**Meeting Notes App Documentation**

**Overview**

This app transcribes audio meetings using OpenAI Whisper and extracts notes & action items using the LLaMA 2 language model via Ollama. It runs locally without relying on paid OpenAI API calls by leveraging free/open source models.

**1. Installation & Setup**

**1.1. Prerequisites**

* Python 3.10 or higher installed
* Git (optional, for cloning repo)
* [FFmpeg](https://ffmpeg.org/download.html) installed and added to system PATH
* Ollama installed with LLaMA 2 model downloaded

**1.2. Install Python Dependencies**

Create a virtual environment:

python -m venv venv

Activate it:

* .\venv\Scripts\Activate.ps1

Note: If you get a script execution error, run PowerShell as admin and execute:

* Set-ExecutionPolicy RemoteSigned -Scope CurrentUser

Install required packages:

* pip install -r requirements.txt
* requirements.txt should contain:
* openai
* whisper
* python-dotenv
* ollama
* streamlit

**1.3. Install FFmpeg and Add to PATH**

* Download FFmpeg from [ffmpeg.org](https://ffmpeg.org/download.html)
* Extract and place the bin folder path in your system’s environment PATH variable:
  + Windows: Search **Edit system environment variables** > Environment Variables > Edit Path > Add the folder path of ffmpeg/bin
* Verify by running ffmpeg -version in a new terminal

**1.4. Install Ollama and LLaMA 2 Model**

* Download and install Ollama from ollama.com
* Install LLaMA 2 model:

ollama pull llama2

**2. Application Code**

**2.1. app.py**

import whisper

import ollama

# Step 1: Transcribe audio with Whisper

def transcribe\_audio(file\_path):

    print("Transcribing audio...")

    model = whisper.load\_model("base")

    result = model.transcribe(file\_path)

    return result["text"]

# Step 2: Extract meeting notes using LLaMA 2 via Ollama

def extract\_notes\_and\_actions(transcript):

    print("\nExtracting notes and action items with LLaMA 2...")

    prompt = f"""

You are a helpful assistant. Extract structured meeting notes and action items from the following transcript.

Transcript:

{transcript}

Respond in this format:

Meeting Summary:

- ...

Action Items:

- Person A: Action 1

- Person B: Action 2

"""

    response = ollama.chat(model="llama2", messages=[

        {"role": "user", "content": prompt}

    ])

    return response['message']['content']

# Main

if \_\_name\_\_ == "\_\_main\_\_":

    audio\_path = "meeting.mp3"

    transcript = transcribe\_audio(audio\_path)

    print("\nTranscript:\n", transcript)

    output = extract\_notes\_and\_actions(transcript)

    print("\nExtracted Notes & Action Items:\n", output)

**3.2. Run via Streamlit Frontend (Optional)**

Create a simple streamlit\_app.py:

import streamlit as st

from app import transcribe\_audio, extract\_notes\_and\_actions

st.title("📝 Meeting Notes & Action Item Extractor")

uploaded\_file = st.file\_uploader("Upload meeting audio", type=["mp3", "wav", "m4a"])

if uploaded\_file:

    with open("temp\_audio.mp3", "wb") as f:

        f.write(uploaded\_file.read())

    st.info("Transcribing...")

    transcript = transcribe\_audio("temp\_audio.mp3")

    st.subheader("Transcript")

    st.write(transcript)

    st.info("Extracting notes and action items...")

    result = extract\_notes\_and\_actions(transcript)

    st.subheader("Output")

    st.write(result)

4. Challenges & Solutions

| **Cause/Issue** | **Solution/Workaround** |
| --- | --- |

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | Some dependencies (e.g., Whisper, certain audio  libraries) don’t fully support Python 3.12 yet | Use Python 3.10 or 3.11 to ensure compatibility | |
| |  |  |  | | --- | --- | --- | |  | FFmpeg not added to system PATH or terminal session didn't reload environment variables | Added FFmpeg bin folder to system PATH, restarted terminal/VSCode | |
| |  |  |  | | --- | --- | --- | |  | Default security policy disallows running scripts | Changed policy with Set-ExecutionPolicy RemoteSigned -Scope CurrentUser | |