



FEASIBILITY STUDY

The cross-division agentic workspace

Can a system built to give coding agents a shared memory carry a marketing brief through finance and out the other side with a manager's approval on it?

Gap analysis · Feasibility · Forward plan

7 August 2026

Cairn / Kuavar · assessed against 00fd267

Prepared by reading the codebase, not the plan. Every claim in this document is anchored to a file and a line; the evidence index is on the last page.

Two disclosures: the recent half of Kuavar hot memory was read (26k of 84k characters), and the workflow migration was not confirmed to have run against a live database.

The verdict

ASSESSMENT

Feasible, and cheaper than it looks — but the cost is not where you'd expect it.

The orchestration half is roughly four weeks. The execution half is two to three months and reverses a documented architectural decision. The thing most likely to kill it is neither: it is that **nobody gets told a task is waiting for them.**

SUBSTRATE

9 / 14

capabilities already built and division-agnostic

ORCHESTRATION

4 weeks

run, roles, approval, notification

EXECUTION

8–12 weeks

server-side agents for non-engineers

HIGHEST RISK

Silence

no notification transport exists anywhere

What the vision asks for

A workspace where AI agents run **a team's own procedures**, across divisions rather than only inside engineering. A marketing video moves from an idea, through a budget in finance, to a manager's approval — each step held by a role, some steps carried by agents and some by people, with the whole thing recorded well enough that anyone can later ask who decided what.

Why this is not a pivot

The codebase already assumes it. The workflow entity file says so in a comment written before the question was asked:

A workflow node names a role and never an identity. “Sarah approves” is wrong the day Sarah changes team; “Legal reviewer approves” survives it — and is also the only form that can be standardised across divisions who do not share people.

```
apps/api/src/contexts/workflow/infrastructure/entities/index.ts:26-29
```

Roles are scoped to the **organisation**, not to a space, for exactly this reason. The decision was taken deliberately and recorded at the time: space-scoping would have made the cross-division case impossible.

THE SHAPE OF THE FINDING

The substrate is built. The seams are not. Nine of fourteen capabilities the vision needs already exist and none of them is engineering-specific. What is missing is the connective tissue: nothing turns a drawn procedure into work, no role names a person, no approval is recorded as a decision, and nothing tells anyone their turn has come.

Gap analysis

Fourteen capabilities the cross-division vision requires, measured against what is in the repository today.

CAPABILITY	STATE	EVIDENCE	GAP
Reusable procedure a canvas, validated and acyclic	BUILT	workflow/domain/workflow.ts:55	None. Dangling-edge check runs before the cycle walk, deliberately.
Roles as vocabulary org-scoped, survive a reorg	BUILT	workflow/.../entities/index.ts:31	Names no people — see row 10.
Execution graph tasks, dependencies, readiness	BUILT	contexts/task — 1,858 LOC	Not minted from a workflow — see row 9.
Human or agent holder mutually exclusive by constraint	BUILT	task/.../entities/index.ts:41,43	None. This is the human-in-the-loop primitive, already correct.
Handoff between steps five fixed sections	BUILT	contexts/handoff — 1,192 LOC	None. A brief crossing to finance is the same shape as one crossing to a reviewer.
Knowledge base per initiative spaces, versions, link graph	BUILT	contexts/knowledge — 3,269 LOC	None.
Provenance person vs agent, on every row	BUILT	actor_kind on tasks, doc_versions, handoffs, attachments	None. This is the part a competitor structurally cannot draw.
Divisional governance org → space, strictest wins	BUILT	policy/domain/policy.ts:194	None. Finance gets tighter rules than marketing with no bespoke wiring.
Tenant isolation RLS, forced, org from the token	BUILT	21 tenant tables, ENABLE + FORCE	None.
Run instantiation a canvas becomes work	MISSING	workflow.service.ts — list, create, read, save, remove. No start.	The single missing link. Everything downstream already works.
Role membership a role resolves to someone	MISSING	org_roles — id, slug, name, description. No members table.	A step cannot be assigned to “Finance Approver” because nobody is one.

