Find the Tempo

Turn a screenshot of songs into openable streaming links

Project

Find the Tempo. Turn a screenshot of songs into openable streaming links

Role

Solo. Product, design, build

Timeline

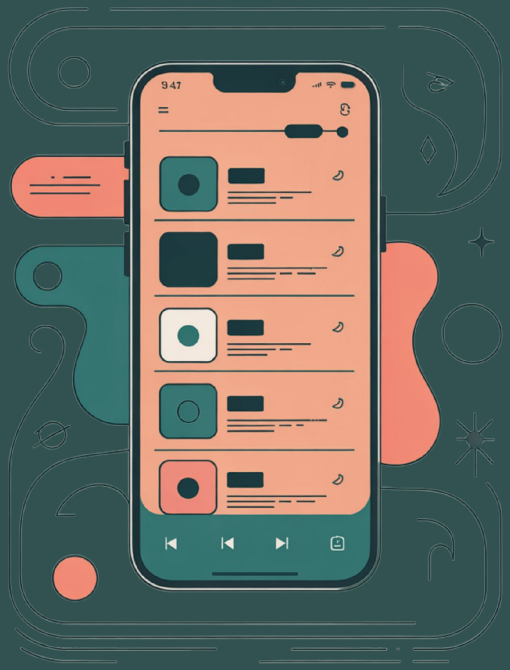
July 2026

Tools

Next.js, TypeScript, Tailwind, shadcn/ui, Vertex AI OCR, Netlify, GA4

Live

findthetempo.netlify.app



The Problem

Song recommendations arrive scattered: screenshots, shared links, Shazams. Collecting them into one playable list is manual work, and most of them get lost.

I built this because it was my problem. Friends kept sending me song names as Apple Music screenshots. I don't use Apple Music, so I had no way to turn those names into something I could play.

Who It's For

Primary user

People who collect tracks across several streaming services and want them in one place.

What I know

My own behaviour as the first user.

What I don't know yet

Whether the real segment is everyday listeners or DJs building crates. Two surveys are drafted to answer that.

Goals and Success Signals

User goal

Screenshot in, playable links out, minimum steps in between

Product goal

Test demand before paying for deeper integrations

Success signals

Candidates: repeat uploads in GA4, willingness-to-pay answers from the surveys, link clicks per upload. Pick the ones I will actually check.

Constraints

Cost

Streaming API pricing put the original playlist-conversion idea out of reach for a user base this small

API access

No official partner APIs available to me. Beatport is partner-only, Bandcamp has no API. OAuth and quota terms add work the MVP doesn't justify

Solo builder

Shipping speed beats polish at this stage

Options Considered

1 **Full playlist conversion via streaming APIs**

The experience users want, but the API cost was unjustifiable for unproven demand.

2 **Query-based search links, no API**

The user lands on the service's search page instead of the exact track. Less impressive, near-zero cost, no API terms, ships in days.

DECISIONS

Decision 1 — Test before building

Problem

Full conversion was the vision and the expensive part

Choice

Ship the query-search core only, then research whether people would pay

Why

Building the expensive version first bets real money on demand nobody has confirmed

Revisit when

Survey results show willingness to pay, or a partner API opens up

Decision 2 — Core function first, design later

Problem

Version 1 had no design. Black and white, no header, no footer, nothing aligned

Choice

Held off on visual design until the core function worked

Why

A styled app that doesn't work proves nothing. An ugly app that works proves the concept

DECISIONS

Decision 3 — Analog hardware, not another SaaS gradient

Problem

Once the function worked, the app needed an identity

Choice

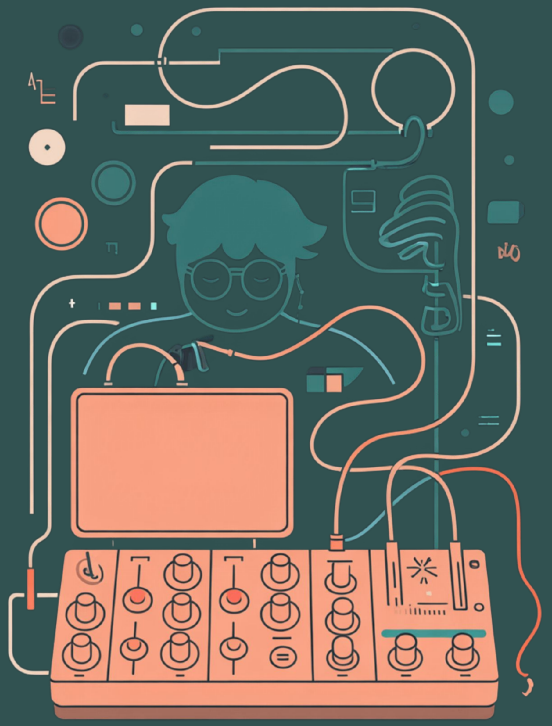
Bright orange palette, borrowed from Teenage Engineering mixers

