Resonance and AI Led Application Design

How an open invitation became a musical instrument through implementation feedback and testing

Brad Micallef and Sol | VisionLeadership AI Guidance
Standalone application white paper | 15 November 2026

Executive summary

An invitation to let AI choose an application produced Resonance, a constellation instrument in which connected stars become musical paths. The released Mac edition lets people draw, listen, revise, record, and export their own arrangements. Its development shows how an AI proposal becomes a usable product through human judgment and explicit checks.

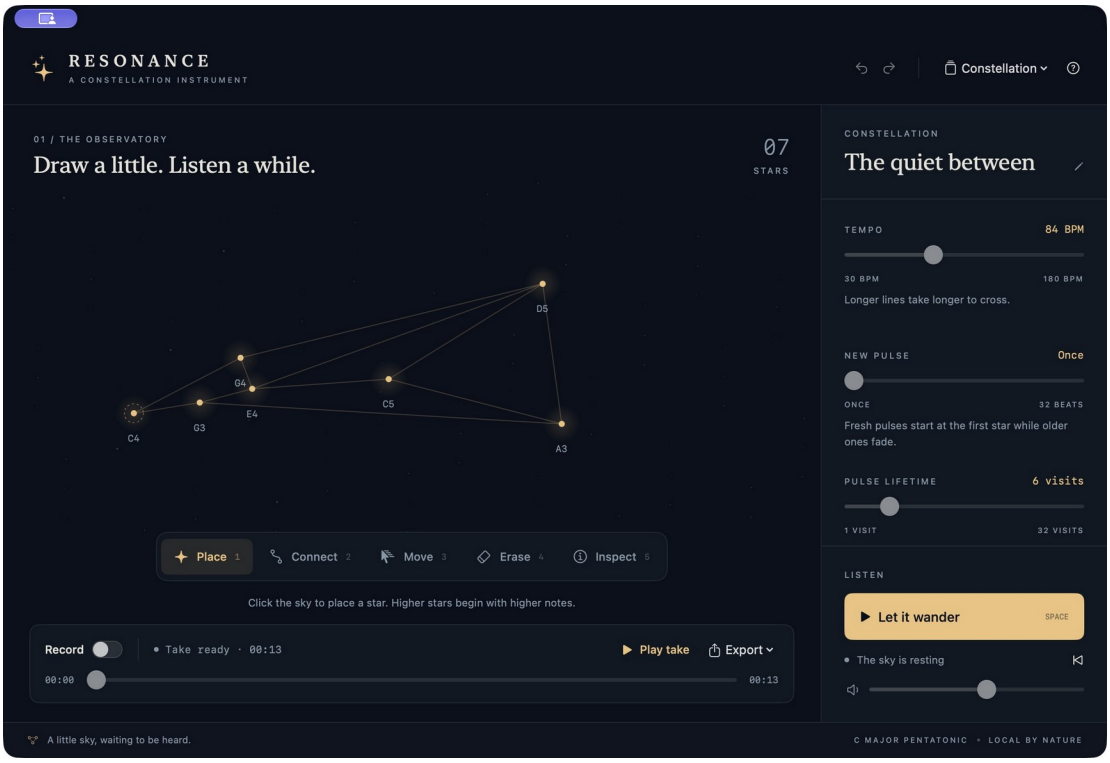

Figure 1 The released Mac interface showing the musical sky, consistent sliders, pinned playback controls, and recorded take preview. Actual application capture.

The decisive changes came from use. Brad Micallef noticed that unequal connections sounded together and that notes accumulated indefinitely. Finite pulse lifetimes and recurring launches changed the musical behavior. Later feedback reorganized the interface and completed the recording workflow. Testing then exposed runtime, touch, export, and packaging issues that appearance alone could not reveal.

