Technical Decision-Making

I used Next.js to handle both backend and frontend within a single repository. Since the project is small and doesn’t use a database, I opted for an all-in-Next.js approach to keep it simpler to manage and deploy. Because Next.js is created by Vercel, I chose Vercel as the deployment platform.

One reason I avoided using a database is to allow users to keep their tokens on their side. I store their tokens as HTTP-only cookies to prevent them from being stolen via XSS attacks.

To avoid losing the converted video when navigating between pages, such as going to the Google OAuth page, I save it to IndexedDB. I don’t use localStorage because video files can be large and may not fit within localStorage limits.

For the UI, I used shadcn/ui since it provided the simple, clean look I wanted for the project.

One downside of not having a backend is that, since I planned to deploy on Vercel, I couldn’t download ffmpeg binaries there, as it wasn’t recommended due to size limits. Therefore, I used ffmpeg.wasm, which runs in the browser. Since the browser environment is slower than the server environment, the conversion is a bit slower. However, because we’re creating fully black videos, adjusting the resolution makes it possible to quickly upload long audio files.

I used AI to understand the ffmpeg.wasm and Google Auth libraries. I first asked Cursor to write the necessary code for audio conversion and the Google Auth process. While it didn’t work perfectly, it gave me an idea of how to use these essential libraries. I reviewed documentation, browsed forums, found examples, and built upon the AI-generated code. I also used AI to assist with straightforward tasks like error handling and finding allowed audio types. However, the AI-generated code required cleanup due to redundancies and inconsistencies, which only a human can reliably fix.