Audio_Analysis-API Documentation

V.Rakshitha

2010040123ece@gmail.com

Table of Contents

1. Project Overview
2. Audio Feature
3. Installation Guide
4. Usage Instructions
5. API Endpoints
6. How to Test with Postman
7. Deployment
8. Bonus Features
9. Tech Stack
10. Author and Contributing
11. License

Project Overview

Audio_Analysis-API is a FastAPI-based backend application designed to process audio files, extract speech features (MFCC, ZCR, RMS), and calculate a cognitive risk score. The API accepts multiple audio files, processes them to extract features, and calculates a cognitive score that could be used to assess speech patterns indicative of cognitive impairments.

Key Features:

- Upload 5–10 audio files in formats like .wav, .mp3.
- Extract Mel-frequency cepstral coefficients (MFCC), zero-crossing rate (ZCR), and root mean square (RMS) from the audio files.
- Calculate a cognitive risk score based on these extracted features.
- Clean up uploaded files after analysis to save space.
- RESTful API for easy integration and testing.

Audio Feature Breakdown

1. MFCC Mean (Mel-Frequency Cepstral Coefficients)

- **Definition**: MFCCs capture the timbral and phonetic characteristics of audio, especially voice. They're widely used in speech and speaker recognition.
- **Interpretation**:
  - **Lower values (more negative)** may indicate **less structured, more distorted, or less voiced** signals (e.g., noise, silence, impaired speech).
  - **Higher values (less negative or positive)** may suggest **more intelligible, tonal, or energetic** speech.
- **Examples**:
  - test01_20s.wav: **-18.21** → Possibly quiet, mumbled, or low-energy speech.
  - jackhammer.wav: **-5.46** → High-energy noise.

2. ZCR Mean (Zero-Crossing Rate)

- **Definition**: ZCR measures how often the signal changes sign (crosses zero), which reflects how noisy or percussive the signal is.
- **Interpretation**:
  - **Higher ZCR** → Sharp, noisy, or non-voiced elements (e.g., consonants, background noise, unstructured sound).
  - **Lower ZCR** → Smooth, voiced sounds (like vowels or continuous tones).
- **Examples**:
  - test01_20s.wav: **0.1238** → High ZCR, possibly noisy or chaotic.
  - Sports.wav: **0.0441** → More structured, consistent sound.

3. RMS Mean (Root Mean Square Energy)

- **Definition**: RMS represents the average power/energy of the signal, indicating loudness or amplitude strength.
- **Interpretation**:
  - **Higher RMS** → Louder or more energetic signal.
  - **Lower RMS** → Quiet or soft signal.
- **Examples**:
  - Chorus.wav: **0.1386** → Likely a loud or strong vocal section.
  - test01_20s.wav: **0.0446** → Softer, possibly weak speech or background.

🧠 Risk Score Interpretation

- The **cognitive risk score** is a weighted average based on:
  - MFCC: 50% weight
  - ZCR: 30% weight
  - RMS: 20% weight

Your Score: -4.771

- This **low (negative)** score reflects:
  - Generally **low MFCC values** across files (may suggest less coherent speech patterns).
  - A mix of medium-to-high **ZCR** (some noisiness or abrupt changes).
  - Varying **RMS**, but not particularly high (no consistently strong speech energy).

📌 Meaning (In Your Context)

- This might indicate:
  - Cognitive stress or low vocal engagement in the test file (test01_20s.wav).
  - Higher background noise or non-speech elements in other files (e.g., jackhammer.wav, Chorus.wav) influencing the aggregate.
  - Possibly impaired speech or passive audio input—could be worth isolating speech-only samples for clearer insights.

Installation Guide

Requirements:

Before setting up the project, ensure that you have the following:

- **Python 3.7 or higher** (Ensure you are using an appropriate version of Python for compatibility with FastAPI and required libraries)
- **pip** (Python's package installer)
- System dependencies such as ffmpeg for audio processing

Steps:

1. **Clone the repository:** To start, clone the repository to your local machine:

```
PS D:\visual studio\Memotag> git remote add origin https://github.com/rak-shi/Audio_Analysis-API.git
>> git branch -M main
>> git push -u origin main
>>
```

2. **Create a virtual environment:** It's recommended to use a virtual environment to manage your dependencies.

```
python -m venv venv
source venv/bin/activate  # For Linux/MacOS
venv\Scripts\activate     # For Windows
```

3. **Install dependencies:** Install the necessary Python dependencies listed in the requirements.txt file:

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

Usage Instructions

Running Locally:

To run the FastAPI server locally, use the following command:

```
PS D:\visual studio\Memotag> uvicorn main:app --reload
>>
```

This command will start the application on http://127.0.0.1:8000/. You can visit this URL in your browser to test the API.