The outcome is a free, Developer ID signed and Apple notarized Mac download. Mobile prototypes were exercised in simulators, but public mobile distribution is paused. This case distinguishes software correctness, an enjoyable experience, and acceptance by a distribution channel; each needs its own evidence. [1-4]

[Try Resonance at Knowledge Locker]

1 The invitation and the purpose of the application

Brad opened a second project space in the spirit of AI Wants 001 and asked the AI collaborator what it would like to build. He then authorized Resonance with the instruction to make the first version its own. This delegated the choice of concept and the first implementation, while leaving the work inside a human directed collaboration. The earlier project, Understory, provided context for exploratory software rather than a prescribed specification. [1, 9]

Inspiration expressed through the design

Resonance joins the visual language of a constellation with the behavior of a musical network. Stars are notes, connections carry pulses, and space becomes audible as travel time. The purpose is to make composition approachable through arranging and listening. A person can discover an interesting pattern without reading notation, following a score, or reaching a winning state.

This rationale is visible in the interaction itself: draw a relationship, hear its effect, then change it. The night sky supplies a readable metaphor and a calm visual setting. The graph supplies rules that can be inspected and changed. The application offers a small environment for musical inquiry rather than an attempt to replace a full recording studio.

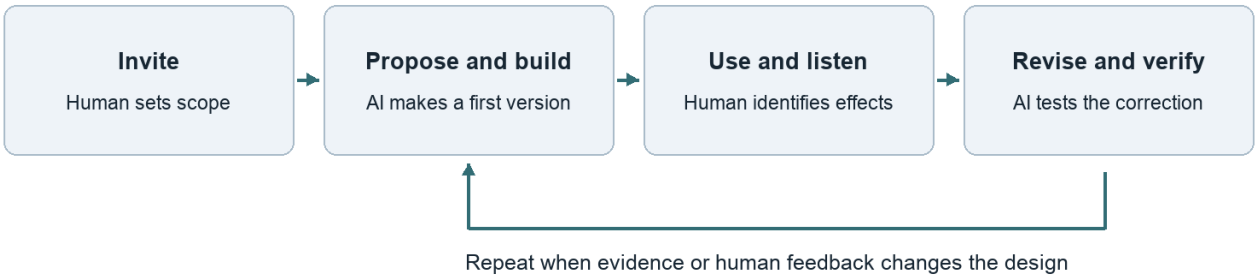

Figure 2 The development loop. AI execution remains responsive to human experience and to evidence from verification.

The scope of the claim

In this paper, AI led design means that the AI proposed the application and carried substantial design and implementation work forward. Brad selected the direction, listened to the results, supplied corrections, authorized release actions, and ultimately decided where to stop. The exercise does not demonstrate independent AI desires or a stable preference shared by other systems.

The finished application does not contain a language model. Its notes and pulses are computed locally by ordinary software. It requires no model API, account, or network connection to make music. AI describes the development process, while generative music describes how the instrument produces patterns. Keeping those meanings distinct prevents the case from claiming capabilities the product does not have. [2]

2 Turning a drawing into a musical system

The first design question was how a drawing should affect sound. The final model gives each star an identity, a normalized position, and a pitch. Connections are undirected edges. A selected starting star launches pulses; arrivals trigger notes. At a junction, pulses continue along available forward connections. At a loose end, they can reflect until their remaining visits are exhausted. [2]

