Voice Assistant with Local LLM and Audio I/O

Overview

This project implements a fully offline **voice-based assistant system** that performs speech recognition, natural language understanding, and text-to-speech synthesis. It uses:

- **OpenAI Whisper** for transcription
- **TinyLlama (1.1B GGUF)** for local LLM-based responses
- **Custom TTS module** for voice synthesis
- **CTransformers** for lightweight, CPU-friendly model execution

All components run locally, making this assistant private, fast, and fully internet-independent after initial setup.

🔧 Features

- 🎙️ **Voice-to-Text** via OpenAI Whisper
- 🧠 **Local Chatbot Intelligence** via TinyLlama 1.1B
- 🗣️ **Offline Text-to-Speech (TTS)** playback
- 🔁 **Voice Conversation Loop**: Record → Transcribe → Respond → Speak
- ⚡ **Low-latency execution** with CTransformers
- 🧠 **Built-in factual lookup** for predefined knowledge queries
- 🧱 **Modular design** for easy customization and future upgrades
- 🚫 **No internet required** during operation

📁 Project Structure

```
voice-assistant/
├── app/
│   ├── audio_processing/
│   │   ├── record_audio.py      # Microphone input recording
│   │   └── play_audio.py        # Audio playback system
│   ├── models/
│   │   ├── chatbot.py           # Chat logic & LLM fallback
│   │   ├── tts.py               # Text-to-speech synthesis
```

```
|   |   └── whisper_model.py      # Speech-to-text (Whisper)
|   └── utils/
|       └── config.py           # Global config variables
├── local_llm.py                # Loads and runs TinyLlama via CTransformers
├── main.py                     # Runs the complete voice interaction loop
└── README.md
```

📦 Installation & Setup

1. Install Dependencies

pip install -r requirements.txt

Also install ffmpeg for audio processing.

2. Download Models

- 🔊 **Whisper Model**: Handled via the openai-whisper package
- 🤖 **TinyLlama (GGUF)**:
  Download from [HuggingFace: TinyLlama-1.1B-Chat-v1.0-GGUF] and place inside models/

🎙️ Audio Workflow

Step 1: Record

record_audio(path, duration, sample_rate)

Captures audio from mic and saves as a WAV file.

Step 2: Transcribe

WhisperModel.transcribe(path)

Uses Whisper to convert audio to text.

Step 3: Respond + Speak

TTS.synthesize(text, filename)

play_audio(filename)

Generates voice from text and plays it.

🧠 Chatbot Intelligence

1. Fact Lookup

Handles known queries locally, such as:

"Who was the Prime Minister of India in 2014?"

2. LLM Responses

Fallback to TinyLlama if no match is found:

response = llm.generate_response(prompt)

- Enhances prompt with current date
- Stateless model; no history/context yet

⚙️ **Configuration (config.py)**

AUDIO_DIR = "path/to/audio"

DEFAULT_DURATION = 5

SAMPLE_RATE = 16000

WHISPER_MODEL_SIZE = "base"

🔄 **Main Interaction Loop**

assistant = ConversationManager()

assistant.start_conversation()

- Records audio input
- Transcribes and analyzes query
- Responds using fact logic or TinyLlama
- Converts response to voice and plays it

🧯 **Error Handling**

Situation	Behavior
Silence / noise	Asks to re-record
Exceptions	Logged and bypassed
KeyboardInterrupt	Gracefully exits

🚧 **Known Limitations**

- No memory between queries
- No real-time information access

- May generate verbose or hallucinated outputs
- Only English is supported currently
- CPU-only execution may be slow (no GPU fallback yet)

🌟 Future Improvements

- Add **long-term conversation memory**
- Support **multi-language** queries
- Use **Faster-Whisper** for quicker transcription
- Improve TTS voice quality using **neural models**
- Add **desktop GUI** or **mobile interface**

🙏 Credits

- OpenAI Whisper – Speech-to-text
- TinyLlama (GGUF) – Offline LLM
- CTransformers – Lightweight LLM framework

👨‍💻 Author

Dhruv Khatter
(Offline AI, Speech Interfaces, Embedded NLP Systems)