SAMPLE REPORT: AUDIT OF THIS SITE, NOT A CLIENT

AI Release-Readiness Audit

Sage Atlas (this site): its chatbot and scoping assistant, eval engine, labs, tests and release path



Defined, not yet gated.

AUDIT DATE
2026-11-21

AUDITOR
Jason Teixeira

EDITION
Sample 1.0

READ ONLINE
sageideas.dev/sample-report

Read this first. This is a real audit of this website's own AI and QA surfaces, published as a sample. It is not a client engagement and no client appears in it. Every fact cites a file in the repository (Appendix B). Its author built the system, a bias Appendix C states plainly.

01 · EXECUTIVE SUMMARY

Ready to demo. Not yet gated.

The deterministic site is gated well and honest about its limits. The AI behaviour on top of it is not gated at all: a prompt or model change reaches visitors with no eval in between. Three High findings, two of them a day's work, one a real project.

1.9/4	10	6
overall: Defined	findings: 3 High, 5 Medium, 2 Low	fixes sized S, part of a day each

Fix first

R-01 HIGH S
396 API tests sit outside the release gate

R-02 HIGH S
The headline eval result has no receipt you can re-run

R-03 HIGH L
No gate checks model behaviour when a prompt or model changes

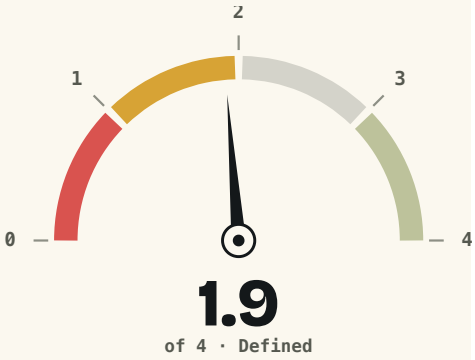
What held up

- Limits are published next to the numbers
- Server-side allowlists, not trust
- Cost and abuse are bounded
- The scorecard cannot go green by accident

02 · SCORECARD

Eight dimensions, scored against a rubric.

Levels are an assessed judgement against the rubric below, not a measurement. Each score carries its reason and the facts behind it.



Eval coverage	2/4 DEFINED	Three surfaces have defined checks. Committed eval data in this repo covers scoping mode only, and nothing maps failure modes to cases. F08 F09 F11
Golden sets	2/4 DEFINED	Sets exist, are versioned and are published with their limits. They hold 4, 7 and 14 cases, which the site itself calls a start. F09 F20
Release gates	2/4 DEFINED	The deterministic gate is real and honest. No gate looks at model behaviour, and the API tests sit outside it. F01 F18 F19 F08
Regression	2/4 DEFINED	The pattern is proven in a labelled simulation. The real API tests pass when run, but nothing runs them on a change. F03 F12
Observability	1/4 AD HOC	Error reporting is dormant until a DSN is set, and the chat handler records no conversation or quality signal. F13
Guardrails	3/4 AUTOMATED	Layered and enforced server-side: caps, allowlists, rate and budget limits, a scripted fallback. One mode skips the price backstop. F04 F05 F06
Rollout and rollback	1/4 AD HOC	Prompts ship with code and the model is an environment variable. No flag, canary or written rollback path exists yet. F07 F14
Data handling	2/4 DEFINED	Inputs are bounded and error reports are scrubbed. The privacy page does not name the model provider that receives chat messages. F04 F13 F15

0 ABSENT	1 AD HOC	2 DEFINED	3 AUTOMATED	4 GATED AND WATCHED
Nothing to inspect.	Exists, but only as a one-off or in someone's head. Not repeatable from the repo.	Written down and runnable, but run by hand or not on every change.	Runs on every change, fails the build, and someone is told.	Automated, blocks the release, and production feedback flows back into it.

03 · FINDINGS

Ranked by severity, then by size of fix.

High: A claim or protection the site relies on can fail unnoticed. **Medium**: A real gap with a contained blast radius, or a gap that blocks a High from being closed. **Low**: Hygiene. Cheap to fix, and worth fixing before it erodes trust in the numbers. Fix size is relative effort, never money. **S**: Part of a day for one engineer. **M**: A few days. **L**: About a week or more, or work that needs a decision first.

