Voice Interaction Implementation

The voice-enabled meeting agent extends the existing CrewAI summarizer by adding a lightweight audio layer for speech input and spoken output. The underlying logic and summarization template remain unchanged. The new code handles microphone recording, transcription, and text-to-speech, then routes the resulting text through the same CrewAI summarizer used in previous assignments.

For speech-to-text, the system uses the open-source *OpenAI Whisper* model. Audio is captured through the *sounddevice* library, which records from the system's default microphone into *NumPy* arrays. The *soundfile* package writes temporary WAV files at 16 kHz, single-channel audio. *PortAudio* must be installed at the operating system level for microphone access. Whisper is called via *model.transcribe()* and automatically normalizes audio using *ffmpeg*. On machines without GPU acceleration, it runs in FP32 precision and issues a non-fatal CPU warning. Users can select the model size through a command-line flag such as *--stt-model base*.

Text-to-speech output is generated with the *pyttsx3* library, which synthesizes the agent's text summary into a WAV file. The same interface can play the audio immediately or save it for later. Command-line arguments control the output voice and speech rate. Together, the *STT* and *TTS* functions form a loop that allows the user to speak, listen, and iterate through turns in real time.

The program is launched with *python -m cli --mode meeting --voice*. When the session begins, the user presses Enter to start recording, speaks their meeting notes, and presses Enter again to stop. The audio is transcribed by *Whisper*, passed to the CrewAI meeting agent, and summarized into four sections: TLDR, Decisions, Risks or Blockers, and Next Steps. The summary is printed to the terminal and then converted to speech through *pyttsx3*. The loop repeats until the user says "exit" or interrupts the process. Additional flags enable or disable playback, preserve intermediate files, or run one-off transcriptions of prerecorded audio.

The agent stack remains identical to earlier assignments. It uses *Claude 3.5 Sonnet* through CrewAI as the underlying model. The summarization template and output format are unchanged, ensuring the same clarity and structure in both text and voice modes. Environment configuration includes the *Anthropic API* key*, ffmpeg,* and *PortAudio*. The Python dependencies are listed in a requirements file for easy setup.

When Whisper introduces minor transcription errors, such as mishearing names or technical terms, the system continues to function because the summarizer relies on sentence structure rather than individual tokens. If an essential word is misheard, the output may omit or generalize that item. In most cases, the meeting template helps contain such effects by leaving missing information marked as "not found" rather than fabricating details. A possible improvement would be to add lightweight correction logic that prompts the user when uncertain words appear in the transcript.

Example Run

The system starts with the command *python -m cli --mode meeting --voice*. The terminal displays a message indicating that the voice session is ready. After pressing Enter, the user speaks for about fifteen seconds describing progress in a team stand-up. Whisper transcribes the speech accurately, including domain-specific terms such as “flaky tests” and “regression run.” The summarizer produces a concise four-part report listing completed work, detected test issues, and next steps for Friday. The summary is printed and then spoken aloud through pyttsx3.

In the second turn, the user adds new tasks and owners: assigning Mia to fix the flaky tests by Thursday, committing to a regression run Friday at noon, and requesting Ops to update dashboards before rollout. The transcription captures all names and dates correctly. The summarizer restates these as clear decisions and structured next steps. The TTS module converts the output to speech and plays it immediately.

In the third and final turn, the user says “Exit.” The loop terminates cleanly, deletes temporary files unless preservation flags are set, and prints an exit note.

Across all turns, latency remains a few seconds, consisting of recording time plus STT, model response, and TTS synthesis. Accuracy of transcription is high enough for consistent summaries, and small phrase differences do not change the meaning. The structured output stays uniform, leaving missing data marked instead of inferred. Overall, the system demonstrates a complete multimodal meeting agent that listens, summarizes, and speaks with reasonable responsiveness and stability.