🎵 Outlever – Podcast Summarizer Task

Project Background

We're exploring new AI-powered experiences and want to test how you approach building a clean, fast prototype with autonomy and good judgment. This task focuses on building a **standalone podcast summarizer tool** that fetches podcasts, lets users choose one, and summarizes the selected episode using an AI model. The summaries should be saved to MongoDB.

Our goal is to see how you think through problems, design interfaces, and integrate external APIs and LLMs to solve a user-centric problem.

Your Task

You're expected to build a **single-purpose web app** with the following flow:

User Flow:

1. On load, the user sees a list of podcast episodes (fetched from the Listen Notes API).
2. The user can select an episode and click a “Summarize” button.
3. The app uses Gemini (preferred) (or any LLM of your choice) to generate a **summary** of the podcast episode.
4. The summary is displayed in the UI after it's complete.
5. On revisiting or refreshing, if a summary already exists for that episode, show the saved version instead of regenerating.

Feature Requirements

🔍 Podcast Discovery

- Use the free tier of the **Listen Notes API** to fetch a list of podcast episodes
- Display episode **title, publisher, thumbnail, and description** in a clean layout.

🧠 Summarization

- Let the user click **“Summarize”** on any episode.
- Fetch episode audio or description and pass it to a summarization endpoint using **an LLM**.
- Display the generated summary in the UI after it's complete.

📂 Persistence

- Store each generated summary in **MongoDB**, linked to the podcast episode ID or URL.
- If a summary for that episode already exists, return the saved version instead of calling the LLM again.

□ Architecture

- NextJs and React on the frontend, Tailwind and Shadcn UI also preferable.
- Backend can be integrated directly in the app or separated (your choice).

Technical Considerations

- **Listen Notes API**: Free version available — https://www.listennotes.com/api/docs/
- **Summarization Model**: Gemini (preferred), or any model that suits the task

- **MongoDB**: Set up a simple schema to store podcast ID + summary + timestamp
- **Error Handling**: Gracefully handle failures from Listen Notes or LLM APIs
- **Performance**: Show loading states and handle async operations cleanly

---

Deliverables

- Working code hosted on GitHub
- Deployed demo (e.g., Vercel, Netlify)
- Video walkthrough (screen recording):
    - Overview of the app
    - Your thought process and architecture
    - Challenges or trade-offs
    - Any known limitations

Timeline & Design

- You have **3 days** to complete the task.
- There is **no design spec** — we want to see your UI/UX taste.
- Bonus points for:
    - Clean architecture
    - Reusability and composability
    - Responsive and polished UI

- Thoughtful state management
- Efficient prompt design for summarization

Questions?

Reach out to Emma if anything is unclear or you'd like to clarify a constraint.