Applied AI Portfolio Case Studies

Jarvis for macOS, Treadway, and a practical application-preparation guide

Prepared for: Paxton Raithel

Two shipped, AI-assisted projects demonstrating product judgment, failure-driven engineering, privacy boundaries, evaluation discipline, and end-to-end ownership.

How to use this document

- For implementation or customer success roles, lead with Treadway and use Jarvis as evidence of technical troubleshooting.
- For AI evaluation or operations roles, lead with Treadway Brief and Jarvis's proposal-versus-authority safety model.
- For technical support roles, lead with the real failures that shaped Jarvis and the disciplined escalation patterns they produced.
- Use the 90-second versions in interviews; share the full case study only when a recruiter or hiring manager wants evidence.

The short positioning statement

I build applied-AI products around reliability, privacy, and clear human control. My strongest work is turning ambiguous user needs and real-world failures into tested workflows: Jarvis is a local-first macOS agent with safe computer control, while Treadway is an offline-capable planning PWA with an explainable, privacy-controlled AI handoff.

What the projects prove together

- Product ownership: problem definition, architecture, interaction design, testing, documentation, and release decisions.
- Applied AI judgment: deterministic logic where possible, model use where useful, and explicit boundaries where automation would be unsafe.
- Operational discipline: reproducible tests, honest manual-verification limits, privacy scans, deployment checks, and documented failure behavior.
- Technical communication: complex systems explained through readable architecture, case studies, evaluation protocols, and user-facing workflows.

Authorship note: both projects are openly AI-assisted and owner-directed. Product requirements, architecture constraints, privacy decisions, acceptance criteria, and final verification are human-owned.

Case Study 1 — Jarvis for macOS

A local-first voice agent designed for noisy rooms, scarce resources, unreliable permissions, and safe computer control.

Project: Jarvis for Mac on GitHub

Native presence state: local reasoning without keeping a large model or visual pipeline permanently resident.

The problem

Most desktop-assistant demos optimize for the happy path. A daily-use assistant has to share a microphone with Siri, understand imperfect speech over background television, survive macOS permission changes, protect private data, preserve system responsiveness, and avoid treating a plausible model response as a completed action.

The constraints

- No paid model API is required for the core workflow.
- The always-on path cannot keep a large model, OCR loop, or visual-analysis process resident.
- Background media and far-field speech are normal operating conditions.
- Consequential actions must remain visible, confirmable, interruptible, and verifiable.
- Financial analysis must remain decision support; live trade execution is prohibited.
- The assistant must fail clearly when permissions, applications, models, or desktop handoffs are unavailable.

The engineering response

1. Separate proposals from authority

Models can propose; deterministic code authorizes and executes. Every action candidate is normalized into a typed allowlist, grounded against the user's request, assigned a risk tier, previewed, and executed behind preconditions and postconditions. The system stops on the first unverified result and records a receipt instead of fabricating success.

2. Route by capability and cost

Common commands bypass inference. Apple Foundation Models and Qwen 4B handle lightweight local work; Qwen 9B is reserved for useful multi-step reasoning when memory, battery, and thermal headroom allow it. Deeper visual or coding work can use a desktop reasoning adapter. Models unload after idle periods.

3. Engineer speech for real rooms

The voice pipeline combines on-device transcription, a command vocabulary, whole-utterance intent recovery, bounded endpointing, transcript-stability timers, and on-demand Whisper recovery. Silence and unrecognized room audio use separate deadlines so a television cannot keep recording open indefinitely. Owner Voice uses multiple reference embeddings, while Touch ID remains the authorization boundary for protected changes.

4. Treat cleanup as correctness

The four-timeframe TradingView workflow temporarily changes application state to collect 5m, 15m, 1h, and 4h evidence. Restoring the chart to 5m with Interval and Date Range synchronization off is a transaction invariant, including partial-failure paths. The workflow fails closed when it cannot obtain a real capture.

Failures that became product features

Observed failure	Engineering response	Resulting capability
The desktop socket existed before its listener was ready.	Added bounded connection retry and a precise terminal error.	Reliable handoff behavior without infinite retry loops.
Jarvis blocked the standalone Siri wake word.	Tear down Jarvis audio, launch Siri, then resume listening.	Two assistants can coexist without permanently competing for the microphone.
Background television held recording open.	Separated transcript inactivity, hard-stop, and unrecognized-audio timers.	Bounded capture in noisy rooms.
A failed multi-step action could leave later steps running.	Added preconditions, checkpoints, stop-on-failure, receipts, and cleanup.	Transactional automation instead of optimistic click sequences.

