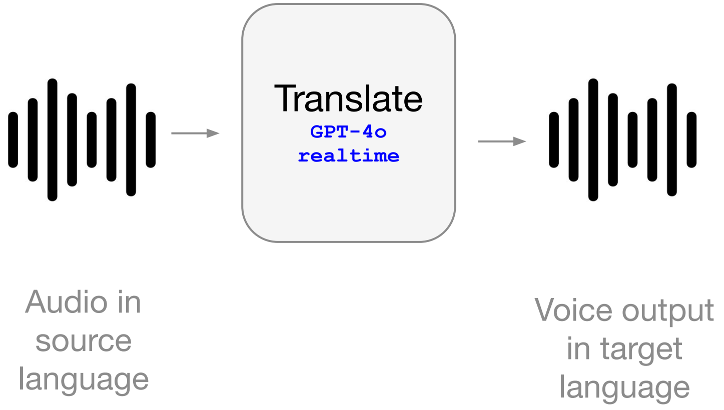
- High control over each stage
- Easy to audit and debug transcript
- Higher Latency
- Transcription step means you lose accent, tone

Voice Translation of an Audio (GPT-4o audio)



- Captures inflection and tone by translating raw audio
- More expensive than chained with tts & stt
- If you want to evaluate the transcripts, you need to transcribe the input language audio to compare with the output translated transcript
- GPT-4o audio must wait for the full audio input to be uploaded and processed

Voice Translation of live Audio (with Realtime API)



- Streaming, lowest latency
    - Still turn-based, model cannot simultaneously listen and translate
- More expensive than chained with tts & stt or GPT-4o audio
- More complex architecture, requires websockets or WebRTC

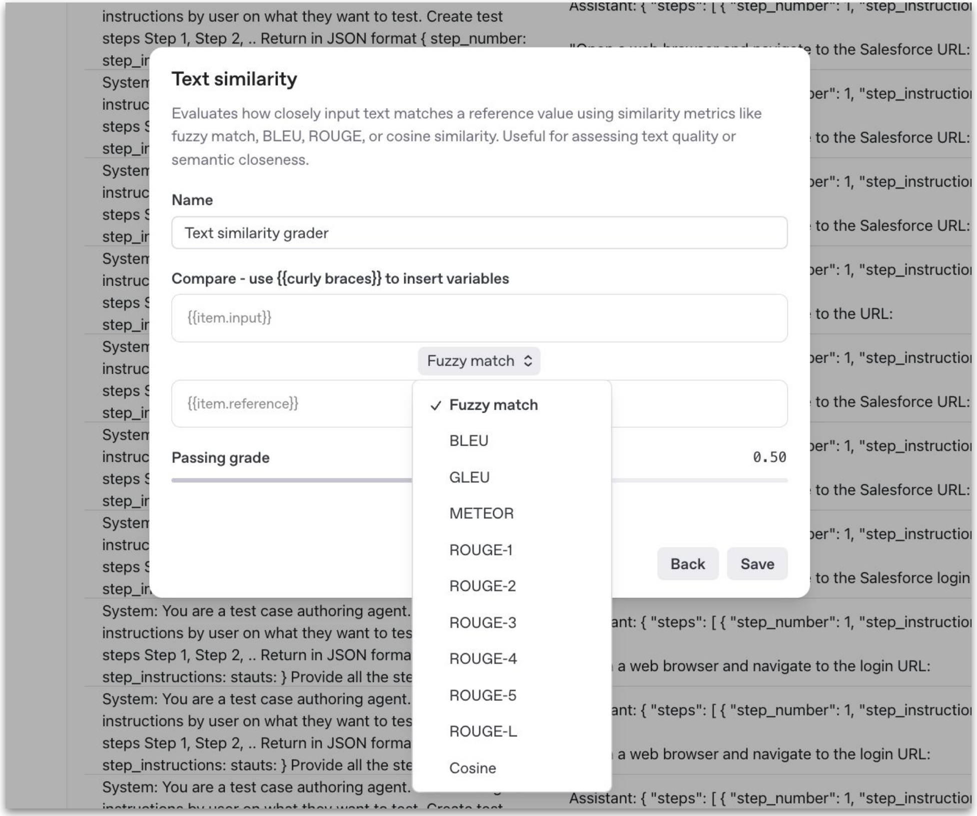
Evaluating Translations: Standard Evals

Standard Evals like BLEU and ROUGE gauge quality based on token overlap (not semantic)

