When storing user voice recordings in a MongoDB database, is it better to store the raw audio files directly inside MongoDB, or to store only a link to the file hosted in cloud storage?

Answer:

It is better to store voice/audio recordings in cloud storage (like AWS S3, Google Cloud Storage, or Firebase Storage) and only keep the file's URL in MongoDB instead of storing the raw audio directly.

Reasons:

1. **Performance** – Databases are optimized for structured data (documents, numbers, metadata), not for serving large binary media files. Cloud storage is optimized for fast media access and streaming.

2. **Scalability** – Cloud storage can handle very large files and millions of requests efficiently, while MongoDB can become slow and heavy if large files are stored directly.

3. **Cost Efficiency** – Storing and retrieving files from cloud storage is cheaper compared to database storage.

4. **Ease of Use** – Applications can directly stream or play the audio from the cloud using the URL (e.g., via `<audio>` tags or mobile media players).

5. **Separation of Concerns** – MongoDB stores the metadata (filename, userId, duration, and file link), while the cloud handles the actual audio.

Step-by-step Flow

1. **User speaks / records**
    - Example: A blind user presses a mic button and says *"Find me IT jobs in Nagpur"*.
    - At this point, their **voice (sound waves)** is captured by the phone's microphone.

2. **Audio file is created in the app (this is what "generated in the app" means)**
    - The app takes those sound waves and **converts them into a digital file** (e.g., `.wav`, `.mp3`).
    - This file is temporarily saved **inside the app's storage or memory**.
    - Without this step, the voice cannot be processed, because the system needs a digital file format.

3. **Audio is uploaded to backend / cloud**
    - That digital audio file is then **sent to your server or cloud storage (like Firebase, MongoDB, or Google Cloud Storage)**.
    - So now the file exists outside the app too, safely stored.
4. **Speech-to-Text Conversion**
    - A Speech-to-Text engine (like Google Speech API) processes the file.
    - The spoken sentence *"Find me IT jobs in Nagpur"* is turned into **text**.
5. **App understands the request**
    - The text is read by your system (NLP or backend).
    - The system understands that the user is looking for IT jobs in Nagpur.
6. **System fetches results**
    - Backend searches the database (MongoDB, job listings, etc.) for relevant jobs.
7. **Results sent back**
    - The job listings are sent back to the app as text data.
8. **App gives output in accessible format**
    - The app can:
        - **Read out loud** (Text-to-Speech → "Here are 5 jobs in Nagpur").
        - Or show results in **big text, Braille display, or sign language animation**, depending on disability type.

in short: **Voice → Audio File → Upload → Speech-to-Text → Process → Fetch Jobs → Output (Voice/Text/Other)**