Evidence and outcome

- Approximately 27,400 lines across the public Python, Swift, and test sources.
- A public verification record reports 186 of 186 automated tests passing across routing, privacy, memory, voice recovery, screen-capture failures, packaging, and transactional restoration.
- The public verification includes clean-clone checks, native helper compilation, privacy and token scans, wheel build, and ad-hoc hardened-signing verification.
- The system includes native voice and screen pipelines, structured SQLite FTS5 memory, tiered model routing, App Intents, owner verification, and confirmation-gated computer control.

Honest boundary: automated tests are not proof of live voice accuracy, non-owner rejection, animation smoothness, or current ScreenCaptureKit permissions. Those remain hardware-dependent checks in the published evaluation protocol. The repository does not ship a notarized public binary.

My role and judgment

I owned the product requirements, failure priorities, safety boundaries, architecture tradeoffs, hardware acceptance, and definition of done. AI coding tools accelerated implementation and review, but did not decide which actions should be allowed, which failures mattered, or what evidence was strong enough to support a public claim.

90-second interview version

“I started Jarvis as a small local voice bridge and evolved it from failures on my daily-use Mac. The central design decision was that models can propose actions but cannot authorize them. Deterministic policy grounds each proposal against the request, previews the exact plan,

executes it transactionally, and verifies the result. I also kept the always-on path lightweight by loading larger local models only when their capability is useful. Real issues—Siri microphone contention, noisy-room endpointing, stale screen permissions, and partial UI failures—became testable product requirements. The public snapshot documents 186 passing automated tests while clearly separating those checks from hardware-dependent claims. That project taught me how to turn an impressive AI demo into a system with boundaries, failure behavior, and evidence."

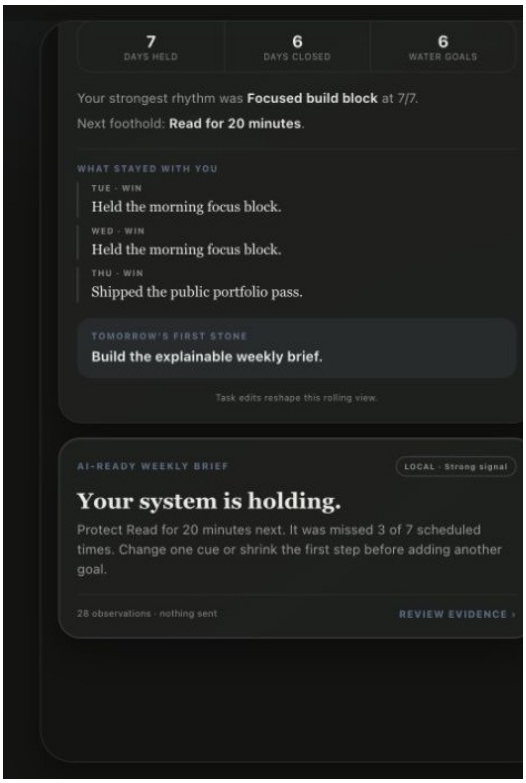
Resume-ready bullets

- Architected a local-first macOS AI agent across Swift and Python, combining on-device speech, tiered deterministic/Apple/Qwen routing, structured SQLite FTS5 memory, and native App Intents.
- Built confirmation-gated computer control with typed allowlists, transcript and plan previews, risk classification, pre/postconditions, failure isolation, receipts, and an emergency stop.
- Encoded a 186-test verification suite covering routing, privacy, memory, action safety, voice recovery, permission failures, screen capture, packaging, and transactional TradingView restoration.

Case Study 2 — Treadway

A private, offline-capable planning PWA with deterministic daily logic and an explainable AI-ready weekly brief.

Project: Treadway on GitHub



Treadway Brief exposes evidence strength and requires a deliberate user-controlled AI handoff.

The problem

Habit trackers often reduce different commitments to generic checkboxes, punish recovery, and turn accountability into surveillance. Treadway began as a personal product for building a disciplined day while preserving reflection, privacy, recovery rules, and calm interaction.

