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

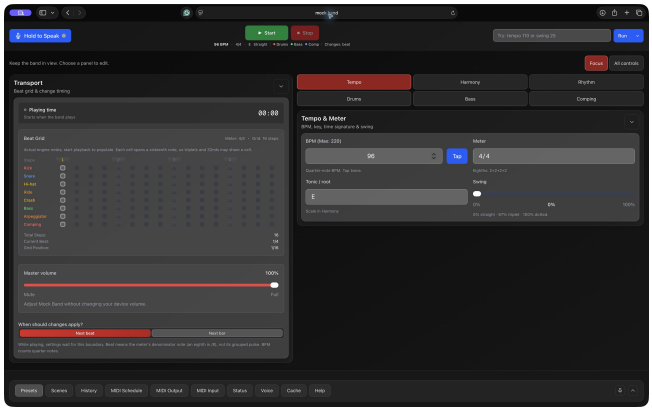
When I am jamming alone on guitar, changing a metronome or backing groove can break the moment: stop playing, reach for the laptop, change the settings, then find the idea again. With bandmates, I can call out a new key or signal a change while keeping my hands on the instrument. I wanted that kind of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

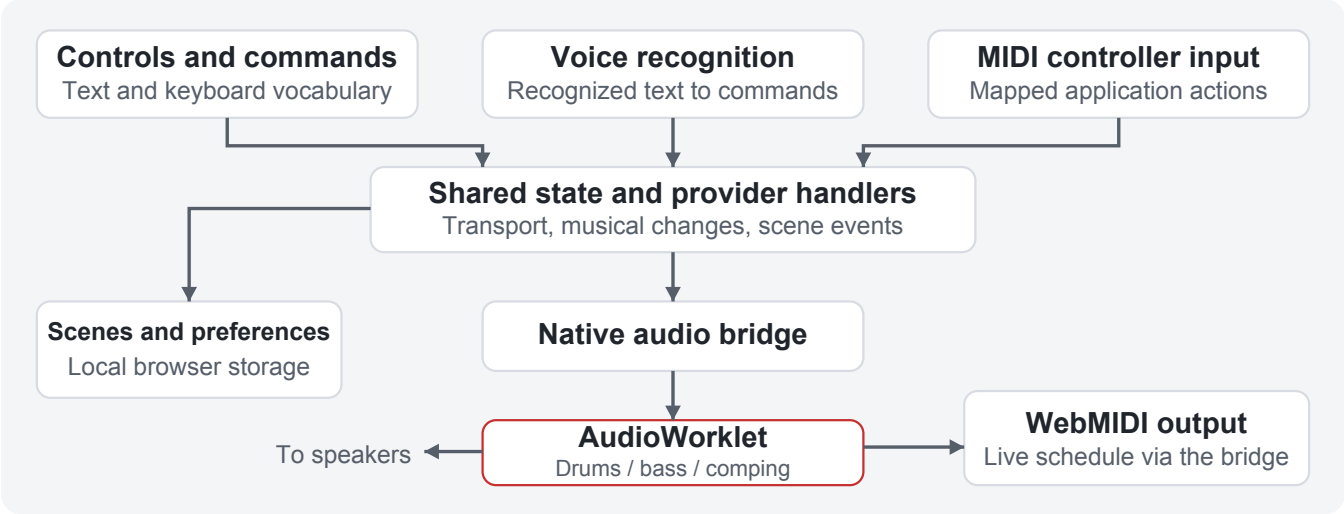
PLATFORM AND INTERACTION DECISIONS

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.

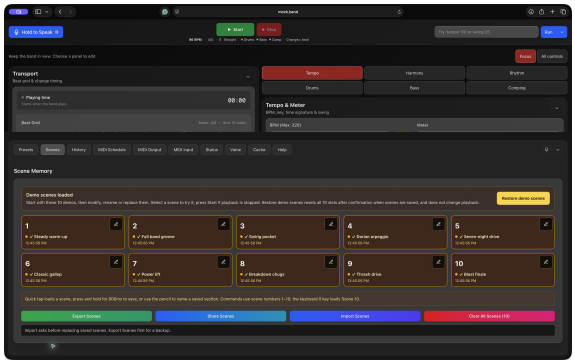
The question shifted from collecting contacts to whether visitors could start a useful session. Measurement now distinguishes opening the app, starting audio and reaching uninterrupted playback milestones. Raw commands, voice transcripts and scene contents stay out of product analytics, and delivery 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.

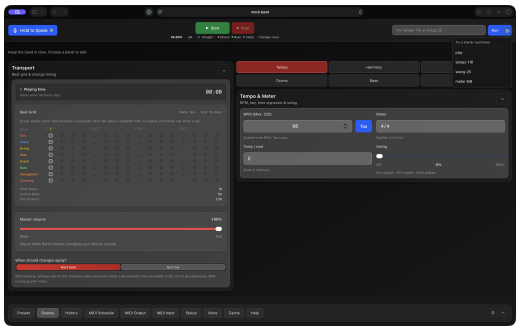
A dated October 1 analytics snapshot recorded 44 anonymous browsers entering the app, 21 starting audio and 12 reaching 60 seconds of uninterrupted playback in an ordered one-hour funnel. The UTC report starts September 22 and ends October 2, so it was still partial. Production/version filters excluded 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.

Follow-up: October 14, 2026

This dated follow-up accompanies the original October 1 edition on the preceding four pages. The original edition is preserved, including its historical evidence and limitations.

