Backend Assignment

1)REST API for Interacting with Audio Elements using django.

Goal

The goal of this assignment is to create a simple REST API service that let's users/clients add/edit/get/delete audio-elements in projects created on a video-editing platform.

Description

Consider a video-editing service named "VideoMaker". Projects in this platform are stored in JSON format. An example project in JSON format can be found in example.json file. Any project has two sets of elements video_blocks and audio_blocks. This assignment is primarily working on the audio_blocks.

You need to create a REST API that allows users to perform CRUD operations for the audio-elements in the project. VideoMaker app supports three types on audio-elements:

- Voice Over (represented as vo)
- Background Music (represented as bg_music)
- Video Music (represented as video_music): This element is a placeholder to represent audio within a video. For this element, it stores the reference to an existing video block and doesn't duplicate duration/url information.

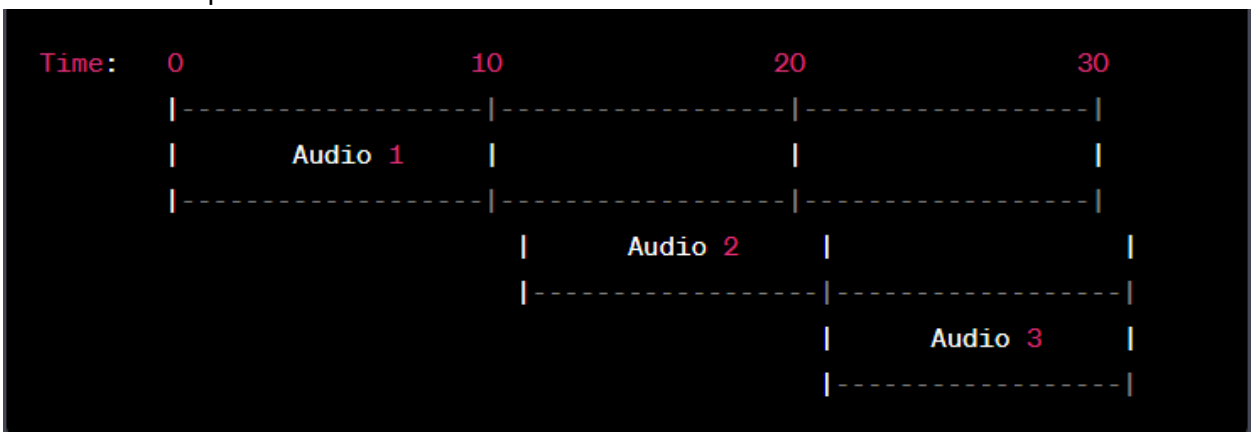
Description of fields in Audio-Element:

Each audio-element has following fields:

```
{
      "id": "<unique identifier generated server side>",
      "type": <vo|bg_music|video_music>",
      "high_volume": <volume when this element is not overlapping with
other audio element>,
      "low_volume": <volume when this element is overlapping with other
audio element>,
      "video_component_id": <video-component-id if type is video_music else
null>,
      "url": <url for the audio file. null if type is video_music>
      "duration": {
            # this is null if type is video_music
            start_time: <start time for audio>
            end_time: <end_time for audio>
      }
}
```

Audio Element Volume

As seen above, an audio-element have two different type of volumes defined i.e. High volume and Low volume. This is because audio-elements of different types can overlap. A particular audio-element plays at High volume when its not overlapping with any other audio-elements, otherwise each of the overlapping elements plays at Low volume.
In this visualization, we'll use different colors to represent different audio elements. When two or more audio elements overlap, they will be shown as stacked bars. The height of the bars will indicate the volume of each audio element.
Here's an example:



In this example, we have three audio elements: Audio 1, Audio 2, and Audio 3. At time 0, only Audio 1 is playing, and it occupies the full height of the bar, indicating high volume. At time 10, Audio 1 stops playing, and Audio 2 starts. Since there is an overlap between Audio 2 and Audio 1, both are shown at a reduced height, indicating low volume. At time 20, Audio 2 ends, and Audio 3 starts. Again, there is an overlap between Audio 3 and Audio 2, so both are shown at a reduced height.