The product response

Treadway organizes the day as a path rather than a scoreboard. Users build recurring routines, track hydration, protect streaks with explicit rest and saved-day rules, complete a short end-of-day reflection, and review a rolling weekly trail. Partner accountability is intentionally narrow: encouragement and aggregate completion can be shared, while task details and journal text remain private.

The engineering response

1. Deterministic time and recurrence

Product days are anchored to America/Denver and advanced as calendar dates rather than fixed 24-hour durations. That keeps recurrence, midnight resets, reminders, and streak behavior correct through 23-hour and 25-hour daylight-saving transitions.

2. Privacy in the data model

Supabase Postgres Row Level Security is the server-side privacy boundary. Users own their routines, completions, hydration, and reflections. Partner-readable records are deliberately aggregate and exclude task titles and private text. A device PIN hides an authenticated session but is not misrepresented as encryption or server authorization.

3. Offline-first interaction

Mutations update local state immediately, persist an account-scoped cache, and queue failed writes for replay. Reconnect replays the outbox and reconciles against the server while RLS remains authoritative. This preserves phone responsiveness without confusing a local cache with permission.

4. Explainable AI without an API dependency

Treadway Brief turns seven days of product evidence into deterministic metrics, confidence labels, ranked observations, and a versioned prompt contract. Every insight exposes its evidence. Private Close text is excluded by default and can enter only one reviewed payload after an explicit opt-in. Nothing is transmitted automatically; the user chooses whether and where to paste the prompt.

Why the AI implementation matters

- The context engine is pure, deterministic, locally executed, and shared between the browser and Node evaluation.
- The prompt contract instructs a downstream model to use supplied evidence only, resist prompt injection in user text, admit weak evidence, and stay within a bounded response shape.
- Context is bounded to prevent an unexpectedly large record from becoming an expensive or unreviewable payload.
- The feature delivers value without inference cost, background model work, an API key, or provider lock-in.
- A future model can be attached behind the existing context and consent contract rather than bypassing the privacy boundary.

Product decisions shaped by real constraints

Constraint	Decision	Why it mattered
A product rename could break installed sessions and push subscriptions.	Changed visible branding while retaining cairn.surge.sh and compatibility identifiers.	Avoided a destructive migration disguised as cleanup.
Private reflections could improve coaching but increase privacy risk.	Exclude reflective prose by default; require a one-action opt-in.	Made consent specific, visible, and non-persistent.
A daily-use product must remain responsive during connectivity loss.	Use optimistic local state, an account-scoped cache, and a durable outbox.	Preserved usability without weakening server authorization.
AI features can become theater without reliable product evidence.	Built a deterministic evidence layer before any model connection.	Created useful output, inspectable grounding, and a testable integration seam.

Evidence and outcome

- Working access-gated private beta deployed as an installable PWA with cross-device sync, offline continuity, web-push reminders, export/deletion controls, themes, and accessibility support.
- The current web verification passes syntax, PWA invariants, privacy boundaries, and Treadway Brief evaluation for grounding, confidence, determinism, private-text controls, and malformed inputs.
- The Foundation-only native domain package currently passes 62 Swift tests and a separate 57-check verification harness covering recurrence, DST, hydration, streaks, notifications, merging, and archive behavior.
- The architecture includes vanilla JavaScript, a service worker, local caching/outbox behavior, Supabase Auth, Postgres RLS, realtime synchronization, and Edge Functions for reminders and account deletion.

Honest boundary: Treadway is a private beta, not a commercially launch-ready service. Real-iPhone gesture feel, safe areas, VoiceOver, and end-to-end push delivery remain device checks. The native Apple track is a tested prototype and is not represented as feature-parity with the PWA.

My role and judgment

I defined the user problem, product behavior, design direction, privacy boundaries, acceptance criteria, and release decisions. The project demonstrates that AI product engineering includes deciding what not to automate, preserving user control, and separating repeatable evidence from attractive but unsupported claims.

