Voice Interaction System Implementation Description

Gabriel Manso
MIT EECS
gmanso@mit.edu

October 2025

1 Implementation Overview

The voice interaction system integrates several libraries to handle speech-to-text, natural language processing, and text-to-speech. For speech recognition, I used OpenAI's Whisper model with the "small" variant, which balances accuracy and speed for conversational speech. Audio capture uses the `sounddevice` library to record six seconds of audio at 16kHz mono, saving temporarily as WAV files via `soundfile` before passing to Whisper for transcription.

The text-to-speech component uses Kokoro TTS exclusively as the primary engine, configured with the "am_puck" voice at 24kHz. Kokoro produces significantly more natural-sounding speech compared to system TTS alternatives. The system includes a voice mapping mechanism that maps various voice names (like "alloy", "echo", "fable", etc.) to the preferred "am_puck" voice for consistency.

The AI layer is built on CrewAI, a multi-agent framework where I used an adapted version of the agents used in HW1 & HW2. Additionally, the system creates dynamic Gabriel Response Agents on-demand for general queries that don't fit the specialized categories. All agents use GPT-4o-mini with Ollama as a fallback and maintain a consistent persona across interactions.

The complete pipeline flows from recording through transcription, intent classification, agent processing, response generation, speech synthesis, and playback. Each stage has comprehensive error handling since components can fail independently—microphones might be unavailable, transcription can fail on noisy audio, API calls can timeout, or playback devices might have issues. When voice components fail, the system falls back to text-only mode with manual transcript correction capabilities.

2 Example Run and Analysis

Consider a user speaking: "Hey Gabriel, can you introduce yourself?" The system captures the audio, and Whisper transcribes it accurately. The intent classifier detects the "introduction" keyword and routes to the Personal Assistant agent rather than the general response agent. Kokoro TTS then synthesizes this text into speech, which takes about four to five seconds. The complete pipeline from the end of user speech to system response takes eight to twelve seconds. After playback, temporary audio files are cleaned up automatically.

Example from the video:

User: Can you please introduce yourself?

Gabriel: Hey there! So nice to meet you! My name is Gabriel Manso, and I'm a PhD student at MIT EECS. I'm originally from Brazil, but I've been living in Cambridge for a few years now, and I absolutely love it here. When I'm not buried in research papers or coding away on my laptop, you can find me jamming out to some Pink Floyd or Bombay Bicycle Club—I'm a huge music lover! My friends would describe me as warm, empathetic, and always up for a good conversation. I'm passionate about systems thinking, rapid synthesis, and hypothesis-driven validation—basically, anything that helps us understand the world better. And, of course, I care deeply about privacy and ethics in tech. When I'm not geeking out over code or research, you can find me trying to squeeze in a workout (I love yoga!) or exploring the Boston area with friends. So, what brings you here today?

Several observations emerged from testing. Whisper's "small" model achieves about 95% accuracy on conversational speech, with most errors on proper nouns or technical terms that the language model can usually infer from context. Kokoro TTS sounds noticeably more natural than system TTS alternatives, with the "am_puck" voice providing consistent personality and emotional expression. The multi-agent architecture maintains consistent personality across different interaction types, with each agent specialized for specific domains while maintaining Gabriel's conversational style.

The eight- to twelve-second latency is acceptable but approaches the limit of what feels conversational. Visual feedback helps manage expectations during processing. Error recovery proved important—audio device conflicts are common, so having text input fallback and manual transcript correction prevents complete interaction failures. The system gracefully handles various failure modes including audio recording failures, STT errors, API timeouts, and TTS synthesis problems.

3 Technical Architecture and Implementation Details

The system follows a modular architecture with clear separation of concerns. The *VoiceInteractionSession* class orchestrates the entire pipeline, managing *AudioInterface* for recording and playback, *SpeechToTextEngine* for Whisper-based transcription, *TextToSpeechEngine* for Kokoro TTS synthesis, and *GabrielCrewAI* for AI processing.

The audio interface uses *sounddevice* for real-time audio capture and playback, with configurable sample rates (16kHz for recording, 24kHz for TTS output). The system automatically manages temporary audio files in a dedicated *voice_artifacts/* directory, with automatic cleanup after processing.

The CrewAI implementation includes the persona management through a centralized *PERSONA* configuration that defines Gabriel's characteristics, values, and preferences. This ensures consistency across all agents while allowing for specialized behavior in different domains. The system uses current date context in all interactions, ensuring responses are timely and relevant.

Error handling is comprehensive, with graceful degradation at each stage. If audio recording fails, users can switch to text input. If transcription fails, manual correction is available. If TTS fails, text responses are still provided. The system includes health checks and diagnostic tools for troubleshooting.