- BLEU measures precision of n-grams
- ROUGE measures overlap of n-grams and the longest common subsequence.
  - An n-gram is simply a grouping of tokens/words.
- METEOR is a hybrid eval: it still uses n-gram matching but adds semantic cues like synonymy and paraphrasing.
- *BLEU is better when precision matters, ROUGE when recall matters and summaries, METEOR when both matter.*

These metrics are deterministic, can be a canary and are used for academic benchmarks

But these don't capture semantic similarity, just syntactical.



Text similarity

Evaluates how closely input text matches a reference value using similarity metrics like fuzzy match, BLEU, ROUGE, or cosine similarity. Useful for assessing text quality or semantic closeness.

Name

Text similarity grader

Compare - use {{curly braces}} to insert variables

{{item.input}}

Fuzzy match

{{item.reference}}

✓ Fuzzy match
BLEU
GLEU
METEOR
ROUGE-1
ROUGE-2
ROUGE-3
ROUGE-4
ROUGE-5
ROUGE-L
Cosine

Passing grade 0.50

Back Save

Evaluating Translations: Semantic

Semantic evals

- **Cosine** similarity measures semantic similarity

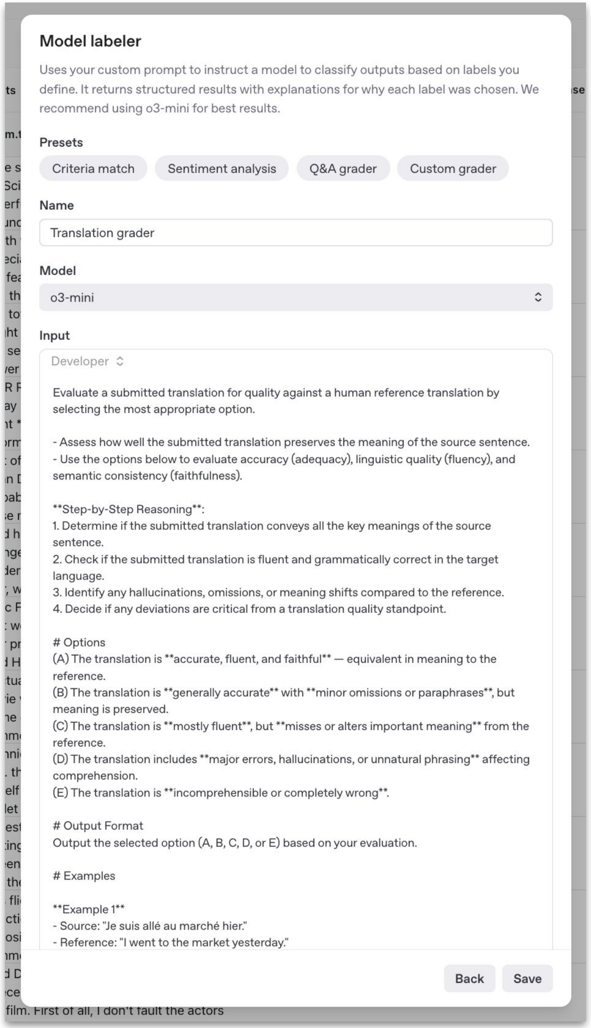
LLM-as- judge evals

- Prompt GPT-4o-mini with key evals.

Note: our eval platform only handles text, so you need to compare translations via transcript (not raw audio).