90-second interview version

“I built Treadway because most habit trackers treat every commitment like the same checkbox and often make accountability invasive. I designed it as an offline-capable daily path with deterministic Mountain Time recurrence, Supabase Row Level Security, narrow partner visibility, and a durable write outbox. The most relevant AI feature is Treadway Brief. Instead of adding a chatbot, I built a deterministic seven-day context engine that shows evidence and confidence before preparing a versioned prompt. Private reflections stay out by default, and nothing is sent automatically. The current web verification passes, while the native domain package passes 62 tests and a 57-check harness. The main lesson was that useful AI implementation starts with product state, grounding, consent, and evaluation—not with choosing the largest model.”

Resume-ready bullets

- Built and deployed an installable planning PWA with deterministic recurrence, offline write recovery, cross-device Supabase synchronization, owner-scoped Row Level Security, and scheduled web-push reminders.
- Developed a locally executed, vendor-neutral context engine that turns seven-day product data into evidence-backed insights, confidence labels, and a versioned model prompt without automatic data transmission.
- Verified the production web path plus a Foundation-only native domain package covering recurrence, DST transitions, hydration, streaks, notification planning, synchronization merging, and archive behavior.

Application Preparation Guide

Learn enough to speak clearly, produce evidence, and apply—without postponing applications for unnecessary credentials.

Priority rule

Apply while preparing. The strongest roles reward clear judgment, careful written feedback, implementation thinking, and technical curiosity. Advanced machine learning, cloud certifications, Docker, and Kubernetes are not prerequisites for the first application wave.

A focused 48-hour plan

1. **Block 1 —** API, JSON, and webhook vocabulary (90 minutes). Learn GET versus POST, request and response JSON, authentication, webhooks, status codes, and the difference between a user error, integration error, and server error.
2. **Block 2 —** SaaS implementation lifecycle (75 minutes). Practice explaining discovery, requirements, configuration, data validation, UAT, training, go-live, hypercare, and handoff as one connected process.
3. **Block 3 —** AI evaluation practice (60 minutes). Score one response for accuracy, coverage, instruction adherence, evidence grounding, and clarity; then write specific feedback that identifies the exact failure.
4. **Block 4 —** Support and escalation writing (45 minutes). Convert one Jarvis failure into a professional ticket containing context, reproduction steps, expected behavior, actual behavior, impact, evidence, workaround, and priority.
5. **Block 5 —** Healthcare workflow basics (45 minutes). Learn what an EHR is, how referrals and intake move through a practice, what PHI means, and how HIPAA's minimum-necessary principle affects access and escalation.
6. **Block 6 —** Interview rehearsal and tailoring (90 minutes). Record the two case-study explanations, prepare three STAR stories, and tailor the top third of the resume for the role being submitted.

1. API, JSON, and webhook essentials

- API: a defined interface that lets systems exchange requests and responses.
- JSON: a structured text format made of objects, arrays, keys, and values.
- GET: retrieve data without intending to change server state.
- POST: submit data or request creation/action; exact semantics depend on the API.
- Authentication: proves who or what is making a request; common forms include API keys, sessions, OAuth, and signed tokens.
- Webhook: an outbound event notification sent to another system when something happens.

- Useful status codes: 200 success, 201 created, 400 invalid request, 401 unauthenticated, 403 unauthorized, 404 not found, 409 conflict, 429 rate limited, and 500 server failure.

Mini exercise

Explain this scenario aloud: “A customer says completed tasks are not syncing. I confirm the account and affected records, inspect the request payload and response code, distinguish a local cache issue from an authorization or server issue, capture reproducible evidence, provide a safe workaround if one exists, and escalate with customer impact and priority.”

2. SaaS implementation plan template

- **Discovery:** Identify the customer goal, current workflow, stakeholders, constraints, success criteria, and decision owner.
- **Requirements:** Separate required behavior from preferences; record dependencies, risks, and missing information.
- **Configuration:** Map the workflow into permissions, routing rules, templates, integrations, and account settings.
- **Validation and UAT:** Test happy paths, edge cases, data quality, permissions, and expected-versus-actual behavior with customer representatives.
- **Training:** Teach the workflow by role, provide a short guide, and confirm users can complete the critical path.
- **Go-live:** Use a readiness checklist, obtain sign-off, monitor the launch, and communicate status and risks.
- **Hypercare and handoff:** Triage launch issues quickly, document final configuration, transfer open items, and define the ongoing owner.

Practice deliverable

Write a one-page implementation plan for a fictional clinic adopting Medsender. Include the clinic’s referral workflow, role-based permissions, CSV/data validation, UAT cases, training audience, go-live criteria, and the escalation path for any issue involving sensitive patient information.