CAPABILITY	STATE	EVIDENCE	GAP
Approval as a decision who, when, and why not	PARTIAL	TASK_STATUSES includes review	No decision record, no rejection path, no reason.
Notification your turn has come	MISSING	mailer.ts – signup and invites only. No SSE. No change bus.	A step waiting on a human who never learns it is waiting.
Non-engineer runtime where a marketing agent runs	MISSING	workbench – 27,458 LOC, the only client that runs an agent	Terminal panes, git pane, shell history. Finance will not open it.



Fig. 1 – Fourteen required capabilities by state.

WORTH STATING PLAINLY

The workbench is the only engineering-shaped thing in the repository. At 27,458 lines it is roughly the size of the entire API, and it is the one component the vision cannot reuse. Everything else — the memory, the graph, the governance, the provenance — was general-purpose all along and nobody had said so.

The structural fork

There is one modelling decision that decides whether this vision is a weekend or a rewrite. It has to be taken before any code is written, and the free answer is also the better one.

The trap

If **divisions are modelled as spaces**, the vision is blocked at the schema. A marketing step cannot depend on a finance step, and the refusal is explicit:

`APPS/API/SRC/CONTEXTS/TASK/APPLICATION/TASK.SERVICE.TS:365-367`

"A dependency must join two tasks in the same space — a blocker on another board is not visible to the people working this one."

Reinforced at :157 — a task cannot change space, because its links and its handoffs resolve inside the one it was filed in. Both rules are correct. Neither is going to be relaxed for this.

The way through

A workflow run is a space.
A division is a role.

"Ramadan campaign video" is a space. Creative, Finance, Brand and Approver are `org_roles` — already organisation-scoped precisely so the same "Legal reviewer" can appear in a marketing flow and an engineering one. Every step is a task in that one space, so dependencies are legal, links resolve, and handoffs land where the next person will look.

This requires no schema change at all — and it is the better product design independently of that. A campaign's memory should live with the campaign, not be scattered across four departmental filing cabinets that each hold a quarter of the story.

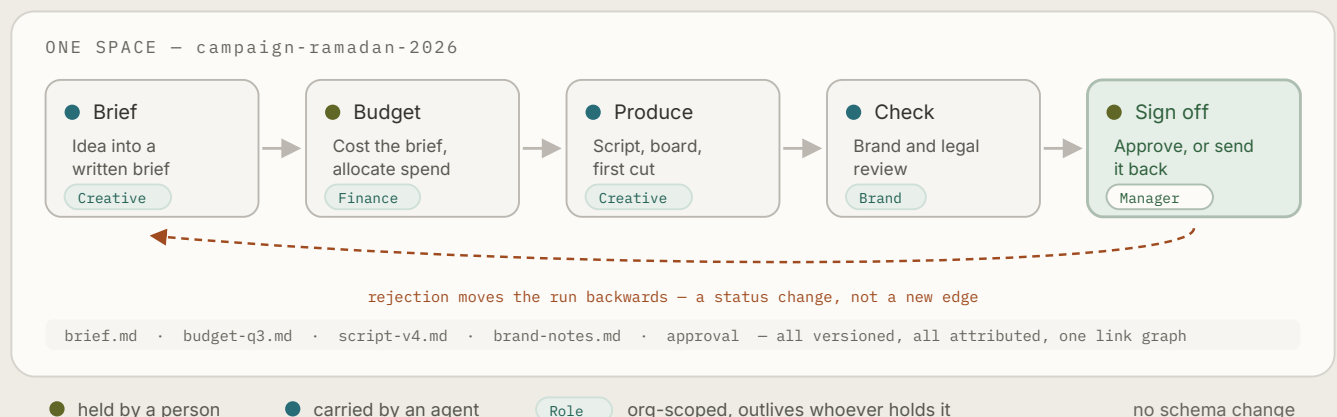


Fig. 2 — The marketing-video flow under the recommended model.

