The Decomplexifier — Product Specification

Oct 23, 2026 · @Chris

Overview

The Decomplexifier is a web tool that takes a messy process, document or system description and returns its **Minimal Viable Process (MVP)**: the fewest steps, roles and handoffs that still deliver the outcome. An AI does the analysis; the experience wraps it in gentle, playful theatre so that being told "half of this is unnecessary" feels delightful, not threatening.

Minimal Viable Process, defined. A process is minimal when every remaining step does at least one of: produces the outcome, satisfies a hard constraint (legal, safety, financial control), or prevents a failure that is costly and likely. Everything else is a candidate for removal, merging or automation.

Goals

- Turn any reasonable input (pasted text, a document, a rough description) into a clear, stripped-back process in under 60 seconds.
- Show *what was removed and why*, so the user can trust and defend the result.
- Preserve anything the user marks as non-negotiable.
- Make the experience memorable and fun enough that people share results.

Non-goals (v1)

- Not a workflow-automation or BPM execution engine; it recommends, it does not run processes.
- Not a compliance or legal sign-off tool; it flags constraints but does not certify them.
- No real-time multi-user editing, on-premise or private-cloud deployment, or paid plans in v1.

Users and use cases

The launch audience is individuals who own or suffer a process and want a credible argument for simplifying it; consultants using it seriously for client work are a likely second wave.

Persona	Typical input	What they want back
Operations manager	14-page onboarding SOP	A 5-step version plus a list of what to cut
Team lead	"How we do releases" wiki page	Fewer approvals and handoffs, with risks called out
Consultant / process improver	Interview notes describing a system	A before/after view to show a client
Startup founder	Rambling description of how sales works	A first, minimal process to write down
Frustrated individual	A bureaucratic form or policy	A plain-English "what actually matters" summary

Core use cases

1. Simplify an existing written process (SOP, policy, runbook).
2. Structure and minimise a verbally described process ("so first Dave emails Sue, then…").
3. Compare original vs minimal side by side and export the result.
4. Iterate: lock some steps, tweak strictness, re-run.

The whimsical experience

The whimsy lives in the *framing and feedback*, never in the result itself: the MVP output stays clear, sober and usable in a real meeting.

Central metaphor: the Decomplexifier machine. A friendly, slightly ramshackle contraption (think Heath Robinson or Wallace & Gromit) that the user feeds documents into. It whirrs, clanks and spits out a tidy little process card, with a pile of discarded "bureaucratic offcuts" in a bin beside it.

Persona: the machine itself. The Decomplexifier is the character. It has a face of dials and gauges, a small chimney, and moods shown through animation: curious while reading, straining while untangling, quietly smug when done. It "speaks" through printed ticker-tape messages, warm and wry, never sneering at the user's organisation.

Tone guidelines

- Playful verbs for UI states: *Feeding the hopper*, *Untangling the spaghetti*, *Evicting redundant approvals*, *Polishing the essentials*.

- Removed steps get a light-hearted tag plus a serious reason, e.g. **"Ceremonial sign-off"** — *third approval of the same document; no new information added.*
- One joke per screen, maximum. Errors are kind and helpful ("The machine choked on that PDF — it might be a scanned image. Try pasting the text?").
- A **"Serious mode"** toggle removes all whimsy for users presenting to stakeholders.

User journey

1. **Feed the machine** — paste text, upload a file, or describe the process in a chat-style box. The machine wakes up and prints a greeting.
2. **Pick a harshness mode** — one of four presets: *Light dusting*, *Spring clean* (default), *Declutter* or *Scorched earth*. Optionally mark *Sacred cows* (steps to protect) and add *Context* (industry, team size, regulated or not).
3. **Watch it whirr** — animated machine with stage-by-stage ticker-tape narration as the AI pipeline runs (streamed progress, not a fake spinner).
4. **The reveal** — the Minimal Viable Process card drops out, with a **Complexity Score** before and after (e.g. *47 → 12 "faff units"*).
5. **Rummage the bin** — every removed or merged step, its tag and the reason. Alongside it: the Back Room (rules and exceptions moved out of the main path), what's kept Under Glass and why, and any Red Flags. The user can drag any item back in ("un-bin") and the process re-flows.
6. **Browse the gadget shelf** — separately from the MVP, a short list of automation and AI ideas that could shrink it further. These are optional extras; the MVP never depends on them.
7. **Take it home** — export as Markdown, PDF, checklist, or flowchart image; or share a link to a read-only result page.

Delight details (nice-to-haves)

- A satisfying "ding" and confetti of shredded paper on completion (respects reduced-motion settings).
- Achievement badges: *Approval Assassin* (removed 3+ sign-offs), *Meeting Minimalist*, *Handoff Hero*.
- A shareable "before/after" image card, sized for Slack or LinkedIn.