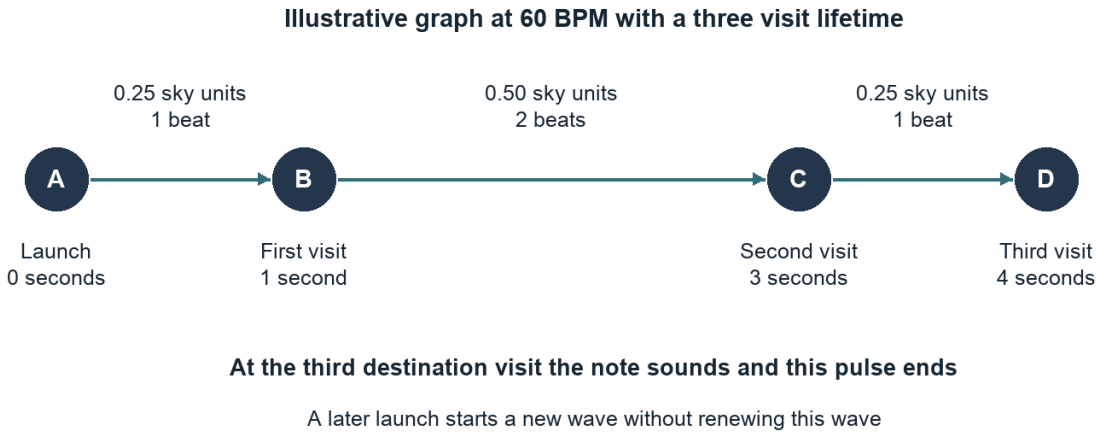

Figure 3 An illustrative path explains distance based timing and finite lifetime. These are explanatory values, not a captured performance.

Independent travel and bounded propagation

Travel time is calculated from normalized distance. A separation of one quarter of the sky takes one beat, with a minimum of one eighth of a beat. At 60 beats per minute, the one beat connection in Figure 3 takes one second. Changing tempo changes elapsed beats per second; resizing a window does not change the saved geometry or its underlying rhythm.

Each pulse carries its own elapsed time and remaining destination visits. A lifetime of three means three destination arrivals after launch. At branching points, each child inherits the reduced visit budget; it does not receive a fresh lifetime. This is a per descendant path limit, not a promise that an entire branching wave will sound only three notes.

A separate launch interval creates new waves every one to 32 beats, or Once. New launches overlap older waves without resetting them. The combination lets a constellation keep playing while individual pulses fade and disappear. A hard limit of 32 traveling pulses bounds dense graphs; excess outgoing paths are skipped, which is an audible design compromise rather than an invisible guarantee of unlimited polyphony.

What correction changed

Brad’s timing and accumulation observations were therefore behavioral requirements. The correction had to preserve different event times, terminate cycles, and permit recurrence without rejuvenating old notes. A visual animation that merely appeared to travel would not meet those requirements. Tests for unequal edge arrivals, inherited lifetime, and overlapping launches make the intended rules independently checkable. [1-3]

3 Architecture and responsibilities

The implementation uses Swift 6 and native Apple frameworks. A shared model and audio layer support a Mac interface and an adaptive iOS interface. This separates the musical rules from the platform layout, while keeping the scope small enough to inspect. There are no third party package dependencies in the app. [2]