Why

DJ and audio hardware uses bright, high-contrast colour so performers can find controls in a dark room. The product is about music, so the visual language of music hardware connects the look to the job

Revisit when

The segment question is answered. A DJ product can push this further, a consumer product may need softening



Visual Design System

Colours

Defined as OKLCH custom properties in app/globals.css. Hex values below are the sRGB equivalents.

Light Theme

Role	Colour	OKLCH / Hex
Primary accent (--primary, --accent, --ring)	Signature orange	oklch(0.71 0.22 34) / #FF5C34
Background (--background)	Warm grey, tinted toward the accent	oklch(0.936 0.026 84) / #F2E9D7
Card / panel (--card, --popover)	Near-white cream	oklch(0.995 0.012 84) / #FFFDF5
Text (--foreground)	Warm near-black ink	oklch(0.17 0.025 39) / #190B07
Muted text (--muted-foreground)	Warm grey	oklch(0.46 0.027 66) / #635548
Secondary surface (--secondary, --muted)	Warm grey	oklch(0.9 0.026 84) / #E6DDCB
Border / input (--border, --input)	Warm grey	oklch(0.8 0.026 84) / #C6BDAB
On-accent text (--primary-foreground)	Off-white	oklch(0.99 0 0) / #FCFCFC
Destructive (--destructive)	Red	oklch(0.577 0.245 27.325) / #E7000B

Dark Theme

.dark, a separate palette, not a filter over the light one

Role	OKLCH	Hex
Primary accent	oklch(0.74 0.185 38)	#FF794B
Background	oklch(0.175 0.022 66)	#170F06
Card / panel	oklch(0.225 0.022 66)	#231A11
Text	oklch(0.95 0.022 84)	#F5EEDE
Muted text	oklch(0.72 0.023 75)	#ADA395
Secondary / accent surface	oklch(0.27 0.022 66)	#2E241B
Border / input	oklch(0.31 0.022 66)	#382E25
Destructive	oklch(0.6 0.2 27)	#DE3C37

The accent hue sits at 34 in light and shifts to 38 in dark so it stays legible against the dark ground. Neutrals are never pure grey. Every one carries chroma 0.02–0.03 at hue 66–84, which is what makes the interface read as warm plastic rather than default shadcn slate.

Brand Accents

Each streaming service keeps its own colour on the platform toggle, hardcoded in components/platform-toggle.tsx:

Service	Hex
Apple Music	#FA2D48
Spotify	#1DB954
YouTube Music	#FF0000
Beatport	#A8E00F
SoundCloud	#FF5500
Discogs	#333333
Bandcamp	#4FD1C5

The loader uses a VU-meter ramp in components/music-loader.tsx: green #3AC15A low, yellow #F5C518 upper-mid, red #E5484D at the top.

Typography

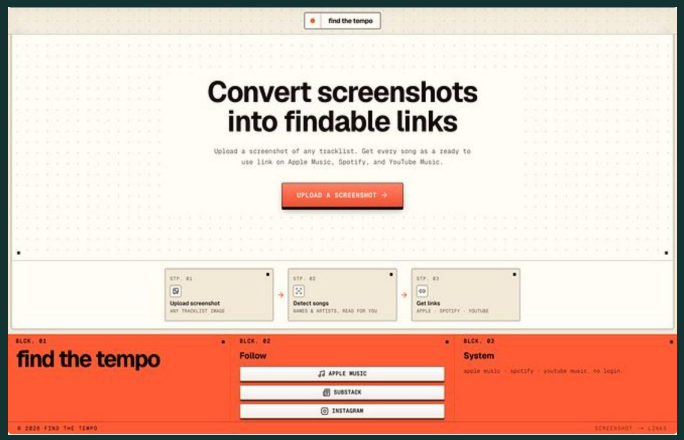
Current

Geist Sans for body and UI, Geist Mono for labels, both loaded via next/font/google in app/layout.tsx

