Documentation - Django Intern Evaluation, applyin.co - Anant Yadav

---

1 Performance Metrics

- Average Transcription Latency: 0.84 seconds
- Average Accuracy: 0.89

2 Limitations of Product

1. Scalability of the webpage is not checked.
2. How the webpage responds to multiple users concurrently is not tested.
3. Implementing groups for real-life usage.
4. Challenge: Making it work on Vercel, a static website deployer, and WebSocket deployment challenges.

3 Challenges and Learnings

1. **Django Realtime Communication:**
    - Technologies: Django Channels, ASGI, Consumer, WebSocket.
2. **Realtime Transcription with Deepgram API:**
    - Technology: Deepgram API integration for real-time transcription, accuracy, and latency calculation.
3. **Deploying Django on Vercel:**
    - Approach: Vercel - Django Implementation.

4 ChatGPT/LLM Integration Documentation

4.1 Application Overview

- **Prompt:** Describe the Real-Time Audio Transcription Webpage application.
- **Output:** Provided an overview of the application's functionality, including the record button, Whisper API integration, and user interface design.

4.2 Recorder

- **Prompt:** Implement views for recording, transcribing, and displaying results.
- **Output:**

```
    # Real-Time Audio Transcription Webpage

    ## Setup JavaScript for Audio Recording and Whisper API Integration

    1. Open the HTML file where you want to implement audio recording and Whisper API integration
       (e.g., transcripti..../home.html).

    2. Add the following script to handle audio recording using Recorder.js.
       Make sure to adjust the code based on the Whisper API documentation.

    ‘‘‘html
    <!-- (e.g., transcripti..../home.html). -->
    <!-- ... (previous HTML code) ... -->

    <script>
        // Implement JavaScript for audio recording and Whisper API integration here
        // Adjust the code based on the Whisper API documentation

        // Example: Basic audio recording using Recorder.js
        var recorder;

        navigator.mediaDevices.getUserMedia({ audio: true })
            .then(function (stream) {
                recorder = new Recorder(stream);
            })
```

~~~~
            .catch(function (err) {
                console.log('Error accessing
                   ↪ microphone:', err);
            });

        function startRecording() {
            recorder && recorder.record();
        }

        function stopRecording() {
            recorder && recorder.stop();
            // Handle the audio data and send it to the
               ↪ server for transcription
        }
    </script>
    ```
~~~~

4.3 Error Handling

- **Prompt:** How can robust error handling be implemented for poor audio quality or network interruptions?
- **Output:** Suggested using try...except blocks, providing clear user feedback for poor audio quality or network interruptions.

4.4 Performance Metrics

- **Prompt:** What key performance metrics should be measured and reported, and how can they be calculated?
- **Output:** Advised measuring transcription latency and accuracy, with suggestions on using timestamps for latency and evaluating text with Levenshtein distance for accuracy.

4.5 Closing WebSocket Connection

- **Prompt:** How to automatically close the WebSocket connection after 90 seconds and restart it when the user clicks the record button?
- **Output:** Proposed using a timer or `setTimeout` function to close the connection after 90 seconds, with the initiation of a new connection on the record button click.

4.6 Handling Rate Limit Error

- **Prompt:** How to handle the 'RateLimitError' from the OpenAI Whisper API?

- **Output:** Advised implementing try...except blocks to catch the 'RateLimitError' and providing user feedback to check the plan and billing details.

4.7 Django Channels WebSocket Connection

- **Prompt:** How to configure WebSocket connection in Django Channels consumer?
- **Output:** Shared a general structure for connecting to Deepgram using Django Channels, including connection, disconnection, and receiving functions.

4.8 Connecting to Deepgram with Django Channels

- **Prompt:** How to connect to Deepgram using Django Channels and handle WebSocket events?
- **Output:** Presented a code snippet for connecting to Deepgram, handling CLOSE and TRANSCRIPT-RECEIVED events, and raising exceptions for connection issues.

4.9 Code Generation

- **Task:** Write Python code to send real-time audio transcriptions using Deepgram API through Django Channels and WebSockets. Include handling for transcript and accuracy.
- **Subtasks:**
  1. Connect to Deepgram API using provided credentials.
  2. Implement WebSocket consumer to handle real-time transcription events.
  3. Extract transcript and accuracy from received data and send them through WebSocket.
  4. Ensure proper connection handling and message reception.

4.10 Transcription Latency Calculation

- **Task:** Provide a Python function to calculate transcription latency based on Deepgram API duration field.
- **Subtasks:**
  1. Extract duration field from the transcription result.
  2. Return the duration as the transcription latency in seconds.

5 Useful Links:

- Django Channels
- Live Transcript using Deepgram API
- Realtime-Audio-Transcriber
- Deployable Vercel link
- Video Link
- Whisper Real time