Feasibility, blocker by blocker

Five things stand between the repository as it is today and one marketing video going through it end to end. They are listed in the order they must be built, which is not the order of their difficulty.

01 The run does not exist

≈ 3 days

`workflow.service.ts` offers `list`, `create`, `read`, `save` and `remove`. There is no `start`. Nothing turns a drawn canvas into work, and the page says so on screen rather than implying otherwise.

One method does it: take or create a space, mint a task per node, mirror `workflow_edges` into `task_dependencies`, resolve each role to a holder. Readiness from an empty `blockedBy`, the board, and handoffs all already work.

Trap, already written down: `replaceGraph` is delete-then-insert, and its own comment notes this “becomes a diff the day a run references node ids.” That day is this feature.

02 Roles name nobody

≈ 3 days

`org_roles` is a vocabulary table — id, slug, name, description — and nothing more. Nothing maps “Finance Approver” to a person or to an agent, so a minted step has no one to go to.

Needs an `org_role_members` table plus a resolution rule for what happens when a role has zero holders, one, or five. Zero is the interesting case: a run that mints an unassignable step should refuse at start, not stall at step four.

03 Approval is a status, not a decision

≈ 1 week

Statuses run `todo` → `in_progress` → `blocked` → `review` → `done`. `review` is almost approval, but there is no record of **who** decided, **when**, or **why not** — and an approval you cannot audit is the one thing this product exists to sell.

Rejection is the harder half and less hard than it looks: sending work back is a status change on an upstream task, not a new edge, so the graph stays acyclic and `findAnyCycle` is untouched. What is genuinely undecided is what a rejection does to downstream steps already marked done.

04 Nobody gets told — the sleeper

≈ 2–3 weeks

There is **no notification of any kind** in the repository. No server-sent events, no change bus, no websocket. `nodemailer` appears in exactly one file — `identity/infrastructure/mailer.ts` — for signup verification and invites.

An engineer will keep a board open all day. **A finance manager will not.** A step that waits on a human who never learns it is waiting is a workflow that stalls in silence, and one stalled approval is enough to send a team back to WhatsApp permanently.

This is the cheapest item to defer and the most likely to sink the product. Email first; in-app later.

05 Where does a marketing agent run?

≈ 8–12 weeks

The workbench spawns agents on the engineer's own laptop, and `local-execution-model.md` §4.2 states that source never leaves the machine. Marketing has no laptop client and never will — a finance manager is not going to open a terminal emulator with a git pane in it.

The answer is already designed. The review-agent feature puts headless Claude Code on your own infrastructure: a Postgres queue claimed with `FOR UPDATE SKIP LOCKED`, a worker spawning `claude -p`, a metered unit per run, a short-TTL server-side session mint. Its own task text records that it “reverses `local-execution-model.md` §4.2.”

That runner **is** the cross-division execution model. Not a new architecture — the same one, generalised from “review a pull request” to “run a workflow step.” Build the review agent first and roughly seventy per cent of this arrives with it. Skip it and you will design the same thing twice.

THE FRONT DOOR IS CHEAPER THAN THE LAST ONE

`apps/web` is 11,673 lines against the workbench's 27,458, and already ships `/tasks`, `/spaces`, `/workflows`, `/graph` and `/search` across seventeen routes. Non-engineers largely have a home already. The only thing missing from it is agent execution.

Unit economics — a real constraint

This is not a technical risk. It is an inversion of the margin structure between the two halves of the product, and it comes from a term of service rather than from the code.

No claude.ai login or claude.ai rate limits may be offered to our users... must authenticate with an API key.

Anthropic SDK terms, recorded in Kuavar hot memory, 2026-08-06

ENGINEERING AGENTS

Cost of goods $\approx$ zero

A workbench pane runs the engineer's own `claude` binary on the engineer's own subscription. You sell the memory, the governance and the client; you buy none of the tokens.