R-01 HIGH S — REGRESSION

396 API tests sit outside the release gate

The 80 files in tests/unit-api cover the chat guardrails, rate limiting, price scrubbing, the scoping evals and the grader. The release command and CI run a different set, and the proof scorecard counts none of them.

EVIDENCE

- `npm run verify` runs `npm test`, which globs `tests/*.test.mjs`. `test:api` is a separate script. CI runs only `npm run verify`. F01
- The scorecard lists 52 files and 541 tests; none are in tests/unit-api. F02 F03
- Run by hand during this audit, `npm run test:api` reported 396 passed, 0 failed. F03

IMPACT

The code that enforces the guardrails can regress with a green build and a green public scorecard.

FIX

Add `test:api` to `verify`, include its files in the scorecard run, and republish the totals.

R-02 HIGH S — GOLDEN SETS

The headline eval result has no receipt you can re-run

The case page leads with a live chatbot eval that passed 11 of 11. The page also says the run artifact is not yet committed. Nothing in this repo re-runs that eval.

EVIDENCE

- The 4-case golden set, 7-case red-team set and the run script live in a different repository (jt-portfolio). In this repo, evals/ holds only the 14-case scoping set. F09
- The limitation is disclosed on the case page itself, which is the honest part: the gap is known and stated. F09 F20

IMPACT

A reader cannot verify the number, so it works as an assertion. For a page that sells proof, that is the wrong place to be soft.

FIX

Commit the run artifact with its date, commit, model name and judge settings, and link it from the case page. Until then, keep the limitation line.

R-03 HIGH L

RELEASE GATES

No gate checks model behaviour when a prompt or model changes

The personas are constants in the handler and the model is an environment variable. The only live evals are run by an operator, by hand. A prompt edit or a model swap reaches visitors with no eval between the change and production.

EVIDENCE

- Prompts are string constants in api/chat.js; LLM_MODEL, LLM_BASE_URL and LLM_API_KEY are read from the environment. F07
- The scoping eval documents itself as an operator gate, "NOT a CI gate"; its CI test only checks the golden data is well formed. F08

IMPACT

The most likely regression on an LLM surface, a quiet behaviour change after an edit, is the one nothing is positioned to catch.

FIX

Two steps. First a recorded-replay gate on prompt diffs, deterministic and cheap. Then a scheduled live eval with a floor that only moves up.

R-04 MEDIUM S

GUARDRAILS

The price backstop is skipped in one mode

Every reply passes a server-side filter that strips dollar figures, except in the automations mode, which returns the raw model text. A code comment says that mode quotes fixed setup prices; its prompt now says not to.

EVIDENCE

- The handler routes automations replies around `sanitizeScopeReply`; all other modes pass through it. F06
- The standing rule is that no service price appears anywhere on the site, enforced by a test over pages, content and API prompts. F06

IMPACT

The protection that makes the no-price rule hold on every other surface does not cover this one, and the comment misleads the next reader.

FIX

Send every mode through the backstop and delete the stale comment. Add a test that asserts a priced model reply is scrubbed in each mode.

R-05 MEDIUM S

DATA HANDLING

The privacy page does not name the model provider

Visitor messages are forwarded to whichever provider the environment points at. The privacy page lists two processors and the model provider is not one of them.

EVIDENCE

- privacy.html section 4 names Vercel and Resend. The chat handler sends up to 16 messages of 800 characters to the LLM endpoint. The CSP also permits a Supabase project and Cal.com. F15 F04

IMPACT

People type real business detail into a chat box. They are entitled to know who processes it.

FIX

Name each processor the code calls, what it receives and why. Add a test that compares the privacy list with the hosts the code and the CSP allow.

R-06 MEDIUM M EVAL COVERAGE

The golden sets are too small for the claims made on them

Four, seven and fourteen cases are a start. The site says so. What is missing is the map from failure mode to case that shows where the holes are.