Evaluating audio

- You can pass audio samples to GPT-4o audio and prompt for tone, inflection, speech rate, etc



Model labeler

Uses your custom prompt to instruct a model to classify outputs based on labels you define. It returns structured results with explanations for why each label was chosen. We recommend using o3-mini for best results.

Name: Translation grader

Model: o3-mini

Input

Developer

Evaluate a submitted translation for quality against a human reference translation by selecting the most appropriate option.

- Assess how well the submitted translation preserves the meaning of the source sentence.
- Use the options below to evaluate accuracy (adequacy), linguistic quality (fluency), and semantic consistency (faithfulness).

Step-by-Step Reasoning:
1. Determine if the submitted translation conveys all the key meanings of the source sentence.
2. Check if the submitted translation is fluent and grammatically correct in the target language.
3. Identify any hallucinations, omissions, or meaning shifts compared to the reference.
4. Decide if any deviations are critical from a translation quality standpoint.

Options
(A) The translation is **accurate, fluent, and faithful** — equivalent in meaning to the reference.
(B) The translation is **generally accurate** with **minor omissions or paraphrases**, but meaning is preserved.
(C) The translation is **mostly fluent**, but **misses or alters important meaning** from the reference.
(D) The translation includes **major errors, hallucinations, or unnatural phrasing** affecting comprehension.
(E) The translation is **incomprehensible or completely wrong**.

Output Format
Output the selected option (A, B, C, D, or E) based on your evaluation.

Examples

Example 1
- Source: "Je suis allé au marché hier."
- Reference: "I went to the market yesterday."

Evaluating Translations: Example

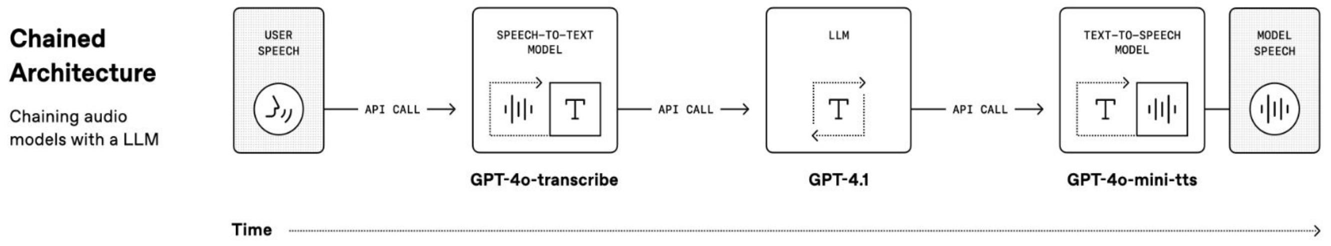
Reference: ***“He is eating.”***
Candidate: ***“He is having dinner.”***

Metric	Score / Grade	Notes
BLEU	~0.41	Penalized for paraphrasing
ROUGE-1/L	~0.67	Partial match
METEOR	~0.65–0.7	Recognizes paraphrase
Cosine	~0.92	Semantically strong
LLM-as-Judge	B (4/5 adequacy)	Fluent, semantically sound, slight meaning shift

