

Mock Band

Keeping guitar practice in flow

An adjustable browser backing band, developed with AI assistance and refined through human product decisions and hands-on testing.

THE PRACTICE PROBLEM

Change the music without stopping the music

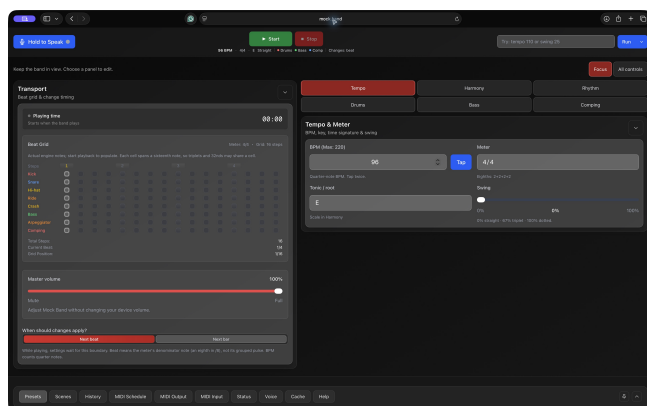
When I am jamming alone on guitar, changing a metronome or backing groove can break the moment: stop playing, reach for the laptop, change the settings, then find the idea again. With bandmates, I can call out a new key or signal a change while keeping my hands on the instrument. I wanted that kind of control when practicing alone.

I captured the idea in a Keep note on September 17, 2023: choose drums, bass and guitar, repeat a groove, then reshape it with follow-up instructions such as changing key or moving to half-time. A footswitch would activate voice listening.

I named it Mock Band and sketched logo ideas on graph paper. The player would supply the riff and musical direction; the software would supply an adjustable rhythm section. It was not an app that listened to a guitar and automatically followed its performance.

The product today

Today, Mock Band generates drums, bass and rhythm accompaniment in the browser. Players can adjust tempo, meter, key and feel, save ten local scenes, and recall contrasting sections. Visible controls work alongside text, keyboard shortcuts, supported voice recognition and compatible MIDI controllers. The current positioning is metal-first, open to everyone. The app is free to use without an account, with browser and device limits stated explicitly.



Mock Band's Focus view, with transport beside the active editor.

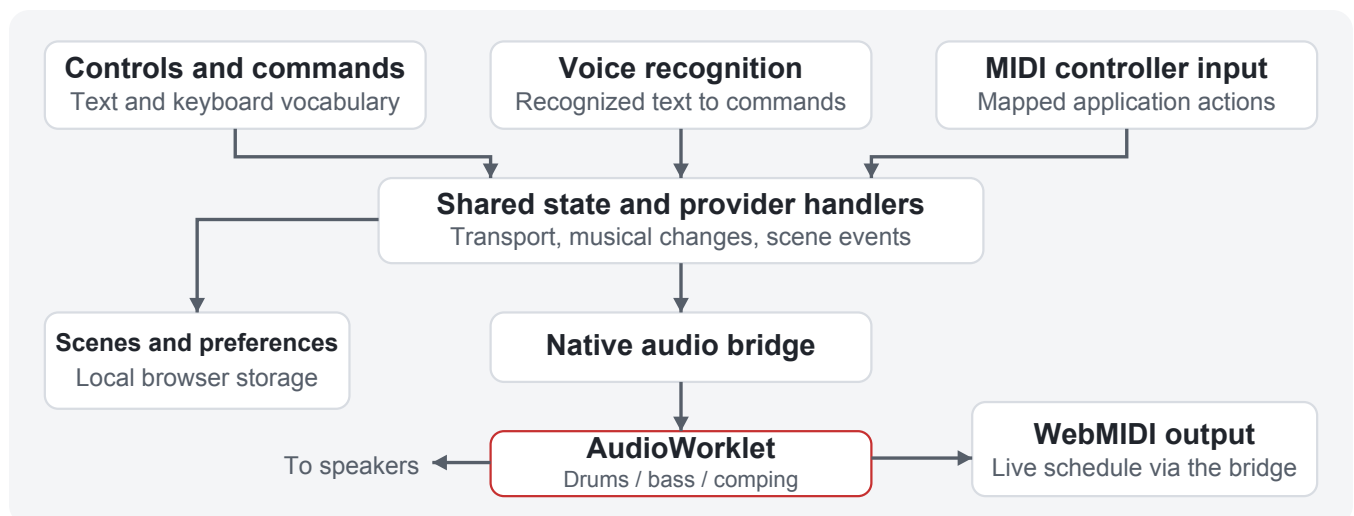
PLATFORM AND INTERACTION DECISIONS

Choosing the browser with its limits

I chose a browser application after researching WebMIDI and browser music APIs. That made a browser-based rhythm section a credible direction, while accepting uneven support for voice, MIDI and permissions. Those capabilities remain optional: a player should still be able to operate the band through visible controls or text.

The current implementation uses React and TypeScript for the interface and application state. Text and recognized speech use a shared command vocabulary; mapped MIDI actions connect to the same state and scene handlers. The audio bridge passes changes to an AudioWorklet, which renders the instrument voices and handles timing away from interface rendering. Its live schedule can also feed WebMIDI output.

Musical changes can wait for the next beat or bar instead of applying at an arbitrary point in a phrase. That adds coordination between selected, pending and applied state. Local scene storage avoids account and cloud setup, but players need JSON export/import for backup or moving configurations between browsers. Local musical state also does not imply that browser speech recognition works offline.