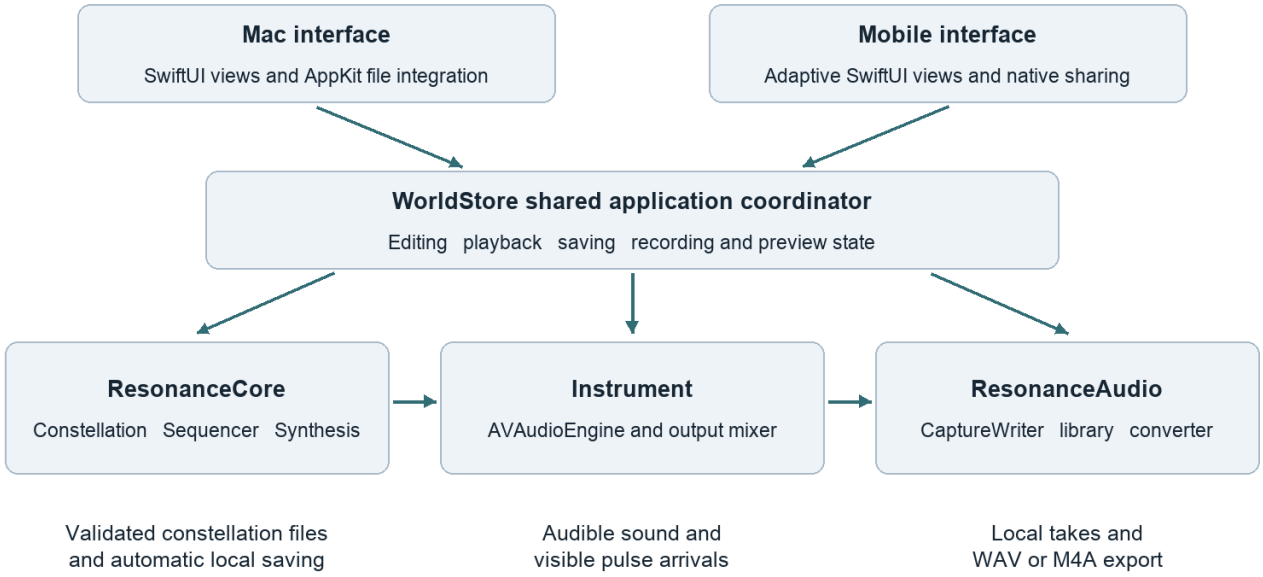

Figure 4 Responsibilities in the implementation. The mobile interface is implemented and simulator tested; the public release is the Mac edition.

The model and application coordinator

ResonanceCore contains the constellation data, validation, sequencer, and synthesis functions. The graph can be serialized and exercised without opening a window. WorldStore coordinates editing, undo, playback, saving, recording readiness, take preview, and export state. SwiftUI observes that state and renders the corresponding controls and sky.

The sound and recording path

Instrument adapts musical arrivals to AVAudioEngine. It prepares a small local bank of synthesized note buffers and plays them through a mixer and reverb. Position controls stereo panning. The audio output and pulse drawing derive from the same application events, which gives the participant a visible explanation for what is heard.

ResonanceAudio supplies the capture writer, completed take library, and file conversion. A tap on the output mixer captures Resonance's sound, including effects, without recording the microphone or other applications. Export and Music integration are explicit user actions. The Mac uses native file dialogs; mobile views use native document and sharing interfaces.

The website is a distribution and support service, not part of the musical runtime. Downloading the app involves the website; arranging a sky and producing sound do not. This boundary keeps server availability and website accounts out of the core experience. [2, 4]

4 Implementation choices and engineering tradeoffs

The design favors a responsive exploratory instrument with local storage. Some choices reduce flexibility in exchange for understandable behavior. Those limits belong in the product description as well as the code.

Decision	Implementation	Consequence
Musical vocabulary	C major pentatonic pitches from C3 to C6; Glass, Felt, and Pluck voices.	Beginners can explore a constrained palette without choosing a scale.
Bounded work	128 stars, 512 connections, and 32 traveling pulses. A pool of 48 audio players.	Memory and event growth are controlled; dense routes may be skipped and audio voices reused.
Timing	Independent beat clocks advanced by a main thread timer in common run loop modes.	Dragging and sliders continue playback; scheduling is not sample accurate.
Persistence	Validated JSON constellation files, local automatic saving, and up to 100 undo edits.	Files stay inspectable; invalid recovery data is preserved rather than silently overwritten.
Audio export	Streaming capture to disk; WAV and AAC M4A conversion.	Long takes avoid accumulating the full recording in memory; disk writes still carry timing costs.

Concurrency required a concrete correction

An iOS recording check found an audio tap closure inheriting main actor isolation. The correction made the closure explicitly Sendable and limited its captured state to the lock protected writer. The successful rerun exercised real capture inside the app process. This was a runtime correction, not a stylistic refactoring. [3]

The writer appends buffers before the audio engine can reuse them. A lock protects shared file state, but writes still occur synchronously in the callback. Moving disk work off the audio callback through a bounded queue would be a reasonable future hardening step if hardware measurements justify it; it is not part of this release.