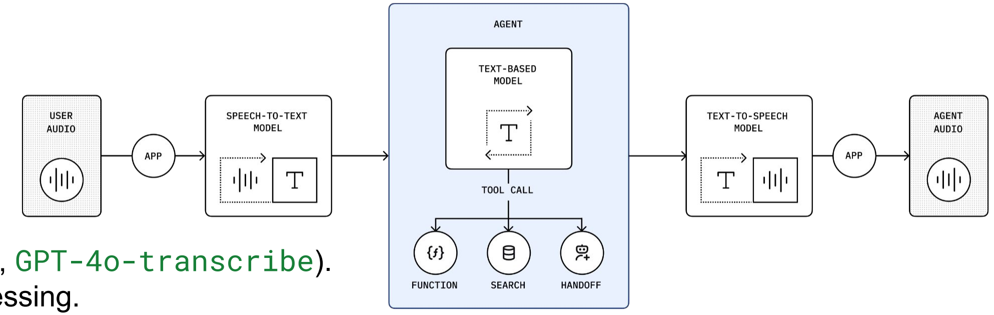
Voice Chatbot - Chained Architecture

Chained Voice Agent Architecture

- **Sequential Audio Processing:**
  - Predictable, easy to debug
  - Full transcript access for both user input and model output
- **Control & Flexibility:**
  - Allows post-processing, safety checks, and application logic
  - Seamlessly converts text-based LLM apps into voice agents
- **Model Chain Example:**
  - **GPT-4o-transcribe** (ASR) → **GPT-4.1** (LLM) → **GPT-4o-mini-tts** (TTS)



Detailed Flow



1. **User Audio Input**
   - Captured via application front-end
   - Passed to the STT module for transcription.

2. **Speech-to-Text Model (ASR)**
   - Converts incoming audio into text using ASR (e.g., `GPT-4o-transcribe`).
   - Ensures text representation for downstream processing.

3. **Text-Based Model (LLM Agent)**
   - Receives transcribed text and processes it using an LLM (e.g., `GPT-4.1`).
   - **Tool Use:**
     - Supports agentic actions:
       - **Function Calls:** Integrates external APIs or logic.
       - **Search:** Retrieves real-time or database information.
       - **Handoff:** Passes control to human or alternate agent.
   - Produces a contextual, intelligent response in text form.

4. **Text-to-Speech Model (TTS)**
   - Synthesizes the agent's text response into natural-sounding audio (e.g., `GPT-4o-mini-tts`).
   - Enables conversational, voice-based user experience.

5. **Agent Audio Output**
   - Audio response delivered back to the user via the application.

Realtime Speech-to-Speech

Realtime

Direct Audio-to-Audio Pipeline:

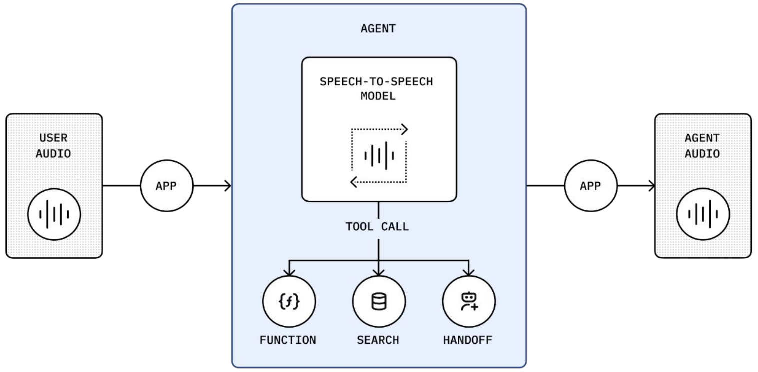
- User audio is streamed directly into a multimodal agent (no intermediate transcript).
- The S2S agent model processes, interprets, and generates speech output in real time.

Multimodal Model Capabilities:

- Ingests and reasons over both audio and text features natively.
- Captures **prosody, inflection, and speaker intent** directly from speech input.
- Filters noise, handles interruptions, and maintains conversational context natively.

Tool Call Integration:

- Supports live function execution, retrieval, and handoff scenarios—all triggered within a single audio session.



Service Mechanics - The Basics

- Stateful API - clients connect to wss://api.openai.com/v1/realtime via **websockets** and **push/receive JSON formatted messages** throughout the duration of a session

- Session - A single conversation between a client and the server

- Triggering Server Responses: Two modes:
  - **server_detection** the server will run voice activity detection (VAD) over the incoming audio and respond after the end of speech
  - **client_decision** the client sends an explicit message that it would like a response from the server. This mode may be appropriate for a push-to-talk interface or if the client is running its own VAD.