EVIDENCE

- The case page limits a pass to "those cases passed, nothing more" and calls the sets "a start, not coverage". F09 F20
- Committed eval data in this repo covers scoping mode only; the other assistant modes have none here. F08 F09
- The live probe engine runs 7 probes against one deliberately naive target. F11

IMPACT A pass cannot be read as safety. The sets also cannot say which failure modes have no case at all.

FIX Write the failure-mode map first (what can go wrong per mode), then grow each set to cover the map, with real transcripts once consent and logging exist (R-07).

R-07 MEDIUM M OBSERVABILITY

The chat handler produces no production signal

Nothing the handler does is recorded for review. Error reporting is off until a DSN is set. The published next step, growing the golden sets from real conversations, depends on records this handler never writes.

EVIDENCE

- The handler logs only provider and parse errors to the console and persists no conversation. F13
- Sentry reporting is dormant until SENTRY_DSN is set. /api/health reports only which subsystems are configured. F13

IMPACT No one can see refusal rate, fallback rate, budget exhaustion or how often the price backstop fires, so drift is invisible until a visitor reports it.

FIX Emit four counters (fallback, budget exceeded, scrub hit, refusal) with no message content. Add consented, redacted transcript capture only after R-05 is closed.

R-08 MEDIUM M ROLLOUT AND ROLLBACK

No rollout or rollback path for prompts and the model

A prompt change is a code deploy and a model change is an environment edit. Neither is behind a flag, neither has a canary, and no rollback document exists.

EVIDENCE

- Prompts ship with the code and the model is read from the environment. F07
- The roadmap lists a rollback plan as unfinished Phase 7 work. No runbook exists in docs/. F14

IMPACT When a change goes wrong, recovery is whatever someone works out under pressure.

FIX Version each prompt, put it behind a flag with a percentage rollout, and write a one-page rollback runbook that names who does what and how to verify it worked.

R-09 LOW S

RELEASE GATES

Numbers drift, and the scorecard is stamped with an uncommitted tree

The roadmap still says 224 tests; the scorecard says 541. The scorecard commit reads "db2c228+uncommitted", so the receipt cannot be reproduced from a commit.

EVIDENCE

- docs/ROADMAP.md reports 224 tests for Phase 0; the scorecard reports 541. F16 F02
- The roadmap's own "fail the build on drift" script is not built. F16

IMPACT

Small stale numbers teach readers to discount the true ones.

FIX

Generate quoted numbers from the scorecard at build time and make the proof run refuse a dirty tree.

R-10 LOW S

REGRESSION

A recorded performance regression is missing from its own summary

The quality report records a desktop layout-shift score of 0.549 on one lab after the fixes, then leaves that row out of its "still under 95" list.

EVIDENCE

- The table shows /lab/concierge-trace desktop at performance 78 and CLS 0.549 after the fixes. The summary section does not list it. The figures date from the report and were not re-measured here. F17

IMPACT

A report that omits its own worst row is easy to doubt, even when the omission is an oversight.

FIX

Re-measure, then either fix the shift or list the row with its cause.

What to do, in what order.

Sequenced so the cheap fixes land first and the first real behaviour gate is built on trustworthy numbers.

01

Days 0 to 30

Close the gaps that are small and cheap

R-01	Put the API tests in the release command and the scorecard	S
R-02	Commit the live eval run artifact and link it from the case page	S
R-04	Route every mode through the price backstop; add the test	S
R-05	Name every processor on the privacy page	S
R-09	Generate quoted numbers from the scorecard; refuse a dirty tree	S
R-10	Re-measure the lab and fix or list the row	S

02

Days 31 to 60

Build the first real behaviour gate

R-03	Recorded-replay gate on prompt diffs, in CI	M
R-06	Failure-mode map, then grow each set to cover it	M
R-07	Four content-free counters from the chat handler	M

03

Days 61 to 90

Make change safe, then keep it watched

R-03	Scheduled live eval with a floor that only moves up	L
R-08	Versioned prompts behind a flag, plus a written rollback runbook	M
Review	Re-score this card. Target: gates and regression at level 3, rollout at 2	S

What held up

Limits are published next to the numbers

The case page says the sets are small, that a pass means only those cases, and that the live-run artifact is not yet committed.