Musical time and rendering time remain different

The sequencer records within step offsets so distinct arrivals are not collapsed into a single logical chord. WorldStore still dispatches those sounds during its timer tick. Event ordering and frame size tests establish properties of the model, not sample accurate playback under every system load. The release is appropriately positioned as an exploratory instrument rather than a production timing reference. [2-3]

5 Human feedback changed the product

Brad's feedback supplied both acceptance criteria and design direction. Several requests changed what the instrument could do; others made an existing capability understandable or reachable. The sequence shows why a first implementation is a proposal to be used, rather than a finished specification. [1]

Observation or request	Design response	Reason it matters
Unequal connections sounded together and notes kept accumulating.	Independent arrivals, visit limits, fading, and pulse bounds.	The drawing must affect timing and the graph must terminate.
New pulses should start while older notes decay.	A beat based launch interval separate from pulse lifetime.	Sustained performance no longer requires immortal notes.
Tempo, launch beats, and visits should look uniform.	Consistent slider controls and label formatting.	The controls expose related timing concepts in a common form.
The right column felt crowded.	Editable title at the top, scrolling settings, pinned playback, and Inspect beside a star.	Global controls and local editing regain clear places.
Recording should sit beneath the sky.	A recording strip linked to live playback and explicit export actions.	Recording follows the performance without taking over the settings column.
Use Ready to record and add preview and scrubbing.	Plain language readiness, Play take, and a progress slider.	Users can understand the state and judge a take before exporting.

User experience is evidence with a defined scope

After the timing and lifetime changes, Brad described a "very dramatic and organic effect on the output." That response supports the importance of the behavior change to this participant. It is not a usability score, a broad user study, or proof that every listener will prefer the result.

The UI requests also revised the information architecture. Moving the inspector near a selected star kept the context of the edit visible. Pinning playback separated a frequent action from settings that may scroll out of view. The AI's role was to translate those observations into interaction rules and then verify the resulting behavior, rather than simply add controls wherever space remained.

6 Recording made exploration worth keeping

Recording converted a temporary musical encounter into something the participant could revisit and share. Brad proposed a strip at the bottom of the sky, with recording beginning and ending alongside the resonance. Follow up requests added listening and seeking, completing a workflow from making a sound to deciding whether to keep it. [1-2]



Figure 5 The ordinary recording path. Turning recording off during playback or closing the app finishes capture immediately rather than waiting for the tail.

State clarity and protection of prior work

Ready to record describes an upcoming action rather than requiring the user to understand the term Armed. Live Play begins capture when readiness is enabled. Stop or natural completion allows five seconds for final notes and reverb. During that brief finishing state, new playback waits. These rules avoid cutting off the sound merely because its visible pulses have ended.

A completed take has its own Play action and progress slider. Pause preserves position; scrubbing can choose a point before or during playback. Preview stops the live sky and never starts recording. Starting a live performance or auditioning a star stops preview, keeping one clear source of listening at a time.

The previous completed take remains available until a replacement completes successfully. An export keeps its source alive even if a new take finishes while conversion is underway. Conversion uses a temporary destination so failure does not destroy an existing export. These choices protect user work across operations that happen at different speeds. [2-3]

File export and Music integration

Save Audio File offers 24 bit WAV or AAC M4A. Add to Music creates a permanent M4A copy and opens it with the Mac Music app. Music controls its own import and library behavior; Resonance does not promise cloud synchronization or automatic availability on another device. The latest take and constellation persist locally, while named exports let the participant keep additional work.

7 Adapting the experience to touch devices

Brad asked for iPhone and iPad compatibility because arranging stars could be enjoyable by touch. The project reused its musical core, store, and instrument while adapting layout, gesture targets, file handling, and presentation to available screen space. The mobile target supports iOS and iPadOS 17 or later. [1-3]