- Interruptions - model can be interrupted, halting model inference but retaining the truncated response in the conversation history.

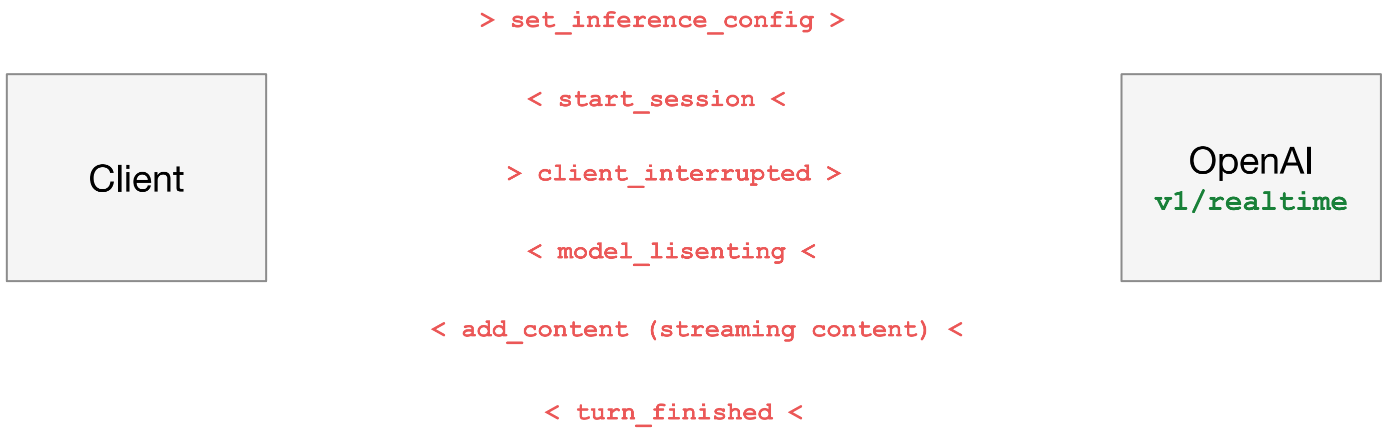
- Tool calls - the client can expose tools to the server in a **set_inference_config** message and the server will respond with tool call messages, if appropriate

- Messages - The API communicates using event based messages over the websocket connection.

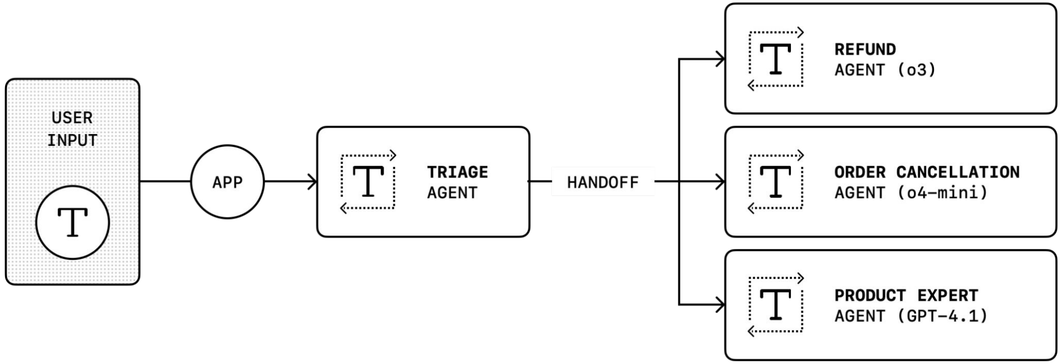
Messages

The realtime API takes advantage of bi-directional event based communication over the websocket to accept and produce messages asynchronously as the session progresses.