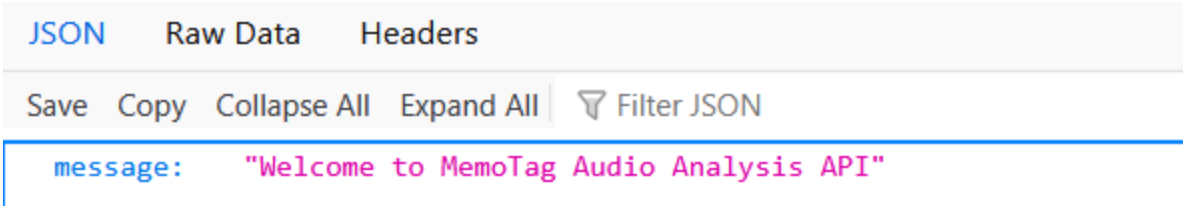
API Endpoints

1. GET /

Description:
Returns a welcome message with a short description of the API.

Response Example:

```
{
  "message": "Welcome to MemoTag Audio Analysis API"
}
```

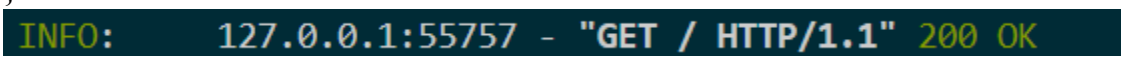


2. GET /health

Description:
Returns a simple health check response to verify the API is up and running.

Response Example:

{
"status": "ok"
}



3. POST /analyze/

Description:
Upload between 5 and 10 audio files for analysis. The API will process the audio files to extract features and return a cognitive risk score along with individual feature details for each file.

Request:

- **Method:** POST
- **URL:** /analyze/
- **Body:** Form-data with multiple audio files (5–10 .wav, .mp3 files).

Response body


```
{
  "result": {
    "risk_score": -4.771,
    "file_analysis": [
      {
        "file": "Chorus.wav",
        "mfcc_mean": -7.57984733581543,
        "zcr_mean": 0.06587161626887811,
        "rms_mean": 0.13863615691661835
      },
      {
        "file": "Conference.wav",
        "mfcc_mean": -7.222042083740234,
        "zcr_mean": 0.101806640625,
        "rms_mean": 0.05466792359948158
      },
      {
        "file": "jackhammer.wav",
        "mfcc_mean": -5.4604339599609375,
        "zcr_mean": 0.09931877162629758,
        "rms_mean": 0.06464697420597076
      },
      {
        "file": "Sports.wav",
        "mfcc_mean": -9.65786075592041,
        "zcr_mean": 0.04412201385171306,
        "rms_mean": 0.11571306735277176
      },
```


Notes:

- The cognitive risk score is calculated as a weighted sum of the mean MFCC, ZCR, and RMS values across all files.
- Each file's features (MFCC, ZCR, RMS) are returned with their mean values for analysis.

How to Test with Postman

You can use Postman to test the API by sending a POST request with audio files. Here's how to test the /analyze/ endpoint:

1. Open **Postman**.
2. Create a **POST** request to:

https://audio-analysis-api-6.onrender.com

3. In the **Body** tab, select **form-data**.
4. Add 5–10 key-value pairs:
    a. **Key:** files (set type to **File**)
    b. **Value:** Upload your audio files
5. Click **Send** to upload the files and receive the analysis and risk score.

Deployment

Deployment on Render

1. **Push code to GitHub**: Make sure your code is pushed to a GitHub repository.
2. **Deploy on Render**:
    a. Create a new service on Render (choose Python as the environment).
    b. Link your GitHub repository.
    c. Set the **Build Command** to:

        pip install -r requirements.txt

    d. Set the Start Command to:

uvicorn main:app --host 0.0.0.0 --port 10000

3. **Access the live URL**: After successful deployment, Render will provide a URL for your deployed API.