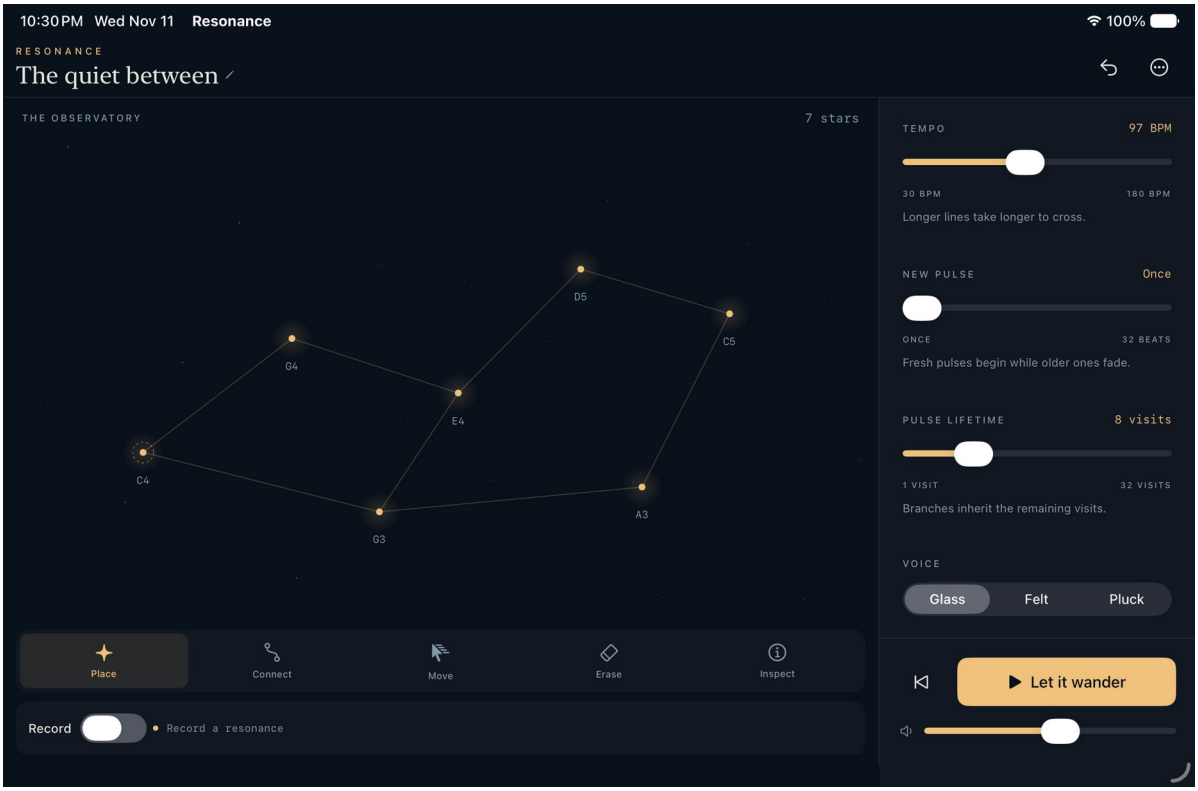

Figure 6 Actual iPad mini simulator capture from development testing. This demonstrates the adaptive interface, not current public availability of a mobile edition.

Adaptation went beyond resizing

Narrow layouts put sound settings in a sheet; wider layouts use a sidebar. Short landscape windows use compact controls. Star inspection is anchored near the selected object and positioned above or below it when side space is insufficient. Touch targets are at least 44 points. These changes retain the meaning of the controls while responding to how a person can reach them.

Simulator use found a recording switch target that did not reliably activate and an inspector that clipped against a phone edge. Both were corrected and tested again. Fold and rotation checks also examined whether recording, playback position, and the saved constellation survived changes in the available display. The tests concern software geometry and state continuity; they do not establish physical touch or Pencil feel.

The final release decision is narrower than the implemented platform range. The public product is the Mac download. Mobile distribution and a possible web edition are paused at Brad's request. Screenshots of mobile prototypes in this paper document the development process and should not be read as installation instructions for a released mobile app. [4]

8 Self testing and correction through evidence

The AI used several forms of verification because a passing build cannot establish musical behavior, export safety, or a usable interface. The record contains successful reruns and retained failures. The table records the evidence available for this case. It does not claim exhaustive quality assurance. [3-4]

Check	Evidence	What it establishes
Shared model and files	14 core tests passed.	Unequal arrivals, terminating cycles, inherited lifetime, repeat behavior, serialization, and input validation.
Recording and export	4 audio and file tests passed.	Playable stereo output, prior take restoration, destination protection, and source retention during export.
Actual Mac audio graph	Offline rendering passed for three voices, pan, decay, polyphony, reverb, and restart.	The assembled audio path produces bounded sound under the tested conditions.
Native mobile workflows	Phone and tablet simulator runs exercised editing, recording, preview, scrubbing, and Files export.	UI gestures reach working app behavior on the recorded simulator configurations.
Public release	Signature, ticket, Gatekeeper, download size, SHA 256, and ZIP integrity checks passed.	The public Mac artifact matches the verified signed and notarized package.