3. AI evaluation practice

Use a 1–5 rubric for each dimension: accuracy, coverage, instruction adherence, grounding, and clarity. A useful evaluator cites the exact source conflict or omission rather than saying an answer is “bad” or “confusing.”

Mock evaluation

Source fact: Treadway excludes private Close text by default. It can enter one prompt only after a visible, user-controlled opt-in, and the product does not transmit the prompt automatically.

AI output: Treadway continuously sends journal entries to an AI coach so it can personalize advice in the background.

Strong feedback: “The output contradicts the source in three places: it changes default exclusion into inclusion, one-time opt-in into continuous processing, and user-initiated copy into background transmission. Accuracy and grounding should score 1/5. A corrected answer should state that private text is excluded unless explicitly included for a single reviewed payload, and that no automatic transmission occurs.”

4. Support-ticket and escalation template

- **Context:** Customer, environment, affected workflow, and when the issue began.
- **Steps to reproduce:** The smallest repeatable sequence that causes the failure.
- **Expected behavior:** What the product or documented workflow says should happen.
- **Actual behavior:** What was observed, including exact error text when available.
- **Impact:** How many users or workflows are affected and whether work is blocked.
- **Evidence:** Screenshots, logs, timestamps, request IDs, browser/app version, and tests already performed.
- **Workaround:** A safe temporary path, or a clear statement that no safe workaround is known.
- **Priority and owner:** Recommended severity, responsible team, and the next update time.

5. Healthcare basics for entry-level roles

- EHR: an electronic health record system containing clinical and administrative patient information.
- Referral workflow: request received, patient/insurance information validated, urgency triaged, destination selected, status tracked, appointment completed, and the outcome communicated.
- PHI: protected health information that can identify a patient and relates to health, care, or payment.
- Minimum necessary: access and disclose only the information needed to perform the authorized task.
- Auditability: sensitive actions should be attributable to a user, time, and purpose.
- Safe escalation: avoid copying sensitive data into an unnecessary channel; provide the minimum reproducible context and follow the company’s approved secure process.

6. Three interview stories to prepare

Story A — Turning a flaky failure into a reliable workflow

Situation: macOS could show Screen Recording enabled while Jarvis's real capture still failed. **Task:** Make capture truthful and reliable. **Action:** Treated successful ScreenCaptureKit output—not permission state—as the acceptance check, stabilized the installed identity, suppressed repeated prompts, and surfaced explicit pre-handoff failures. **Result:** The failure path became predictable and repeatable in tests, with live capture retained as a manual hardware check.

Story B — Preserving users through a product rename

Situation: Cairn identifiers already underpinned sessions, PWA installs, redirects, and push subscriptions when the visible brand changed to Treadway. **Task:** Rebrand without breaking existing users. **Action:** Changed presentation while retaining the production origin, configuration, storage namespace, and migration-sensitive identifiers. **Result:** The product gained a stronger identity without a destructive migration.

Story C — Adding AI without weakening privacy

Situation: Treadway had useful weekly evidence but no privacy-safe AI layer. **Task:** Create AI value without an API bill or silent data export. **Action:** Built a deterministic context engine with provenance, confidence, bounded context, injection defenses, private-text exclusion, and a reviewed copy action. **Result:** Users receive local analysis and an optional provider-neutral prompt while the integration remains testable and user-controlled.

7. What to learn now versus later

- **Learn now:** implementation lifecycle, support escalation, AI evaluation rubrics, JSON/API/webhook basics, EHR/referral/HIPAA vocabulary, and concise demonstrations.
- **Learn after submitting:** SQL fundamentals, deeper Supabase/Postgres operations, Docker/Linux, cloud observability, evaluation datasets, and basic statistics.
- **Do not delay applications for:** advanced ML mathematics, Kubernetes, expensive certifications, a third major portfolio project, or invented professional experience.

Final application checklist

- Choose the case study that matches the role instead of attaching both automatically.
- Tailor the first third of the resume and the first paragraph of every written response.
- Include one public repository link and one precise claim the reviewer can verify quickly.
- Remove claims that depend on private hardware behavior without dated evidence; rehearse each explanation conversationally, then apply before adding another feature.