Objective:

Create an “audio pill” player —inspired by a simplified version of iMovie, but for audio only. Users should be able to:

1. Upload audio files individually (e.g., .mp3, .wav).
2. Arrange and Shift these “audio pills” on a timeline.
3. Manage Multiple Tracks, each track holding one or more audio pills.
4. Play Back the sequence dynamically, reflecting any changes as the user rearranges clips in real time.

This challenge is designed to assess your ability to build an interactive, stateful React application that handles dynamic content and user-driven interactions.

Core Requirements

1. Audio Upload & Rendering
   - ○ Provide a clear way to upload audio files from the user’s local machine.
   - ○ Display each uploaded file as a distinct “pill” or segment within the UI.

2. Timeline & Multi-Track Support
   - ○ Users should be able to create multiple tracks (e.g., Track 1, Track 2, etc.).
   - ○ Drag-and-drop functionality that allows rearranging pills within a track or potentially between tracks.

3. Playback Controls
   - ○ Provide play/pause buttons and show the current playback position.
   - ○ Ensure that if the user repositions or removes an audio pill on the timeline, playback order updates accordingly.

4. Dynamic Updates
   - ○ Changes to the timeline (e.g., moving pills around) should be immediately reflected in the audio playback sequence.
   - ○ Handle any potential overlapping or edge conditions gracefully.

5. React-Based Implementation
   - ○ Use React (with or without frameworks like Create React App or Next.js).
   - ○ You may use any libraries for drag-and-drop (e.g., react-dnd), audio handling, etc., as long as your approach is clear.

Deliverables

1. GitHub Repository
    - Containing your React application code.
    - Clear instructions in a README.md for installing dependencies, running the development server, and building for production.

2. Short Technical Write-Up
    - Describe your approach, including any libraries or design patterns used.
    - Mention key challenges you encountered and how you resolved them.
    - Briefly outline potential enhancements you'd consider if given more time.

3. Demo Video (Loom or similar)
    - A 3–10 minute walkthrough demonstrating:
        - How to upload audio files.
        - How to arrange the audio pills across multiple tracks.
        - Playback in action, showing that reordering in the timeline updates playback order.
    - A short code overview, highlighting your structure and main components.

4. Sample implementation that took our engineer 3 hours to complete: https://iaudio.netlify.app/

Evaluation Criteria

1. Completeness & Functionality
    - Does the application meet the required features (upload, timeline, multiple tracks, dynamic playback)?
    - Are there any bugs or missing pieces?

2. Code Quality & Organization
    - Is your codebase well-structured, with clear separation of concerns (components, utilities, etc.)?
    - Are best practices followed (naming conventions, readability, maintainability)?

3. User Interface & UX
    - Is the drag-and-drop interaction smooth and intuitive?
    - Do playback controls respond promptly to user actions?

4. State Management
    - ◯ How effectively do you handle the audio state across multiple tracks and pills?
    - ◯ Does your approach minimize unnecessary re-renders or performance bottlenecks?

5. Documentation & Clarity
    - ◯ Is the setup process easy to follow in the README?
    - ◯ Does your write-up explain design decisions and trade-offs clearly?

Ready to get started?
We look forward to reviewing your code, reading about your approach, and watching your demo!
Good luck—and bring that Mamba mentality!