API

Add Audio Element

- Add an audio-element to the project. This API will also be called in the event to add audio-elements for the video-elements - in which case it receives the video_component_id from the user/client.
- It is possible for different types of audio-elements to overlap, but two audio-elements of the same type cannot overlap. For example if there is an existing vo element from 5 - 20 seconds, and we receive a new vo element to be added from 15 - 25 seconds, it should actually be added from 20 - 30 seconds (i.e. subsequent empty slot).
- Return a valid HTTP 201 response if the audio-element is successfully added, or verbose error response in case of failure.

```
Timeline:           0-----5-----10----15----20----25----30----35----40
```

```
Existing vo element:       |--------vo--------|
New vo element (requested):            |--------vo--------| (15-25)
New vo element (adjusted):                      |--------vo--------| (20-30)
```

Get Audio Element By ID

- Return an audio-element JSON by ID.
- For video_audio, it should populate the duration and url from the corresponding video-block.
- Return a valid HTTP 200 response if the audio-element is found, or HTTP 404 response error response in case of failure.

Delete Audio Element by ID

- Delete an audio-element by ID.
- In the case of video_audio, we shouldn't delete the original video component.
- Return a valid HTTP 200 response if the audio-element is deleted, or verbose response error in case of failure.

Update Audio Element By ID

- Update an audio element by ID. This method also accepts a payload of the field to update for the corresponding audio-element.
- In the case of video_audio, we shouldn't update the original video component.
- Return a valid HTTP 200 response if the audio-element is updated, or verbose response error in case of failure.

Get Audio Fragments between start-time and end-time

- This API must return all audio-fragments that exist between the provided start and end-time.
- Unlike, audio-element, an audio-fragment has only one volume - i.e. volume at which that particular fragment must be played. What it means by this is that an audio element is divided into multiple audio-fragments depending upon whether it overlaps with other elements or not. Read the example below for better understanding. Consider a video with three audio elements:
    - vo with start time at 5 seconds and end time at 20 seconds.
    - bg_music with start-time at 10 seconds and end-time at 40 seconds.
    - video_music with start-time as 15 seconds and end-time as 25 seconds.

We will get following audio fragments for this video between 0 - 30 seconds
+ vo with start time as 5, end time as 10, volume as High Volume

+ vo with start time as 10, end time as 20 volume as Low Volume

+ bg_music with start time as 10, end time as 25, volume as Low Volume

+ bg_music with start time as 25, end time as 30, volume as High Volume

+ video_music with start time as 15, end time as 25, volume as Low Volume

```
Timeline:    0-----5-----10----15-----20------25----30

vo:                |--------vo--------|

bg_music:                |---------bg_music---------|

video_music:                     |--video_music-|
-------------------------------------------------------------------------
Audio Fragments:

1. vo:               |--vo--| : High Volume (5-10)
2. vo:                    |--vo--------| : Low Volume (10-20)
3. bg_music:                   |--bg_music--------| : Low Volume (10-25)
4. bg_music:                                |--bg_music--| : High Volume (25-
30)
5. video_music:                          |--video_music--| : Low Volume (15-
25)
```

Example response for the project json:

```
[
{
    id: "123",
    url: "",
    volume: 100,  # High volume no overlap
    type: "vo",
    duration: {
        start_time: 5,
        end_time: 10
    }
```

```
},
{
    id: "123",
    url: "",
    type: "vo",  # Low volume since overlap
    volume: 75
    duration: {
        start_time: 10,
        end_time: 20
    }

},
{
    id: "456",
    url: "",
    type: "bg_music", # Low volume since overlap
    volume: 25
    duration: {
        start_time: 10,
        end_time: 25
    }
},
{
    id: "456",
    url: "",
    type: "bg_music", # High volume since overlap
    volume: 100
    duration: {
        start_time: 25,
        end_time: 30
    }
},
{
    id: "abc",
    url: "",
    type: "video_audio", # Low volume since overlap
    volume: 50
    duration: {
        start_time: 15,
        end_time: 25
    }
}
]
```

Deliverables

- A simple server with the following set of API's as mentioned above.
- Source code committed (and pushed) on github(preferable)/bitbucket/gitlab.
- Readme file to let us know how to run the app and any other documentation that you might want to share.
- Deployed on heroku or any other hosting service provider so that we can test the API.

Submission

Form link: https://forms.office.com/r/1txQ3mEyEV

-Fill the mentioned form above for submitting the assignment.

Instructions

- We have included an example video json file, along with its video fragments for your reference.
- The most important part of this assignment is how you design your abstractions and REST API.
- The code should handle edge cases and do proper error handling.
- Please ask questions. It's ok to have doubts and we are eager to answer those.

In case of any queries please contact the following
1)Hema hema@vigaet.com
2)Chinmay chinmay.p@vigaet.com
3)Siddesh siddesh@vigaet.com