CROSS-DIVISION AGENTS

Cost of goods is yours

The moment an agent runs on your infrastructure for a marketing manager, **you pay for every token** — and the terms above mean it must be your API key, not their subscription.

At Starter — Rp 35.000 per member per month, roughly two dollars twenty — a handful of server-side runs is underwater. The metering scaffolding is ready: `usage_counters.metric` carries no CHECK constraint on purpose, so a new billable unit needs no migration at all. What is missing is the numbers.

DECISION REQUIRED BEFORE THE FIRST NON-ENGINEERING RUN

The per-plan allowance for server-side agent runs is **unset**, and the review-agent task already lists it as open. This is not a blocker — it is a pricing decision that must land before the feature ships rather than after, because retro-fitting a cap onto customers who already have the behaviour is a migration and a conversation, not a config change.

The countervailing point

Pricing is **per member per month**. Cross-division means marketing, finance and operations seats on top of engineering ones — so the revenue side improves with exactly the same change that worsens the cost side. The question is not whether the economics work; it is whether the per-seat price rises to meet the tokens, or the allowance is capped hard. The plan model already hard-stops at every cap on every tier, so the machinery for the second option exists.

The forward plan

Seven workstreams. The first four are worth building whether or not the cross-division vision proceeds — a workflow that cannot run is not a feature.

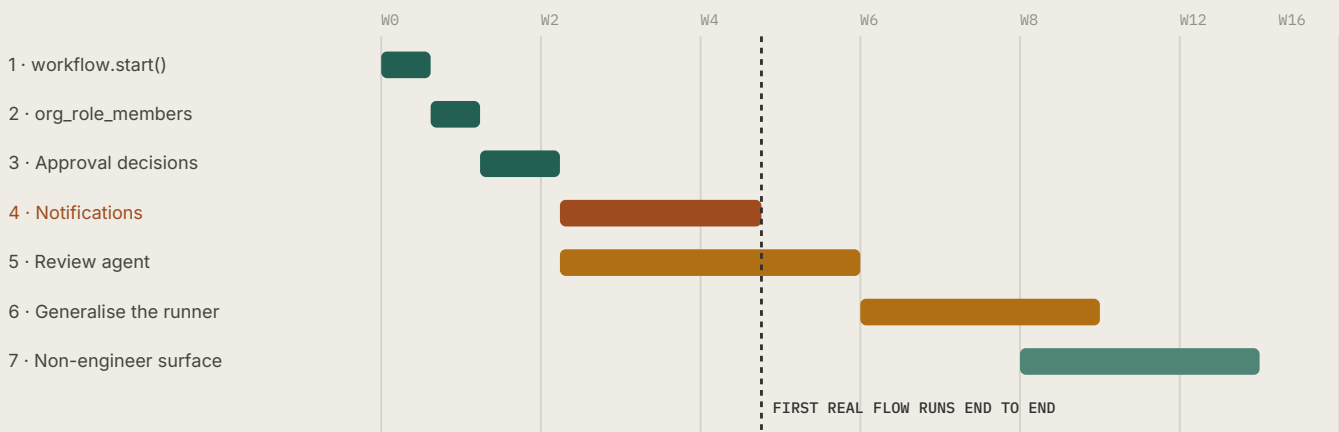


Fig. 3 — Sequencing. The dashed line is the point a real cross-division flow can be demonstrated.

#	WORKSTREAM	SIZE	WHY IT SITS HERE
1	workflow.start() a canvas mints tasks	3 days	Turns a diagram into product. Unblocks every demonstration that follows, and is the smallest diff on this page.
2	org_role_members a role resolves to a holder	3 days	A role that names nobody cannot be assigned to. Immediately after 1 because 1 exposes it.
3	Approval decisions who, when, why not	1 week	The step the marketing-video example actually turns on, and the one an auditor asks about first.
4	Notifications email before in-app	2–3 weeks	Do not defer this. Silence is the failure mode that ends the pilot, and it is invisible until it happens.
5	Review agent server-side runner, first use	5 pts	Builds the server-side execution model under an engineering excuse, where the buyer already exists and tolerates rough edges.
6	Generalise the runner from PR review to any step	4–6 weeks	Now a configuration change rather than an architecture. This is the payoff for ordering 5 before it.
7	Non-engineer surface in <code>apps/web</code>	4 weeks	The cheapest client you will ever build — most of the routes exist; only agent execution is missing.

