Spotter UX Engineer Interview

Develop a UX that allows users to input and modify a YouTube influencer's beat sheet. A beat sheet, originally a concept from screenwriting, is a kind of outline used to plan the content and structure of a piece of media. For YouTube influencers, a beat sheet could be a way to plan out the content of a video or series of videos.
Here's a simple example of what a beat sheet might look like for a YouTube video:

- **Intro (0:00-0:15):** Brief introduction of the host (the influencer) and the topic of the video.
- **Title/Opening Credits (0:15-0:30):** The opening title or credits for the video.
- **Hook (0:30-1:00):** A hook to draw viewers in and get them interested in the rest of the video.
- **Overview (1:00-1:30):** A more detailed overview of what will be covered in the video.
- **Main Content - Part 1 (1:30-5:00):** The first part of the main content. This could be a tutorial, product review, story, etc.
- **Transition (5:00-5:15):** A brief transition between the first and second parts of the main content.
- **Main Content - Part 2 (5:15-9:00):** The second part of the main content.
- **Conclusion (9:00-10:00):** Wrapping up the video, summarizing the key points, and perhaps providing a tease of what's coming in the next video.
- **Call to Action (10:00-10:30):** Asking viewers to like, subscribe, comment, etc.
- **End Credits/Outro (10:30-11:00):** End credits, bloopers, or other outro material.

In the context of this exercise we will assume that a beatsheet is made of acts and each act is made out of beats . Your task is to develop the UX for the beatsheet. An act can be represented as a JSON object as follows:

```JavaScript
{
  "name": "string"
}
```

A beat is represented by the following structure:

```javascript
{
  "name": "string",
  "time": "string",
  "content": "string",
  "cameraAngle": "string",
  "notes": "string"
}
```

Provided APIs

We have provided a service that can handle all the backend operations for the beatsheet. Here's a summary of the available apis:

1. **GET /acts**
   - **Description**: Retrieves a list of acts.
   - **Path Parameters**: none
   - **Body Parameters**: none
2. **POST /acts**
   - **Description**: Creates a new act
   - **Path Parameters**: none
   - **Body Parameters**: Act object (JSON)
3. **GET /acts/{id}**
   - **Description**: Retrieves an act by id
   - **Path Parameters**:
     i. id: The ID of the act to retrieve.
   - **Body Parameters**: Act object (JSON)
4. **DELETE /acts/{id}**
   - **Description**: Deletes an act and its subsequent beats
   - **Path Parameters**:
     i. id: The ID of the act to delete (long)
   - **Body Parameters**: none
5. **PUT /acts/beats/{id}**
   - **Description:** Updates a beat within an act.
   - **Path Parameters:**
     i. id: The ID of the beat to be updated (long).

    - **Request Body**: Beat object (JSON)

6. **DELETE /acts/beats/{beatId}**
    - **Description**: Deletes a beat within an act
    - **Path Parameters**:
        i. beatId: The ID of the beat to delete (long)
    - **Body Parameters**: none
7. **GET /acts/{id}/beats**
    - **Description:** Retrieves a list of beats within an act.
    - **Path Parameters**:
        i. id: The ID of the act to retrieve beats from (long).
    - **Body Parameters**: none
8. **POST /acts/{id}/beats**
    - **Description**: Adds a new beat to an act.
    - **Path Parameters**:
        i. id: The ID of the act to add the beat to (long)
    - **Body Parameters**: Beat JSON object
9. **GET /beats/{id}**
    - **Description**: Retrieves an beat by id
    - **Path Parameters**:
        i. id: The ID of the beat to retrieve.