October 14 follow-up and measurement window

Mock Band is a personal project within Signal & Habit. Its primary purpose is to solve my guitar-practice problem, explore creative and technical ideas, and demonstrate engineering judgment, not generate revenue. I evaluate it through personal usefulness, technical learning and meaningful external feedback.

A fresh production query covers September 22, 2026 at 00:00:00 UTC inclusive through October 15, 2026 at 02:27:33 UTC exclusive. Results were captured by 02:29:37 UTC on October 15, or 10:29:37 PM EDT on October 14. The historical and follow-up observation windows differ and overlap; higher cumulative counts are not evidence of increased adoption or a causal improvement.

The established PostHog report counts unique anonymous actors through App Opened, Audio Started, the 60-second milestone and the 300-second milestone, in order within one hour of app entry. It retains production, exact mock.band host, analytics version 2, both known internal-browser exclusions and QA/internal source, medium and campaign exclusions. Browser actors are an approximation, not identified people, sessions or event totals. The most recent entrants can still have an incomplete one-hour conversion opportunity.

PostHog is the deployed source for these product events and pageviews. The Vercel production query confirms that Web Analytics is not enabled for this project; its SDK was removed earlier. Vercel Speed Insights remains performance telemetry, not a source for the playback funnel. Do Not Track remains respected, so this is not a census of all use.

Ordered one-hour funnel. Percentages use app-entry browsers as the denominator.

Step	October 1 snapshot	October 14 follow-up
App entry	44 (100%)	98 (100%)
Audio started	21 (47.7%)	48 (49.0%)
60 seconds uninterrupted	12 (27.3%)	23 (23.5%)
Five minutes uninterrupted	0 (0%)	2 (2.0%)

Repeat use and longer playback

A separate exploratory check uses first-observed Audio Started events per anonymous browser ID from the version-2 rollout, September 9, 2026 at 01:07:44 UTC, to the same October 15 cutoff. It finds 51 starting browser IDs and 56 browser-session groups containing 113 start events. Four browser IDs started audio in at least two analytics sessions. One of the 50 with at least 24 hours of observation had another start at least 24 hours after its first. Among the newer first-start cohort beginning September 22, the equivalent result is zero of 46 eligible browser IDs, from 47 starters. These are different populations from the ordered funnel and not a native next-calendar-day retention rate or evidence of a stable practice habit.

The existing ended-run report, over the same September 22 to October 15 window as the funnel, has three browser-session-tab groups with at least five minutes accumulated across ended runs. Its longest individual reported run is 623 seconds, or 10 minutes 23 seconds. Accumulated duration can combine separate starts and stops; it must not replace the two uninterrupted five-minute funnel completions. Missing end events leave duration unknown.

Playback milestones use the advancing audio clock rather than idle page time, excluding suspended-context time while allowing audio that continues in the background. They do not establish audible sound, an unmuted output, guitar playing, improvement or regular practice. These checks extend the interpretation of the existing instrumentation; they do not introduce a new tracking system or a claim of retention or product-market fit.

Playback and hardware corrections

On October 12, a live MIDI-output defect was corrected after I heard doubled drums in Logic even with Mock Band's local master volume at zero. The worklet refreshed the full-bar display schedule about every 100 milliseconds, and the bridge resent its notes on every snapshot. A regression reproduced 65 note-ons for 13 distinct drum-note deadlines across five refreshes in 0.501 seconds. A bounded scheduling ledger now sends each deadline once, reconciles changed future notes, preserves releases and cymbal chokes, and clears obsolete queues across Stop, restart and output changes.

The tests exercise the actual worklet and bridge against a simulated MIDI queue, including a review-discovered next-beat boundary regression. After the exact production release, I reported that it worked much better and tested it with the footswitch. That is useful founder acceptance evidence for my IAC/Logic setup, not independent participant feedback or certification of every physical MIDI receiver, hot-plug path or browser.

Starting a session and maintaining the system

On October 14, a first-practice audit found that the loaded Doom setup's advanced tempo field was below the viewport on desktop, tablet and phone. The existing tempo input is now also beside the setup's Start action, sharing the same provider state, 40-220 BPM bounds and keyboard editing behavior. The full Tempo & Meter panel, scenes, voice, MIDI and other advanced controls remain available. Load and Start stay explicit; changing tempo does not start audio.

Three-viewport checks with the real worklet verified that the shortcut and advanced panel stay synchronized, section tempo carry/reset works, and the scene bank is preserved. Six focused suites passed 146 tests, with seven release-manifest contracts, types and scoped lint checked separately. These are layout, state and engine checks, not physical iOS validation or proof that the shortcut changed visitor behavior.

Other merged maintenance removed the remaining optional email signup on October 12, excluded known crawlers from analytics, and tightened Speed Insights delivery to reviewed routes with Do Not Track and persistent QA/internal exclusions. Dependency updates addressed published fixes while retaining a tested development-only mitigation for an advisory without an upstream patch; this is not an all-alerts-cleared claim. Compatible compiler caches reduce repeated CI work without removing required verification or conservative deployment safeguards.

What I want to learn next

The concrete progress is a more usable personal MIDI setup, a smaller obstacle to selecting a comfortable practice tempo, and better-tested scheduling and privacy boundaries. External playback and published resource listings show that the project is being encountered and tried, but not whether it has become useful in someone's practice routine.

My next step is to use the current build for a real original-riff session and record what interrupts the flow. A short performance recording and later voluntary feedback can test whether the simple starting path helps another guitarist without hiding the deeper controls. The recording and completed practice trials remain pending. Personal usefulness and specific feedback should determine the next small improvement; the later feature ideas stay parked.

Maintained web version: https://paulgendek.com/projects/mock-band/