# Audiosense0 System Guide: Modular Audio Analysis Pipeline

The provided scripts form a robust, modular system (referred to here as **Audiosense0**) for detailed audio analysis. This system performs multiple simultaneous analyses (Speech Recognition, Diarization, Pitch, Speaking Rate, Sound Event Tagging, and Zero-shot Emotion Classification) and then merges all the data into a single summary file.

## 1. Environment Setup (Two Virtual Environments)

To avoid dependency conflicts, you must create two separate Python virtual environments.

### Step 1: Create Shared Requirements

First, create the two distinct requirements files based on the global requirements.txt:

# Venv 1: Core ML (ASR, AST, Prosody, CLAP)  
echo "faster-whisper  
torchcrepe  
transformers  
praat-parselmouth  
librosa  
soundfile  
numpy  
pandas" > requirements\_core.txt  
  
# Venv 2: Diarization (Pyannote)  
echo "pyannote.audio>=3.1  
torch  
torchaudio  
soundfile  
numpy  
pandas" > requirements\_diarization.txt

### Step 2: Set up Virtual Environments

| **Environment Name** | **Purpose** | **Installation Command** |
| --- | --- | --- |
| **venv\_core** | Run ASR, AST, CLAP, Pitch, Speaking Rate | python -m venv venv\_core |
| **venv\_diar** | Run Diarization (streaming\_argument.py) | python -m venv venv\_diar |

**Installation Commands:**

1. **Activate and Install venv\_core:**  
   # Activation (Linux/Mac)  
   source venv\_core/bin/activate  
   # Activation (Windows)  
   venv\_core\Scripts\activate  
     
   pip install -r requirements\_core.txt  
     
   # Deactivate  
   deactivate
2. **Activate and Install venv\_diar:**  
   # Activation (Linux/Mac)  
   source venv\_diar/bin/activate  
   # Activation (Windows)  
   venv\_diar\Scripts\activate  
     
   pip install -r requirements\_diarization.txt  
     
   # NOTE: Pyannote requires a Hugging Face token for model access.  
   # Follow setup steps on pyannote documentation or set the HUGGINGFACE\_TOKEN env var.  
     
   # Deactivate  
   deactivate

## 2. Core Analysis Scripts & Execution Flow

The analysis is performed on a single 16kHz mono WAV file. All outputs are fixed-time-bin CSV files, which are then merged by summarize.py.

### A. Execution Order (All 2-Second Binners)

| **File** | **Input (Audio)** | **Output (CSV)** | **Analysis Type** | **Key Metrics/Output** | **Run in Venv** |
| --- | --- | --- | --- | --- | --- |
| streaming\_ASR.py | audio.wav | audio\_asr.csv | **Speech Recognition** | Word transcripts, timestamps, normalized text. | venv\_core |
| streaming\_AST.py | audio.wav | audio\_ast.csv | **Sound Event Tagging** | Top-3 sound classes (e.g., Gunshot, Speech) per 2s bin. | venv\_core |
| streaming\_CLAP.py | audio.wav | audio\_clap.csv | **Zero-Shot Emotion/CLAP** | Flags like [STRESS], [AGITATION] based on CLAP model scoring against text prompts. | venv\_core |
| streaming\_argument.py | audio.wav | audio\_spk.csv | **Diarization/Overlap** | Flags like [HIGH\_OVERLAP], [INTERRUPTION], speaker count/switches. | venv\_diar |
| streaming\_pitch\_online.py | audio.wav | audio\_pitch.csv | **Pitch/Prosody** | Flags like [HIGH\_PITCH], [LOW\_PITCH] based on rolling pitch (F0) Z-score. | venv\_core |
| streaming\_speaking\_rate.py | audio.wav | audio\_prosody.csv | **Speaking Rate** | Flags like [RAPID\_RATE], [PAUSE\_HEAVY] based on syllable nuclei detection. | venv\_core |

### B. Summary & Classification

| **File** | **Input (CSVs)** | **Output (CSV)** | **Purpose** |
| --- | --- | --- | --- |
| summarize.py | All CSVs from above | audio\_summary.csv | **Data Aggregation:** Merges all flags and metrics into the \_ast.csv structure, and replaces generic AST labels like speech with the actual ASR text (\_asr.csv). |
| prompts.txt | (Used by downstream classifier) | N/A | **Decision Agent Rules:** Contains the system and hazard classification rules to be used by a final LLM or rule engine on the audio\_summary.csv data. |

## 3. Utility Scripts

These scripts are standalone tools for pre-processing or post-processing the system's files.

| **File** | **Input** | **Output** | **Purpose** | **Key Dependency** |
| --- | --- | --- | --- | --- |
| safe\_rename.py | Folder path | Renamed files/folders | **Pre-processing Utility:** Recursively cleans up filenames to ensure compatibility with Windows and scripts that fail on special characters. | None |
| embed\_srt.py | Folder path with MP4s and SRTs | New MP4 video file | **Post-processing Utility:** Uses the **FFmpeg system tool** to burn-in (-vf subtitles='...') an .srt subtitle file into a matching .mp4 video. | FFmpeg (System Tool) |