Week 3 - Assignment: Voice Agent Development

From now on, we start to hands on buiding Research Voice Agent, truly useful AI Research Assistants must listen, understand, and respond with voice. **we will give you some simple introduction code as a starter, feel free to write your own code or do optimization.**

📚 Learning Objectives this week

to build a simple Voice Agent, we need these following knowledge.

* **1. Speech Recognition (ASR):** Convert audio to text using models like Whisper or Google Speech-to-Text.
* **2. Dialogue Generation with LLMs:** Feed transcribed user input into LLM (e.g. LLaMA 3) and generate natural language responses.
* **3. Text-to-Speech (TTS):** Use a TTS engine (CozyVoice) to convert generated responses into spoken audio.
* **4. FastAPI for API Serving:** Create a web server with FastAPI to handle audio file uploads and return voice responses.
* **5. Conversation State Management:** Track conversation history to enable multi-turn interaction.
* **6. Low-Latency Real-Time Processing:** Use asynchronous functions to reduce inference time and improve response experience.

✅ You do NOT need Docker. Just ensure your local Python environment works.

🧪 Project: Build an Local Voice Assistant

🎯 Goal:

Develop a real-time voice chatbot that can:

1. Take audio input via HTTP,
2. Transcribe audio to text (ASR),
3. Generate a response using LLM,
4. Convert the response back to speech (TTS),
5. Support 5-turn conversational memory.

Step 1: FastAPI Skeleton

Create a simple FastAPI server that accepts an audio file via POST and returns an audio file in response:

here is the official guidance of FastAPI [fastapi](https://fastapi.tiangolo.com/)

from fastapi import FastAPI, UploadFile, File

from fastapi.responses import FileResponse

app = FastAPI()

@app.post("/chat/")

async def chat\_endpoint(file: UploadFile = File(...)):

audio\_bytes = await file.read()

*# TODO: ASR → LLM → TTS*

return FileResponse("response.wav", media\_type="audio/wav")

Run your server:

uvicorn main:app --reload

Test it with curl, Postman, or a custom frontend.

Curl:

A screenshot of a computer code

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Postman:

A screenshot of a computer

AI-generated content may be incorrect.

Frontend:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

Step 2: ASR (Speech Recognition)

Use OpenAI Whisper to transcribe the uploaded audio to text:

import whisper

asr\_model = whisper.load\_model("small")

def transcribe\_audio(audio\_bytes):

with open("temp.wav", "wb") as f:

f.write(audio\_bytes)

result = asr\_model.transcribe("temp.wav")

return result["text"]

Add it to the /chat/ route:

user\_text = transcribe\_audio(audio\_bytes)

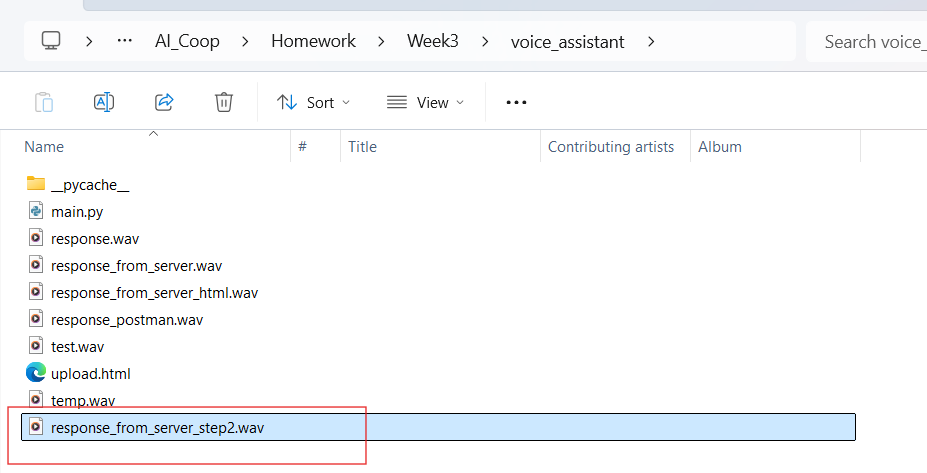
Print user\_text for debugging.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.



 Deliverables

* A runnable FastAPI project with /chat/ endpoint
* A working voice assistant that handles **5-turn** multi-round conversations
* Code with clear structure and modular components (ASR, LLM, TTS)
* A **2-minute screen recording** demo: record 5 turns of real-time interaction
* Optional: Add conversation memory display, prompt formatting logic, async optimization

🌟 Extension Ideas (Optional)

* Use async processing for parallel ASR/LLM/TTS.
* Integrate a microphone frontend UI for live recording.
* Add speaker identification or personalized voice response.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.