THE HONEST TEST

Streams 1 through 4 come to about four weeks. Build them, put **one real non-engineering flow** through the system by hand, and watch whether anyone outside engineering opens it a second time. That answer costs a month and settles a question no amount of further analysis will.

If they do, streams 5 through 7 are worth the quarter. If they do not, you have still shipped a working workflow engine to the division that was already paying — and lost nothing.

Positioning, briefly

The strongest argument for this vision is not technical. **“Agentic workflow with approval gates” is a category buyers already understand and budget for**, whereas “shared memory for coding agents” is one you would have to teach them. The governance substrate is genuinely rare; the memory framing hides it.

The strongest argument against is that cross-division means selling to four buyers instead of one, while the only client that runs an agent today serves the division that is not paying for the other three. **Engineering stays the wedge** — reverse onboarding gives a ninety-second activation demo that nothing else in the market has — and cross-division is what makes the account grow after it lands.

Evidence index

Every load-bearing claim in this document, with the file it was read from. Assessed against [00fd267](#) on 7 August 2026.

LOCATION	WHAT IT ESTABLISHES
workflow/.../entities/index.ts:26–29	Roles name a role and never an identity — written for the cross-division case.
workflow/.../entities/index.ts:31	<code>org_roles</code> is org-scoped, with no membership table.
workflow/.../entities/index.ts (module doc)	“The run of a workflow is deliberately not modelled here... <code>tasks</code> already are one.”
workflow/domain/workflow.ts:55	Whole-canvas validation; dangling edges checked before the cycle walk.
workflow/application/workflow.service.ts	Five methods, none of which starts a run.
task/application/task.service.ts:365–367	A dependency must join two tasks in the same space.
task/application/task.service.ts:157–164	A task cannot change space.
task/.../entities/index.ts:41,43	Person or agent assignee, mutually exclusive by constraint.
task/domain/fields.ts:16	Five statuses; <code>review</code> exists, approval does not.
policy/domain/policy.ts:194	Org → space policy merge, strictest wins; a space may tighten, never loosen.
policy/domain/policy.ts:1–35	Enforced vs advisory controls, and the refusal to draw a lock that lies.
metering/domain/plans.ts:40	Monthly price per member, in rupiah; four plans, hard stop at every cap.
migrations/17856000000000-UsageCounters.ts:35–39	No CHECK on <code>metric</code> — a new billable unit needs no migration.
identity/infrastructure/mailer.ts	The only mail path in the system; signup and invites.
apps/web — 17 routes, 11,673 LOC	The non-engineer front door largely exists already.
apps/workbench — 27,458 LOC	The only client that runs an agent, and the only engineering-shaped component.
Kuavar task bb6edc1f	Review agent — server-side runner, queue, metered unit; reverses §4.2.

LOCATION	WHAT IT ESTABLISHES
Kuavar hot memory, 2026-08-06	Anthropic SDK terms: server-side agents must use an API key, not a claude.ai login.
Kuavar hot memory, 2026-08-06	Workflow migration had not run against a database at the time of writing.

LIMITS OF THIS ASSESSMENT

The recent half of Kuavar hot memory was read — approximately 26,000 of 84,000 characters. Earlier notes may contain decisions not reflected here.

The workflow migration was registered in [a5b7286](#) , but it was not confirmed to have run against a live database, and the tenant-isolation assertion was not confirmed to have been exercised against the four new tables. Standing rule four says checked-in, not looks-right.

Effort estimates are solo-with-agents and assume the existing test discipline is maintained. They are relative sizes, not commitments.