# **Building My AI Interview Bot - Design Approach & Lessons Learned**

## **Setup & Getting Started**

So my plan was simple: get the core tools up and running first, make sure everything works, and then integrate it with an AI agent. No point in dealing with orchestration if the basic functionality is broken.

For joining Zoom meetings, I just used a meeting URL for now. Later, it’ll make sense to hook into the **Zoom SDK** so I can send messages in the meeting and maybe even schedule interviews by parsing chat messages.

I have also attached the console output of the agent, since couldn't record loom video, the output clear shows the transcription and decisions agent made

## **Speech-to-Text (STT) Pipeline**

At first, I thought about using **Assembly AI** for STT, but after burning through the free credits, I realized it’s not practical for long-term use. So I switched to **Whisper.cpp** and ran it locally. It’s:

* **Faster** than cloud-based STT
* **Free** (after setup)
* **Accurate enough** for the job

### **The Nightmare of Audio Routing**

Getting Zoom audio into the STT pipeline on macOS was a pain. I had to set up **BlackHole** (virtual audio cable) to route sound from Zoom to my model. Messing with multiple audio devices on a Mac is always a headache, but I got it working.

### **Detecting When to Stop Recording**

Didn’t want to manually control recording, so I used **Voice Activity Detection (VAD)**:

* Measured **RMS (Root Mean Square)** values of the audio
* If the RMS dropped below **50** and stayed there for **2 seconds**, I assumed the candidate stopped talking
* Triggered STT processing at that point

This setup avoids unnecessary recording time and speeds up processing.

## **AI Model & Agent Orchestration**

For generating interview questions and handling responses, I needed a local model. I went with **Ollama** running **Llama 3.1 8B** instead of **Phi-3.5 Mini** because:

* It gave **more consistent outputs**
* Integrated better with the agent framework

I could have used **Gemma 3** or **Phi 4**, but Llama 3.1 was already installed and working well, so why complicate things?

### **SmolAgents vs. LangChain**

Originally, I tried **Hugging Face SmolAgents** with **LiteLLM** to keep it lightweight, but inference kept breaking. Debugging it was annoying, so I ditched SmolAgents and switched to **LangChain**, which:

* **Just worked**
* Had **better stability**
* Came with **solid community support**

## **What’s Next?**

### **DSPy for a React Agent**

I want to try **DSPy** to:

* Optimize the AI’s **prompting and response workflow**
* Make answers feel **more interactive**
* Fine-tune responses to **HR-style** interview questions with few-shot learning

### **Deeper Zoom SDK Integration**

Right now, I’m just using a meeting URL, but with the **Zoom SDK**, I could:

* Grab the candidate’s **resume from chat**
* Auto-generate **interview questions based on their resume**
* Dynamically adjust follow-ups in real time

### **Candidate Evaluation (Still to Integrate)**

I already wrote a function to **evaluate candidate responses**, but I haven’t plugged it in yet. Once I do, the system will be able to give structured feedback on interview performance.

## **Final Thoughts**

The goal was to build a simple, functional AI interviewer that works **locally** and doesn’t depend on paid APIs. I’ve got the basics down, and now I just need to refine the model, improve orchestration, and add more automation with Zoom integration. Plenty left to do, but the core system is solid!