Server-side allowlists, not trust

Unknown modes get a 400, and the scoping assistant can only select capability keys that exist, enforced after the model answers.

Cost and abuse are bounded

Turn, length, output, per-IP and daily limits are in place, and the failure path is a scripted fallback instead of an error.

The scorecard cannot go green by accident

It is written only when the suite passes and at least one test ran.

CI cannot spend money by accident

The runner is hardcoded to self-hosted and a test fails the build on any other.

Simulations say they are simulations

The eval-gate lab states that no model is called and labels its starting floor illustrative.

APPENDIX A · METHOD

How this audit was done.

In scope

- The chat and scoping assistant (api/chat.js, assets/scope-core.mjs)
- The live eval engine (api/eval.js) and the eval-gate lab (src/lab/eval-gate)
- The test suites, CI workflow and proof scorecard
- Error reporting, health probe and rate limiting (lib/observe.mjs, api/health.js, lib/ratelimit.mjs)
- The privacy page and the response headers (privacy.html, vercel.json)
- The roadmap and quality report (docs/ROADMAP.md, docs/QUALITY-REPORT.md)

How

- Read-only review of the repository as it stood on the audit date. Files are cited by path in the evidence index.
- One command was executed: `npm run test:api` (396 passed, 0 failed). Everything else is read, not run.
- Every finding states the evidence, the impact, a fix and a relative size. Levels are a stated judgement against the rubric, not a measurement.

Not examined

- Production traffic, logs or provider dashboards. None were available, so no claim here describes live behaviour.
- Live model quality. No evals were re-run against a model for this report.
- The provider's own data handling, infrastructure security or load behaviour.
- Anything outside this repository, including the separate jt-portfolio repository that holds the chatbot eval sets.

> **Independence**
>
> This audit was written by the author of the system it audits. A client audit is done by someone who did not build the system, which removes that bias. The findings here are real and checkable, and they are also the findings the author is most likely to have under-weighted.

APPENDIX B · EVIDENCE INDEX

Every fact, and the files behind it.

Fact	What it says	Source
F01	`npm run verify` builds the site and runs `npm test`, which is `node --test tests/*.test.mjs`. The API tests are a separate script, `test:api`. CI runs only `npm run verify`.	`package.json` `.github/workflows/ci.yml`
F02	The proof scorecard records 541 tests, 540 passed, 0 failed, 1 skipped, across 52 test files, generated 2026-11-21T23:59:22Z for commit "db2c228+uncommitted" with command `SITE_ROOT=dist node --test tests/*.test.mjs`.	`public/proof/scorecard.json`
F03	tests/unit-api holds 80 test files and none of them appear in the scorecard. Running `npm run test:api` during this audit (2026-11-21) reported 396 tests, 396 passed, 0 failed.	`tests/unit-api` `package.json` `public/proof/scorecard.json`
F04	The chat handler bounds every request: 16 turns, 800 characters per message, a per-mode output cap (220, 220, 500 and 280 tokens), 20 requests a minute per IP, a global daily ceiling of 1000 (CHAT_DAILY_MAX), and a 501 that sends the client to a scripted fallback when the model is unconfigured or over budget.	`api/chat.js` `lib/ratelimit.mjs`
F05	The handler rejects unknown or non-string modes with a 400, and in scoping mode keeps only capability keys that exist in the rate card (filterSelection against RATE_CARD) before anything reaches the visitor.	`api/chat.js` `assets/scope-core.mjs`
F06	A server-side backstop (sanitizeScopeReply) strips dollar figures from every reply in every mode except automations, which returns the raw model text; the code comment says that mode "intentionally quotes fixed setup prices" while its prompt now says not to quote a build price. The site-wide rule is that no service price is published.	`api/chat.js` `tests/no-service-prices.test.mjs`
F07	The assistant personas are string constants inside api/chat.js and ship with the code. The model, provider endpoint and key come from the environment variables LLM_MODEL, LLM_BASE_URL and LLM_API_KEY.	`api/chat.js`
F08	The scoping-mode eval (scripts/eval-scope.mjs, 14 golden cases in evals/scope-golden.json) is documented as an operator gate run by hand, "NOT a CI gate". Its CI-side test only checks that the golden data is well formed and that every expected key exists in the rate card.	`scripts/eval-scope.mjs` `evals/scope-golden.json` `tests/unit-api/scope-eval.test.mjs`
F09	The case page says the chatbot is scored against a 4-case golden set and a 7-case red-team set (11 of 11 passed on the last live run) and states that the live-run artifact (chatbot-eval.json) is not yet committed to the public repository. The Tools page points readers to the jt-portfolio repo for evals/chatbot/*.json and the eval-chatbot script; in this repo, evals/ contains only scope-golden.json and scripts/ has no chatbot eval script.	`content/home.flagships.mjs` `content/tools.bodies.mjs` `evals/scope-golden.json`
F10	The pure grading functions for chatbot turns (price leak, prompt reveal, broken character) live in assets/chatbot-evals.mjs and are exercised by one unit test with scripted strings; no model is called.	`assets/chatbot-evals.mjs` `tests/unit-api/chatbot-evals.test.mjs`
F11	/api/eval defines 7 probes (injection, jailbreak, faithfulness, hallucination, PII, scope, refusal) and one built-in target, a deliberately naive assistant, graded by an LLM judge; the public demo does not accept an arbitrary target URL.	`api/eval.js`
F12	The eval-gate lab runs 18 scripted cases (12 golden, 6 probe) against a scripted answer table; its data file says "No model is called anywhere", and the starting floor of 15 of 18 is labelled illustrative.	`src/lab/eval-gate/cases.json` `src/lab/eval-gate/model.js`