Tech Stack

- **FastAPI**: For building the API.
- **Librosa**: For audio processing and feature extraction.
- **NumPy**: For numerical computations like averaging features.
- **Render**: For cloud deployment of the application.

Author and Contributing

Author: rak-shi
This project is a prototype developed as part of MemoTag's research and development for cognitive using speech intelligence.

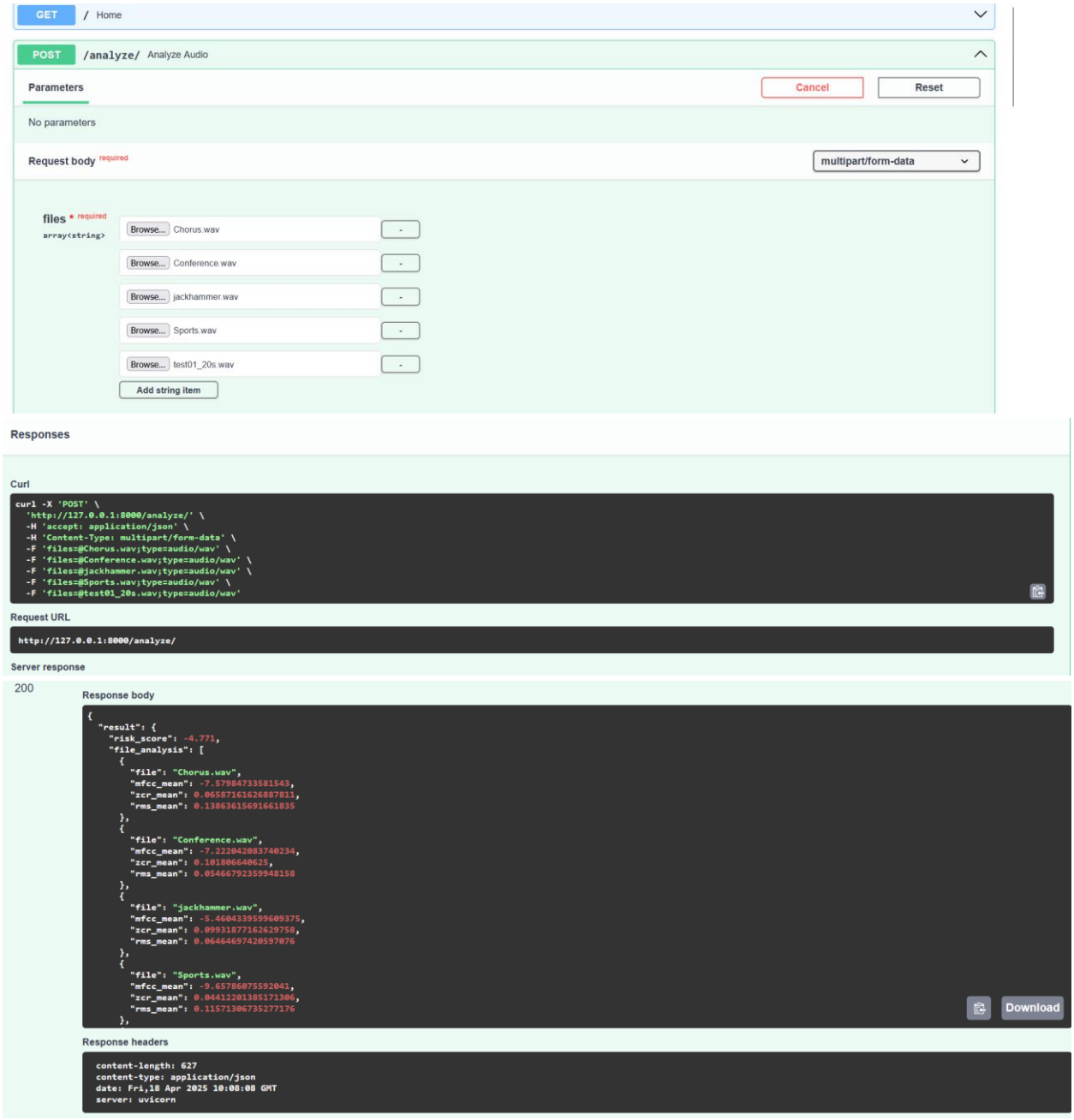
How to Contribute:

- Fork the repository.
- Make changes and create a pull request.
- Contributions are welcome!