Prerequisites

- Docker
- Clone the service from https://github.com/fmatar/beatsheet-exercise

```
Unset
    git clone git@github.com:fmatar/beatsheet-exercise.git
```

Start the backend

Start the docker containers in the cloned repo:

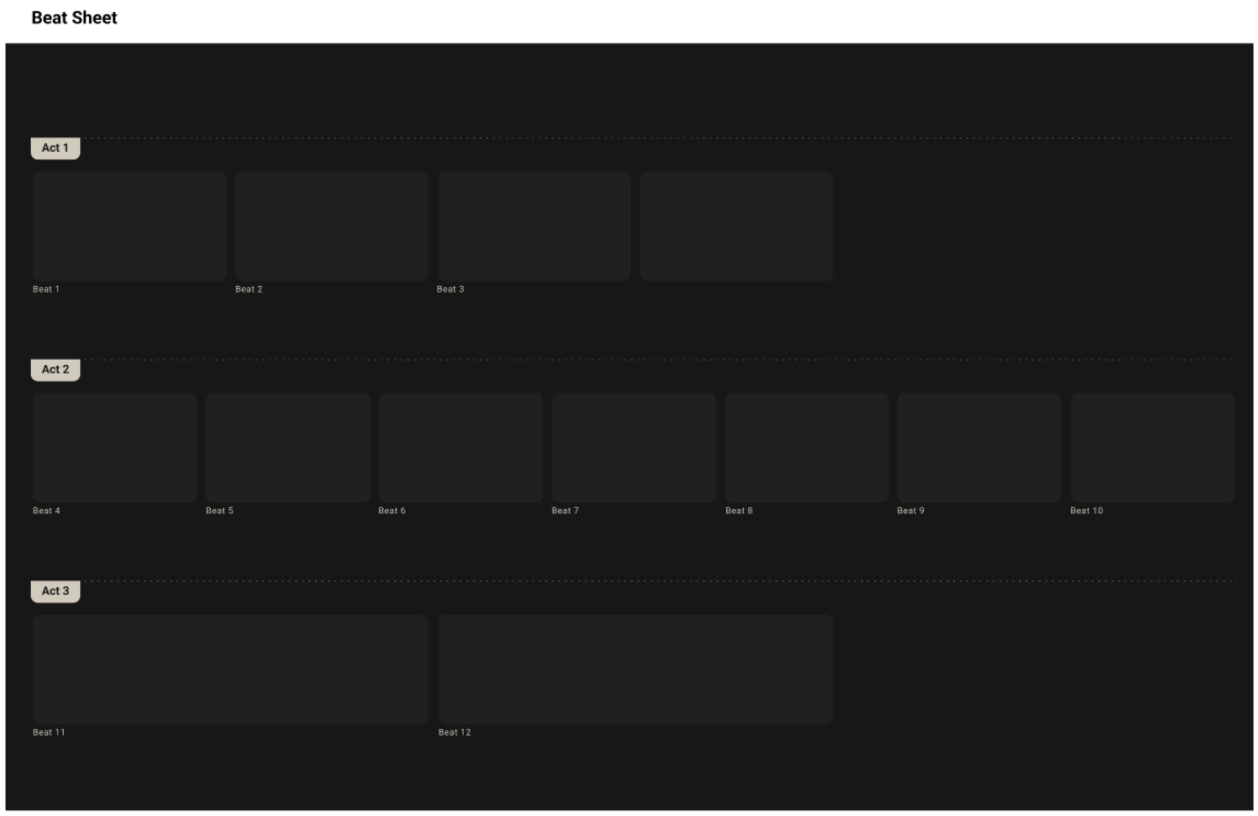
```
Unset
    docker compose create
docker compose start
```

Access the service and documentation:

1. The service is running at: http://localhost:8080
2. The Swagger documentation is available at: http://localhost:8080/swagger-ui

Let's get the job done!

3. Create the interface you can use the following mock as a starter, feel free to improvise and update. Do not limit yourself with this design, this is only a mock.



4. Implement the following features in the application:
   - List all acts and all their subsequent beats
   - List the beats in a specific act
   - Create a new act

    - Delete an act (Will also delete all the subsequent beats associated with)
    - Create a beat in an act
    - Update a beat in an existing act
    - Remove a beat from the act
5. Make sure the UI is responsive for various displays (mobile, tablet, desktop)
6. Use proper state management (You may stick to React defaults or use your own such as Redux or others)
7. Add your twist to the application, beautify it with animations and cool design techniques as you see fit. (Attention to details is a must!)

Deliverables

- Push your code on github and be ready to share it with the team. You can share the link with the recruiter and they can communicate this information to us
- Include a README on how to run and package the application
- Make sure your code follows coding conventions and best practices
- Bonus: Deploy your solution to Vercel (http://vercel.io)