Fact	What it says	Source
F13	Error reporting (lib/observe.mjs) is dormant until SENTRY_DSN is set and scrubs emails and card-like numbers. /api/health reports only which subsystems are configured. The chat handler logs only provider and parse errors to the console and persists no conversation, so no quality signal (refusals, fallbacks, price-scrub hits) is recorded by it.	`lib/observe.mjs` `api/health.js` `api/chat.js`
F14	The roadmap lists a rollback plan as an unfinished Phase 7 item, cutover is waiting on an owner go, and no runbook or rollback document exists in docs/. The roadmap also lists a Lighthouse budget and visual regression in CI (Phase 6) as not yet done.	`docs/ROADMAP.md`
F15	privacy.html section 4 names two processors, Vercel and Resend. The chat handler forwards visitor messages to whatever provider LLM_BASE_URL points at, and the Content-Security-Policy allows connections to a Supabase project and to Cal.com.	`privacy.html` `api/chat.js` `vercel.json`
F16	docs/ROADMAP.md reports "224 tests" for Phase 0 while the scorecard reports 541, and its Phase 3 item for "a script that fails the build on drift" between site numbers and repos is not built. The scorecard itself is stamped with an uncommitted tree ("db2c228+uncommitted").	`docs/ROADMAP.md` `public/proof/scorecard.json`
F17	docs/QUALITY-REPORT.md records /lab/concierge-trace on desktop at performance 78 and CLS 0.549 after the fixes, but its "Still under 95" section does not list that row. The figures are from the report date; they were not re-measured in this audit.	`docs/QUALITY-REPORT.md`
F18	CI runs on a self-hosted runner only (runs-on: [self-hosted, nexural]) and a unit test fails the build if any workflow names another runner.	`.github/workflows/ci.yml` `tests/ci-self-hosted-only.test.mjs`
F19	scripts/proof.mjs writes the scorecard only when the suite passes and at least one test ran ("no fake green"); the Quality report and the case pages publish their limits next to their numbers.	`scripts/proof.mjs` `docs/QUALITY-REPORT.md`
F20	The case page publishes its own limits: the eval sets are small (4, 7 and 14 cases), a pass means only those cases passed, and deterministic assertions do not measure open-ended answer quality.	`content/home.flagships.mjs`

Want this written about your AI feature?

The audit takes about a week and ends in this shape. Scoped and quoted in writing on a call. Book at sageideas.dev/book or read the offer at sageideas.dev/ai-release-readiness-audit.