Observed input and playback path. Voice becomes command text; MIDI mappings invoke application actions. Shared state connects scenes, the audio bridge and worklet. The live schedule reaches MIDI output through the bridge.

Letting hardware shape the interaction

During early development, testing a purchased Bluetooth footswitch exposed a mismatch: its behavior was a toggle, not the assumed hold-to-talk gesture. I fed that finding back into Cursor and refined the interaction around a tap that starts listening while recognition is awaited. Getting it usable required debugging and repeated manual testing. Recalling scenes with the footswitch was a particularly satisfying early result.

This was a product requirement discovered by playing with real hardware. An automated test could check a handler without establishing whether the interaction made sense with both hands on a guitar.

ACCESS AND DEVELOPMENT WORKFLOW

Putting use before email capture

The original early-access gate asked for an email before people could use the app. I wanted immediate access instead. The list was not growing, and I felt capturing addresses offered little value while potentially discouraging use. The September 2026 refresh removed the gate. Updates became an optional, confirmed signup rather than the price of entry.

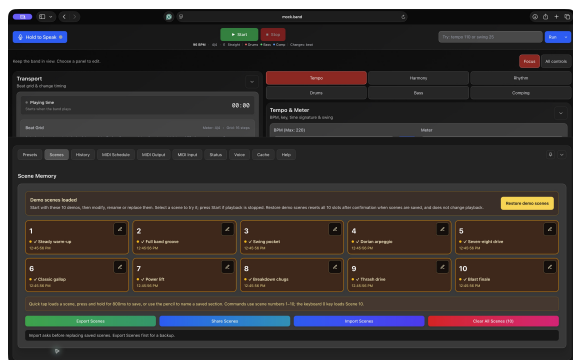
The question shifted from collecting contacts to whether visitors could start a useful session. Measurement now distinguishes opening the app, starting audio and reaching uninterrupted playback milestones. Raw commands, voice transcripts and scene contents stay out of product analytics, and delivery respects Do Not Track. These safeguards limit observation as well as collection.

From close supervision to background tasks

The first product effort took place during parental leave. The earliest preserved commit is August 2025, although local work may have preceded it. Repository history supports Cursor use and later Copilot involvement. I recall prompting the implementation rather than writing code manually. I owned the concept, platform choice, brand, requirements and acceptance testing.

Life pushed the project into the background after the initial feature work, with a later security update. I returned in September 2026 as my musical and creative activity picked up toward fall. Codex let me coordinate background tasks across implementation, automated checks, browser validation and operational setup instead of waiting for one closely supervised thread.

Parallel work also increased review pressure, and some fixes caused regressions. I required QA steps in PR descriptions and kept human merge approval as the gate. When the queue became overwhelming, I asked for related PRs and manual checks to be combined into batches. The improvement was a more manageable workflow, not a measured productivity multiplier.



Ten demo scene slots and tools for local configuration backup, import and sharing.

VALIDATION AND NEXT LEARNING

What the evidence supports

Agents handled automated unit tests and other validation; I supplied manual acceptance testing and merge decisions. The maintained release gate does not certify every legacy test, musical quality or hardware combination. I also reported a successful live MIDI output test through an IAC bus into Logic Pro, which does not establish compatibility with every physical receiver.

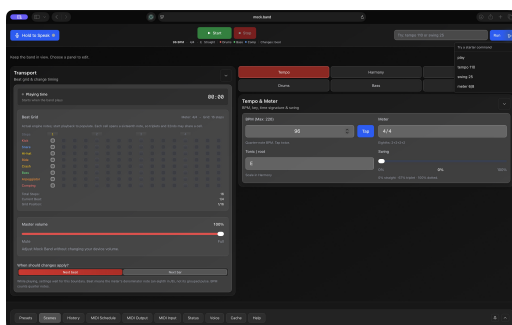
A dated October 1 analytics snapshot recorded 44 anonymous browsers entering the app, 21 starting audio and 12 reaching 60 seconds of uninterrupted playback in an ordered one-hour funnel. The UTC report starts September 22 and ends October 2, so it was still partial. Production/version filters excluded known internal browsers and QA-tagged visits. No browser reached the five-minute step in that funnel.

This is early observational playback evidence. It does not establish that visitors played guitar, improved, or became returning musicians, or that removing the email gate caused growth. Product and distribution changed together. Rekkerd coverage provides public evidence of visibility, not an endorsement or a retention result.

What remains to learn

My main lesson is that implementation capacity must be matched with manageable human review. AI can handle more work in parallel, while musical acceptance and physical-device behavior still need separate checks.

I have not yet used this current version for regular practice beyond QA. An original-riff jam-session reel and voluntary practice trials are next steps, not completed outcomes. Paid tiers, cloud scene storage, collaboration, a native mobile app, audio recording and user-facing MIDI-file export are not available. The next milestone is evidence that musicians find recurring practice value, not a longer feature list.



The starter-command menu offers examples of supported musical changes.

Public references

[Try Mock Band](#) | [Original launch account](#) | [Rekkerd coverage](#)

Prepared from an October 1, 2026 interview, reviewed source and dated measurements. Interface screenshots were captured in Safari on October 1 with playback stopped and no commands executed; they show visual behavior rather than audio or hardware validation.