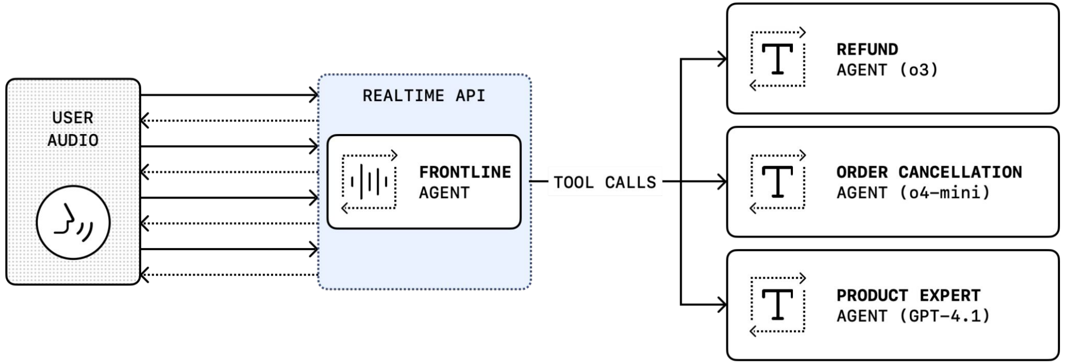
A typical flow may look like this:



Text-based agents: Delegation / Handoff


USER INPUT
APP
TRIAGE AGENT
HANDOFF
REFUND AGENT (o3)
ORDER CANCELLATION AGENT (o4-mini)
PRODUCT EXPERT AGENT (GPT-4.1)

Speech-to-speech agents: Delegation through tools


USER AUDIO
REALTIME API
FRONTLINE AGENT
TOOL CALLS
REFUND AGENT (o3)
ORDER CANCELLATION AGENT (o4-mini)
PRODUCT EXPERT AGENT (GPT-4.1)

Best Practices

Design for Low Latency Across the Stack

Voice experiences demand responsiveness — users expect interactions to feel real-time. Use low-latency transport protocols like WebRTC for client-facing agents and WebSocket for server-based integrations. Prioritize geographic routing, caching, and optimal model selection to minimize processing delays, and continuously monitor roundtrip latency during live sessions.

Focus the Agent’s Task Scope

Voice agents perform best when they are purpose-built for narrow tasks. Avoid overloading them with too many tools or intents. Instead, define a tight goal, provide fallback mechanisms (e.g., escalation to a human), and embed critical information directly in the system prompt to optimize on tool calls. This enables smoother, more reliable voice flows.

Adapt the Voice to the User and Context

The tone, verbosity, and pacing of voice responses should shift depending on task type and audience. A tutoring session may benefit from slower, supportive language, while a voicebot handling appointment bookings should be concise and direct. Use meta-prompting to encode stylistic variation, and consider real-time voice modulation to tailor pacing per session.

Align Voice Architecture w/ Latency Needs

Pick your stack based on how quickly the user must hear a response: choose the Real-Time Voice pipeline for sub-second, conversational turn-taking, and switch to a chained workflow for longer, prescriptive tasks where modest latency trades off for flexibility and lower cost.

Robust Evaluation and Feedback Loop

Instrument the full pipeline — from audio input to speech synthesis — to trace performance issues and optimize continuously. Use eval frameworks to track metrics like turn latency, interruption handling, hallucination rates, and user satisfaction (e.g., NPS). Build feedback loops with real usage data and test iteratively in staging before scaling up.

Key Takeaways

- **Solution Overview:** Voice Solutions enable users to interact with LLMs through speech

- **Use case:**
  - Voice based sales and service solutions
  - Voice assistants for user support
  - Audio narration such as news, podcasts, and audio books
  - Audio transcription such as meetings and customer interactions
  - Tutoring for language and other skills

Resources

API Documentation

- Realtime API
- Best Practices
- Voice Translation Cookbook
- One way Translation With Realtime
- Realtime Agents Walkthrough (github)
- Text to Speech TTS-1 model
- Speech to text Whisper Model

- Agents SDK voice agents quickstart
- https://github.com/openai/openai-realtime-agents
- https://github.com/openai/openai-realtime-solar-system
- https://www.openai.fm/