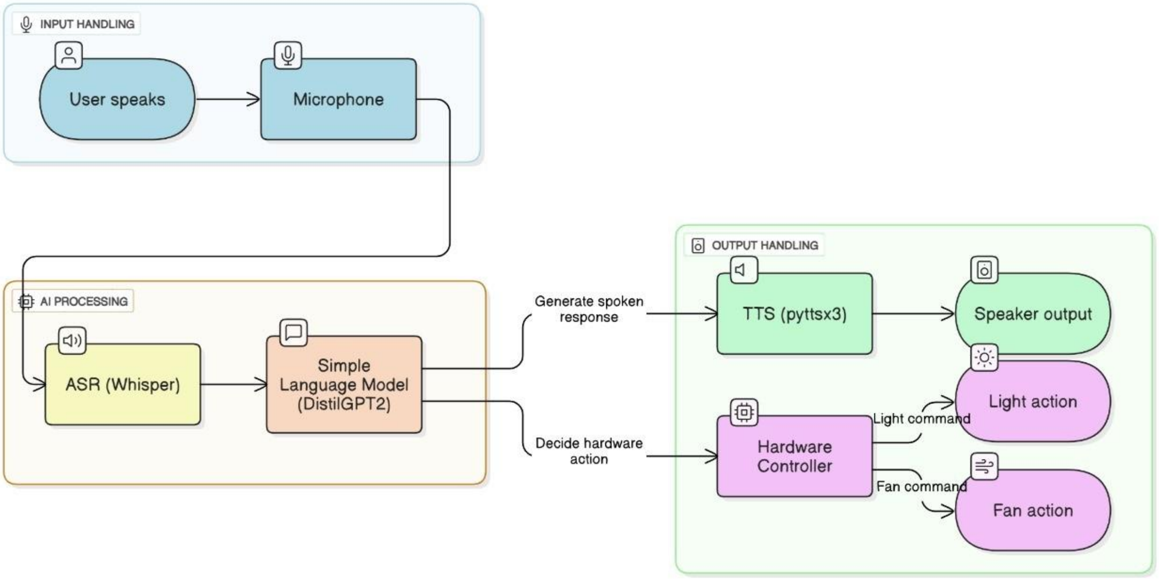
Research & Design Proposal

Local AI Voice Assistant

1) System Architecture



Explanation: -

- User Voice: The user speaks commands like “Turn on the light.”
- ASR (Whisper): Converts recorded speech to text.
- SLM (DistilGPT2): Processes text command, generates response text, and decides the hardware action.
- TTS (pyttsx3): Converts response text to audio for playback.
- Speaker Output: Plays the response audio.
- Hardware Controller: Executes the action (light/fan) on GPIO pins; in your implementation, this is simulated.

2) Model & Library Choices

Stage	Model / Library	Notes / Rationale
ASR	Whisper (small)	Accurate, offline transcription, lightweight enough for local CPU.
SLM	DistilGPT2	Small, fast, suitable for command interpretation and response generation.
TTS	pyttsx3	Lightweight, offline TTS; integrates easily with Python.
Hardware	Python simulation	HardwareController simulates GPIO actions; can extend to Raspberry Pi / Arduino.
Audio	Sound device, simple audio	Used for microphone capture and playback in local environment.

3) Constraints & Trade-offs

Constraint	Discussion
Model Size	Whisper-small (~74 MB) and DistilGPT2 (~332 MB) fit local CPU with limited RAM.
Latency	ASR + SLM + TTS response ~1–3 seconds; suitable for simple home commands.
RAM Usage	Models run on 8–16 GB CPU machines; small models prevent memory overload.
Power Needs	Entire system can run on a laptop; optional Raspberry Pi/Arduino can extend to low-power hardware.
Accuracy vs Speed	Whisper-small is accurate for simple commands; DistilGPT2 may require careful prompt design to avoid repetitive outputs.
Offline Mode	All components work fully offline after downloading models to cache.

4) Example Use Cases

(a) Voice-Controlled Light / Fan Simulation

- Commands like “Turn on light” or “Turn off fan” are processed.
- HardwareController simulates the action with logs and optional GPIO control.
- Response is played via TTS, e.g., “Turning on the light.”

(b) Local Command-Line Voice Assistant

- Runs entirely offline.
- Users speak commands; assistant responds via audio and logs actions.

- Demonstrates end-to-end AI pipeline: ASR → SLM → TTS → Action.

(c) Smart Home Prototype

- Extendable to Raspberry Pi GPIO for controlling lights, fans, or other devices.
- Audio logs maintained for debugging and analytics.

5) Notes & Recommendations

- **Model Cache**: Pre-download Whisper & DistilGPT2 into ~/.cache/huggingface for offline use.
- **Logging**: Logger class stores speech, hardware actions, and errors for traceability.
- **Prompt Design**: Keep SLM prompts structured to avoid repetition.
- **Hardware Extension**: Later add real GPIO via Raspberry Pi or Arduino using pySerial.
- **TTS Quality**: pyttsx3 is lightweight but lower quality; Coqui/Silero can replace if higher fidelity is needed.