Milestone 1 Submission

Application Description:

```
# For our EE109 final project, we are implementing a Python-based audio
transcription and analysis system. It features a custom Digital Signal
Processing
# (DSP) frontend to extract log-Mel spectrograms, an Automatic Speech
Recognition (ASR) stage using OpenAI's Whisper API, and a Natural Language
Processing
# (NLP) module for text summarization and analysis (keyword and topic
identification).
```

Software Implementation:

```
# Link to GitHub Repo:
https://github.com/mohamedi01/ee109.git
cd Final-Project

# Core Components & Workflow:
The system processes input audio in three main stages, orchestrated by a
central pipeline:

1.  **Digital Signal Processing (DSP) Frontend (`audiolib.dsp.mel`)**
    *   The `wav_to_logmel` takes an audio file (various formats like WAV,
FLAC, MP3 supported) or raw audio data as input.
    *   Key Processing Steps: Loading & Resampling, Quantization
Simulation, STFT, Mel Filterbank, Logarithmic Compression & Scaling.
    *   Output: An 80-channel log-Mel spectrogram (`numpy.ndarray` of
`float32`).

2.  **Automatic Speech Recognition (ASR) (`audiolib.asr`)**
    *   `transcribe_features` transcribes log-Mel spectrograms to text
using a pre-trained Whisper model.
    *   `transcribe_audio_file` handles entire audio files, including a
sliding window mechanism for long audio and merging overlapping segments.
    *   Output: A string containing the transcribed text.

3.  **Natural Language Processing (NLP) (`audiolib.nlp.nlp`)**
    *   `analyze_text` performs keyword spotting, topic identification,
and summarization.
    *   Models are loaded via Hugging Face `transformers`.
    *   Output: A dictionary with keyword, topic, and summary.

# Pipeline Integration (`audiolib.pipeline`)
*   `process_audio_to_nlp` combines DSP, ASR, and NLP.
*   Output: Dictionary with transcript and NLP analysis.
```

```
# Main Features:
*   Custom DSP Frontend: Converts audio to Whisper-compatible log-Mel
spectrograms.
*   Whisper-based ASR: Transcribes speech to text, handling short and long
audio.
*   NLP Analysis: Performs keyword spotting, topic identification, and
summarization.
*   Integrated End-to-End Pipeline.
*   Comprehensive Testing Suite (pytest, WER metric).
```

Setup Instructions:

```
# 1. Create and activate a virtual environment
python -m venv ee109_final_project
source ee109_final_project/bin/activate
# On Windows
# venv\Scripts\activate

# 2. Install dependencies:
pip install -r requirements.txt

# 3. Run the Example Script:
# Execute the `run_pipeline.py` script as a module,
# providing the path to your audio file as a command-line argument.
# The script itself is at `src/audiolib/run_pipeline.py`
python -m src.audiolib.run_pipeline path/to/your/audiofile.wav

# Examples:
python -m src.audiolib.run_pipeline data/short_sentences/harvard_f.wav
python -m src.audiolib.run_pipeline data/long_sentences/bird.mp3

#4. Run all tests to see more information
pip install -e.
pytest
# See readme for more specific test cases (i.e just DSP, just DSP-ASR,
just NLP, etc..)
```

Performance Analysis:

```
# The project includes a comprehensive testing suite using pytest.
# Word Error Rate (WER) is a key metric for ASR performance.
# Further performance analysis to be implemented.
```