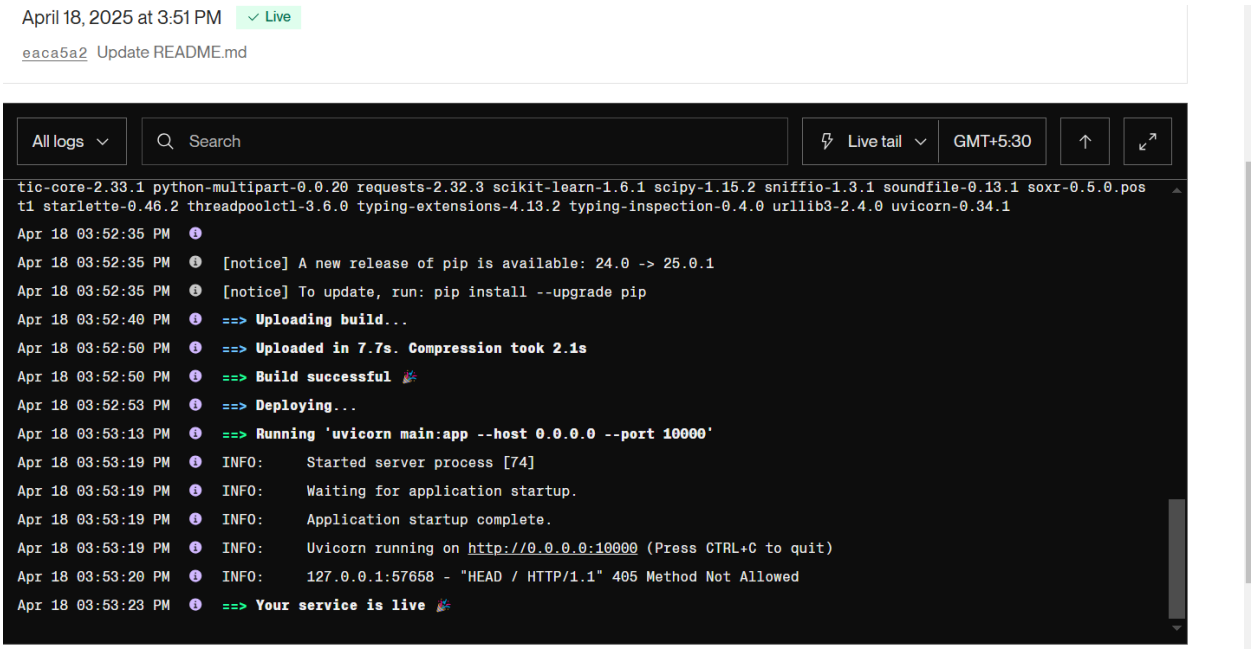
Result:

```
PS D:\visual studio\Memotag>  uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['D:\\visual studio\\Memotag']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [12140] using StatReload
INFO:     Started server process [12844]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:60587 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:60588 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:60592 - "GET / HTTP/1.1" 200 OK
INFO:     127.0.0.1:60593 - "GET /favicon.ico HTTP/1.1" 404 Not Found
INFO:     127.0.0.1:60596 - "GET /docs HTTP/1.1" 200 OK
INFO:     127.0.0.1:60596 - "GET /openapi.json HTTP/1.1" 200 OK
INFO:     127.0.0.1:60617 - "POST /analyze/ HTTP/1.1" 200 OK
```

127.0.0.1:8000/docs



API URL: https://audio-analysis-api-6.onrender.com



API URL: https://audio-analysis-api-6.onrender.com

Conclusion

The **Audio_Analysis-API** project offers a lightweight, modular, and efficient backend for analyzing audio files to assess cognitive risk using speech features. By leveraging tools like FastAPI, Librosa, and cloud deployment via Render, it enables rapid testing and real-time analysis through a simple REST API.

Whether you're building an early cognitive decline detection tool, enhancing health tech applications, or exploring voice-based analytics, this API serves as a strong foundation. With support for batch audio uploads, automatic feature extraction, and risk scoring, it’s both practical and extensible for research and product development.