Failures that remain part of the evidence

On iPhone 17 Pro Max, the targeted workflow passed capture, preview, scrub, and share sheet presentation but failed to open Save to Files. Repeating the interaction with a different accessibility target did not resolve it. It is therefore not counted as a complete export pass. Earlier native exports on the tested iPhone, iPads, and Duo configurations passed.

The Duo test environment also had unreliable automated orientation commands. Direct Device Hub rotation and interactive fold cycles passed, including continued recording and take preview. The completed fold test preserved the saved constellation and produced a valid 52.8 second stereo recording. Those successes do not convert the failed automation commands into passes.

Physical devices, VoiceOver behavior, real Bluetooth and interruption scenarios, and long recordings still need hardware validation. AI written tests can also repeat an implementation’s assumptions. Brad’s listening and use introduced requirements the original checks had not captured. A credible test account retains that distinction rather than treating automation as an independent judge of every design choice.

9 Distribution changed the definition of completion

The initial release plan included the App Store. iOS version 1.5 build 10 was submitted on November 12, 2026. Apple rejected it on November 13 under Guideline 4.3(a), Design and Spam, citing similarity in binary, metadata, and or concept to other submitted apps. The message did not name a comparison app. This was a distribution decision separate from the project's technical tests. [4-5]

A useful correction to the novelty assumption

The subsequent App Store search found meaningful precedents. Senode describes connected nodes and traveling tokens that trigger notes; NodeBeat describes visual generative music with generator to note distance affecting timing. Those published descriptions establish conceptual overlap. They do not identify which app Apple considered or establish shared source code or assets. The research was a comparison of listings, not a forensic binary analysis. [6-7]

This is a process lesson for AI proposed products. An independently developed application may still occupy an established category. Comparable product research should happen before an expensive distribution push. AI choosing the idea is not evidence of market novelty, and a quiet visual identity is not sufficient differentiation by itself.

Date	Decision or result
12 November 2026	iOS 1.5 build 10 submitted with manual public release planned.
13 November 2026	Rejection reported; comparables researched; Brad chose withdrawal and free direct distribution.
13 November 2026	Mac 1.5 build 11 passed Developer ID signing, notarization, ticket, and Gatekeeper checks.
15 November 2026	Website download published and independently downloaded with matching size and checksum.
Current boundary	Public Mac edition retained; iPhone, iPad, and web distribution paused.

A complete outcome within a narrower scope

The App Store submission was canceled and its review overview showed Removed. Review monitoring stopped. Knowledge Locker now provides the free Mac download, support, privacy information, and installation guidance. Development was verified before the same tested website release was promoted to production. Temporary server access failures delayed publication but did not change the app artifact.

Apple notarization is a security check for direct Mac distribution, not App Store approval or an endorsement of the product's originality. The published release is described accordingly. Brad's decision to stop with the Mac edition makes completion explicit: a usable, accessible result can be enough without extending every possible platform. [4, 8]

10 Guidance for AI assisted application projects

Resonance suggests a practical way to organize delegated design. The human can leave room for an AI proposal while keeping authority over purpose, acceptance, distribution, and stopping. The useful unit of progress is a change whose effect can be experienced and whose relevant behavior can be checked.

Ask for a first version that can be experienced

The open invitation became informative because it produced a working instrument. Brad could listen to unequal paths, notice accumulating notes, and judge the layout. A concept description alone would not have exposed those failures. For other projects, identify the smallest interaction that lets a participant encounter the consequences of the design.

Turn observations into explicit rules

The timing complaint became independent arrival clocks. Growing note counts became finite visit budgets. Continuous performance became recurring launches that leave older waves alone. These are assessable rules, not requests for the application to feel more organic. A good revision explains the behavior being changed, the new constraint, and how it will be observed.