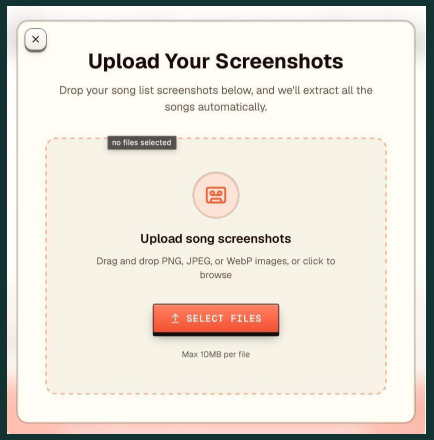
Status

Placeholder. The type system is the next design task

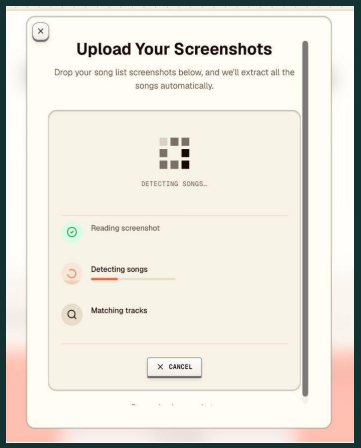
The Flow



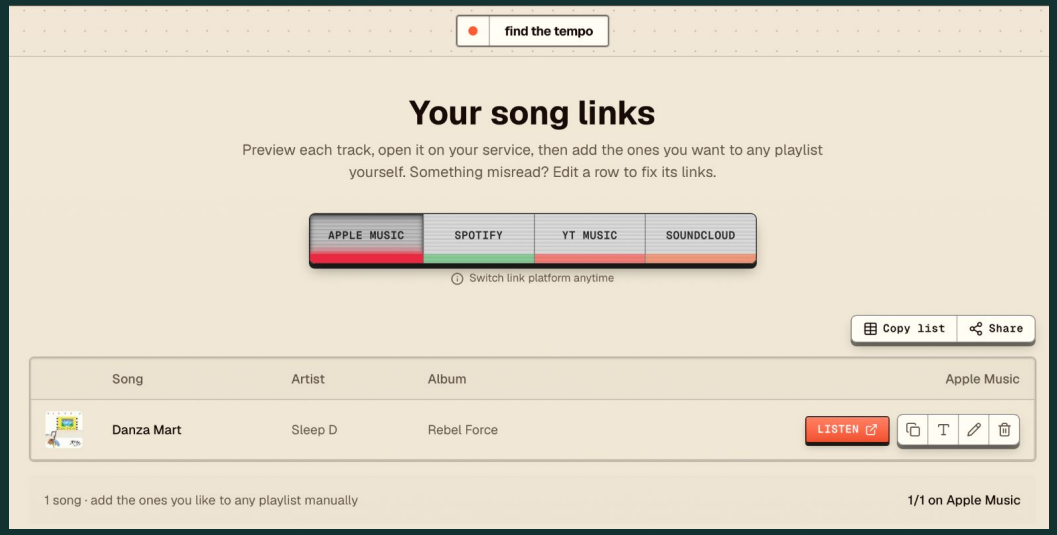
1- Landing Page



2- Upload window



3- Loading screen



4- Results page

What Happened / What's Next

- MVP live on Netlify, GA4 tracking usage
- Two bias-free surveys drafted, one for DJs and one for everyday listeners, to pick the target segment
- Open question: consumer link-collector or DJ crate-building. The surveys decide this, not my preference

Reflection

What worked

Cutting scope to the free core. The product exists and can be tested, instead of sitting half-built and expensive

What I'd change

Find a larger user base and larger volume for user surveys and interviews

Strongest criticism I expect

"The playlist-mover space already has Soundiiz and TuneMyMusic, and the DJ version runs into locked APIs and Lexicon. Why does this exist?"

My answer:

"That's why I'm surveying before building further. The MVP cost almost nothing. The research decides whether anything more gets built."