1. Prerequisites

Before starting, ensure you have the following:

- **Windows or Linux system**
- **Python 3.11.11 installed**
- **A terminal or command prompt to execute commands**
- **Stable internet connection** (to download required packages)
- **Audio file** (a `.wav` file with multiple speakers talking)

To check if Python is installed, run:

```
python --version
```

If not installed, download and install from Python's official website.

---

2. Setting Up the Virtual Environment

A virtual environment helps to keep dependencies isolated.

For Windows:

```
cd path\to\your\project-folder
python -m venv venv
venv\Scripts\activate
```

For Linux/Mac:

```
cd /path/to/your/project-folder
python -m venv venv
source venv/bin/activate
```

Your terminal should now show `(venv)`, indicating the virtual environment is active.

---

3. Installing Required Dependencies

Run the following command inside the virtual environment:

```
pip install -r requirements.txt
```

If `requirements.txt` is missing, install packages manually:

```
pip install pydub pyannote.audio openai-whisper librosa matplotlib ffmpeg
```

Additionally, install `ffmpeg`:

For Windows:

1. Download **FFmpeg** from https://ffmpeg.org/download.html
2. Add `ffmpeg` to system **PATH** (so it can be accessed from any terminal)

For Linux:

```
sudo apt update && sudo apt install ffmpeg
```

Verify installation:

```
ffmpeg -version
```

---

4. Project Workflow - Step by Step

Now that everything is set up, follow these steps to run the project.

Step 1: Convert Audio

This script converts the input audio to the required format.

Run:

```
python convert_audio.py
```

Output:

- `processed.wav` (converted audio file)

Step 2: Perform Speaker Diarization

This script identifies different speakers in the audio and marks when they talk.

Run:

```
python diarization_pyannote.py
```

Output:

- `diarization_results.json` (JSON file containing speaker segments)

Step 3 (Optional): Visualize Audio Waveform

This script generates a visual waveform of the audio.

Run:

```
python visualize_audio.py
```

Output:

- A graph showing the waveform.

Step 4: Full Audio Transcription

This script converts the entire processed audio into text.

Run:

```
python whisper_asr.py
```

Output:

- `whisper_transcript.txt` (full transcript of the audio)

Step 5: Generate Speaker-Wise Transcription

This script combines speaker identification with transcribed text.

Run:

```
python speaker_wise_transcription.py
```

Output:

- `final_transcript.txt` (speaker-labeled transcript)

---

5. Understanding the Code (Detailed Explanation)

Each script has a specific role. Let's break them down.

convert_audio.py (Audio Preprocessing)

```
from pydub import AudioSegment
```

- `pydub` is used to manipulate audio files.

```
audio = AudioSegment.from_file("2p_long.wav")
```

- Loads the input audio file.

```
audio = audio.set_frame_rate(16000)
```

- Converts the sample rate to 16kHz (needed for processing).

```
audio.export("processed.wav", format="wav")
```

- Saves the processed audio.

---

diarization_pyannote.py (Speaker Identification)

```
from pyannote.audio.pipelines import SpeakerDiarization
```

- Uses `pyannote.audio` for speaker diarization.

```
di = SpeakerDiarization.from_pretrained("pyannote/speaker-diarization",
use_auth_token=AUTH_TOKEN)
```

- Loads a pre-trained AI model to separate speakers.

```
di_output = di("processed.wav")
```

- Runs speaker identification.

```
with open("diarization_results.json", "w") as f:
    json.dump(results, f, indent=4)
```

- Saves speaker timestamps to a JSON file.

---

visualize_audio.py (Waveform Visualization)

```
import librosa, librosa.display, matplotlib.pyplot as plt
y, sr = librosa.load("processed.wav", sr=16000)
librosa.display.waveshow(y, sr=sr)
plt.show()
```

- Loads and displays the audio waveform.

---

whisper_asr.py (Full Transcription)

```
import whisper
model = whisper.load_model("base")
result = model.transcribe("processed.wav")
```

- Uses **OpenAI Whisper** to transcribe the entire audio.

---

speaker_wise_transcription.py (Speaker-Aligned Transcription)

```
with open("diarization_results.json", "r") as f:
    diarization_results = json.load(f)
```

- Loads speaker diarization results.

```
whisper_model = whisper.load_model("base")
```

- Loads Whisper ASR model.

```
for segment in diarization_results:
    transcript = transcribe_segment(start, end, speaker, "processed.wav")
```

- Matches text to speakers.

---

6. Running the Project End-to-End

For Windows:

```
cd path\to\project-folder
venv\Scripts\activate
python convert_audio.py
python diarization_pyannote.py
python whisper_asr.py
python speaker_wise_transcription.py
```

For Linux:

```
cd /path/to/project-folder
source venv/bin/activate
python convert_audio.py
python diarization_pyannote.py
python whisper_asr.py
python speaker_wise_transcription.py
```

---

7. Final Output Files

- **processed.wav** → Preprocessed audio.
- **diarization_results.json** → Speaker timestamps.
- **whisper_transcript.txt** → Full transcript.

- **final_transcript.txt** → Speaker-labeled transcript.