Check the model and the experience separately

A sequencer test can prove that a cycle ends under specified inputs. It cannot decide whether the phrase is enjoyable. A screenshot can show a tidy control; it cannot prove the export creates a playable file. Match the check to the claim, retain failure records, and include direct human use in the feedback loop.

Treat state and user work as design responsibilities

Recording, preview, export, and replacement introduced more complexity than the initial drawing canvas. Protecting the prior take and the export source mattered as much as adding a button. The same principle applies wherever a product has ongoing work, completed work, and operations that may fail or finish out of order.

Research distribution and stop deliberately

Technical progress did not guarantee App Store acceptance. Earlier comparison research would have made that risk more visible. After review, the collaboration chose a viable Mac route and paused further distribution work. The case supports iterative, bounded delegation with human judgment; it does not establish that an AI can independently certify product quality or predict a platform decision.

Try the result and form your own judgment

Visit the Knowledge Locker product page, download the free Mac edition, and open its example sky. Move a star to alter future travel distances, compare Once with recurring launches, and change the lifetime. Record a short performance, listen back, scrub, and export a take. These actions let readers evaluate the relationship between the design decisions and the experience. macOS 14 or later is required; Apple silicon and Intel are supported.

Resonance product page and free Mac download

Sources and project evidence

This is a retrospective case study of one collaboration through November 15, 2026. Brad's feedback is attributed to the project conversation; technical claims are checked against the implementation and verification records. Sol is the name used for the AI collaborator in this series. The account describes observable work and outcomes, not hidden model deliberation or a controlled comparison of AI systems.

1 Project conversation. AI Wants 002 Resonance development and release conversation. Includes the open invitation, first version authorization, timing and lifetime corrections, recurring launch request, interface and recording feedback, platform testing requests, withdrawal, direct release, and the decision to pause mobile distribution. Primary project record.

2 Implementation. Resonance Swift package and native application source. Key components: Constellation and Sequencer in ResonanceCore; Synthesis; WorldStore; Instrument in AudioEngine; CaptureWriter, RecordingLibrary, and AudioConverter in ResonanceAudio. Mac 1.5 build 11 release. Reviewed November 15, 2026.

3 Verification records. Project records in Distribution/Verification.md and Distribution/SimulatorTests/README.md, plus the 14 core and four recording tests. Simulator checks dated November 11 and 12, 2026. These records distinguish actual workflows, interactive fold checks, retained automation failures, and hardware checks still outstanding.

4 Release and review records. Distribution/Submission-status.md, AppReview-monitor.md, and Direct-release.md. Includes the later iPhone 17 Pro Max export failure, November 13 review rejection and withdrawal, notarization result, and November 15 public download checks. This record supersedes earlier preparation notes about future submission or unsigned archives.

5 Apple App Review Guidelines. Guideline 4.3 addresses spam and repeated or similar app submissions. The guideline supplies policy context; Apple's specific Resonance message is documented in reference 4. Source link

6 Senode Graph Sequencer. App Store listing describes a connected graph, note activation, emitters, traveling tokens, and probabilistic sequencing. Used as a conceptual comparison, not evidence of code identity. Source link

7 NodeBeat. App Store listing and official product site describe visual music creation and distance based generator to note timing. Used as a conceptual comparison, not an identification of Apple's comparator. Source link

8 Apple notarization. Apple Developer documentation on notarizing Mac software before distribution. Security screening and Developer ID distribution are distinct from Mac App Store review. Source link

9 Companion series paper. Brad Micallef and Sol. AI Exploration and Human Expectations. Paper 6, November 12, 2026. Examines Understory and Resonance as exploratory proposals and distinguishes design orientation from claims about AI subjective experience.

Figures 1 and 6 are application and native simulator captures. Figures 2 through 5 are explanatory diagrams prepared for this paper. The timing example is illustrative. Simulator images document development and do not imply current mobile availability.

Explore Resonance at Knowledge Locker