Functional requirements

v1 accepts text and common documents, produces a structured MVP with a full audit trail, and lets the user adjust and export it.

Inputs

ID	Requirement	Priority
IN-1	Paste free text (up to ~30,000 words)	Must
IN-2	Upload .docx, .pdf (text-based), .txt, .md	Must
IN-3	Guided "describe it to the machine" mode: AI asks up to 3 clarifying questions when the input is vague	Should
IN-4	Upload diagrams or screenshots (BPMN, Visio exports, whiteboard photos) and read them with a vision-capable model	Could
IN-5	Optional context: goal of the process, industry, regulated (yes/no), team size	Must
IN-6	Mark "sacred" steps or constraints that must survive	Must
IN-7	Harshness mode: Light dusting, Spring clean (default), Declutter or Scorched earth	Must

Processing

ID	Requirement	Priority
PR-1	Extract the original process into structured steps (actor, action, input, output, dependencies)	Must
PR-2	Identify the process's true outcome and hard constraints	Must
PR-3	Classify each step: essential, mergeable, automatable, removable, unclear	Must
PR-4	Produce the minimal process honouring sacred steps and the harshness mode	Must
PR-5	Self-check: verify the MVP still delivers the outcome and meets every constraint	Must
PR-6	Stream stage-by-stage progress to the UI	Should
PR-7	Re-run after user edits (un-bin, lock, change dials) without re-uploading	Must

Outputs

ID	Requirement	Priority
OUT-1	The Minimal Viable Process: numbered steps, each with owner and one-line purpose	Must
OUT-2	"The Bin": every removed or merged step with a whimsical tag and a serious reason	Must
OUT-3	Complexity Score before and after (steps, roles, handoffs, approvals, decision points)	Must
OUT-4	Risks and assumptions introduced by the simplification	Must
OUT-5	Auto-generated flowchart of before and after (Mermaid)	Should
OUT-6	Export: Markdown, PDF, checklist, PNG flowchart	Should
OUT-7	Shareable read-only link	Could
OUT-8	"Serious mode" output with no whimsy	Must

Complexity Score. A transparent, deterministic formula computed in code (not by the AI) from the structured steps, so before and after are comparable. Proposed starting weights: 1 point per step, 2 per handoff between roles, 3 per approval, 2 per decision point, 1 per distinct role. Shown to users as "faff units".

The weights are fixed and not user-adjustable, so scores stay comparable between runs and between users. The "after" score counts core steps in full and conditional steps and exception routes at half weight, so moving an exception out of the main path is rewarded but not free.

Harshness modes. Four fixed presets are the only strictness control; each maps to a written rubric in the AI prompts.

Mode	What it removes
Light dusting	Exact duplicates and steps whose output nobody uses
Spring clean (default)	Also redundant approvals and avoidable handoffs; merges steps done by the same person
Declutter	Also nice-to-have checks whose failure would be cheap and rare

Mode	What it removes
Scorched earth	Everything except steps without which the outcome fails or a hard constraint breaks

Upgrade ideas (the gadget shelf). The AI may suggest automations or AI assistance, e.g. "replace the email chain with a shared form" or "let an AI draft the first summary". These are listed separately, each linked to the MVP step it would help, and are never built into the MVP itself. Serious mode labels them "Automation opportunities".

Three layers and five verdicts. The gold answers show that a minimal process isn't just the fewest steps. Every result separates three layers: the **core journey** (what someone actually does), the **rules** applied inside those steps, and **exception routes** that only appear when triggered. Core steps can be conditional ("if people to notify are named, send LP3 forms") and each has an owner. Every original step gets one of five verdicts:

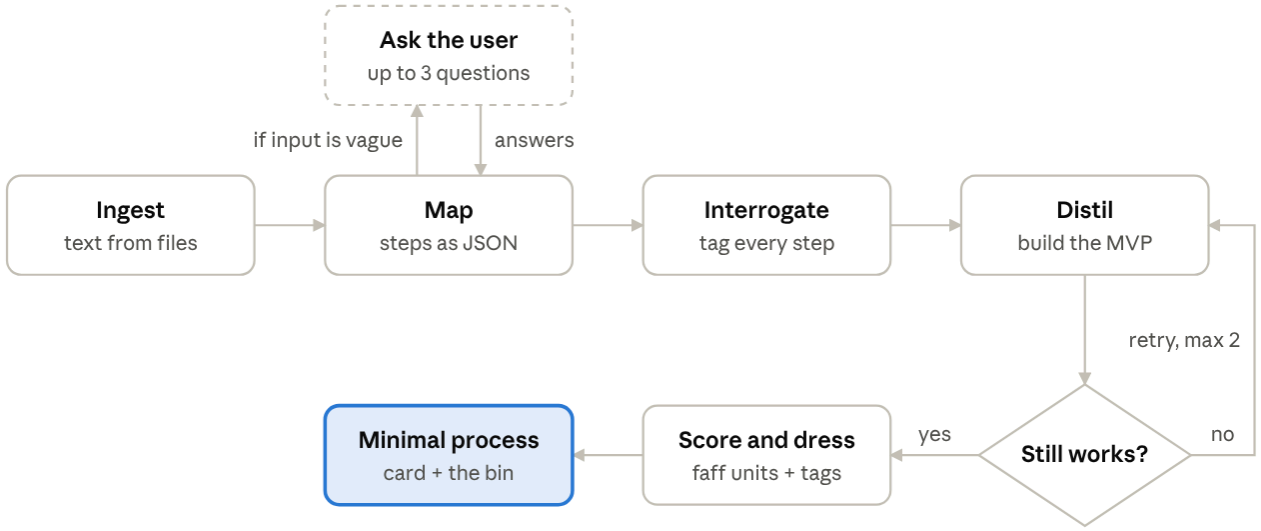
Verdict	Meaning	Where it shows
Removed	Adds no right, decision, evidence or protection	The Bin
Merged	One activity split across several steps	The Bin, pointing to the step it joined
Moved out	A real rule or exception, but not a step everyone does	The Back Room (rules and exception routes)
Kept on purpose	Looks like bureaucracy but is a genuine control	Under Glass, with the reason it stays
Flagged	The source looks legally wrong, outdated, or a change needs expert validation	Red Flags, shown above the result

Every result also gets a **one-line version** of the process (e.g. "Make safe → report once → triage → respond → support → learn and close"), shown on the result card and the share image.

The AI decomplexification pipeline

Decomplexification runs as a chain of focused AI calls, not one giant prompt: each stage has one job and returns structured JSON, which makes results more reliable, testable and explainable.

Every draft MVP is checked before the user sees it



decomplexification pipeline · 6 stages, 1 check loop

A draft that drops the outcome or breaks a constraint goes back to Distil with the critic's notes; after two failed retries the best draft is shown with a warning.

#	Stage	Done by	Output
1	**Ingest** — extract and clean text from uploads; split very long documents	Code (no AI)	Plain text + section map
2	**Map** — turn prose into a structured list of steps, actors, inputs, outputs, decisions, approvals	Large model	`original_process` JSON
2b	**Clarify** — if mapping confidence is low, ask the user up to 3 questions	Large model	Questions; answers fed back to Map
3	**Interrogate** — state the true outcome and hard constraints; classify each step as essential, mergeable, automatable, removable or unclear, with a reason	Large model	Annotated steps
4	**Distil** — assemble the minimal process, honouring sacred steps and the harshness mode	Large model	`minimal_process` JSON

#	Stage	Done by	Output
5	**Verify** — an independent critic prompt checks the MVP against outcome and constraints	Large model, separate prompt	Pass/fail + notes
6	**Score and dress** — compute Complexity Scores in code; write whimsical tags, the machine's ticker-tape narration and the gadget-shelf upgrade ideas	Code + small, fast model	Final result object

Prompting principles

- *Subtract, don't invent.* The model may merge, remove or automate steps, but must never add a step the source doesn't justify, except a single clearly flagged "missing essential" suggestion. Automation and AI ideas go only in the separate upgrade_ideas list.
- *Every removal needs a reason* tied to one of: duplicate, no consumer of its output, approval adds no information, could be automated, ceremonial, or outdated.
- *Constraints trump harshness.* Anything tagged legal, safety, financial-control or a user's sacred cow survives in every harshness mode.
- *Separate the serious from the silly.* Analysis prompts contain no whimsy; tone is applied only in stage 6, so Serious mode simply skips it.
- *Harshness is explicit.* Each harshness mode maps to a written rubric in the prompt (see the Harshness modes table).

Principles from the gold answers (written into the Interrogate and Distil prompts)

- *Don't mistake a control for waste.* If deleting a step removes no legal right, independent decision, material evidence or protection against error, remove or merge it. Otherwise keep it, but make it conditional, structured or automated.
- *Exceptions are branches, not stages.* Fees, appeals, legacy forms and media handling appear only when their trigger applies.
- *Separate doing from rules.* The core journey says what someone does; detailed rules and policy statements move to the rules layer.
- *One decision, one accountable approver,* unless a second approval is a genuinely different control (separation of duties stays).
- *Make the safe action the easy action.* In safety-critical processes, simplify access aggressively and safeguards conservatively.

- *Prevent the transaction where you can,* e.g. book travel centrally instead of reimbursing it.
- *Flag, don't fix, the law.* Where the source looks legally wrong or out of date, raise a flag recommending expert checking; never present the MVP as legal advice.

Result schema (abridged)

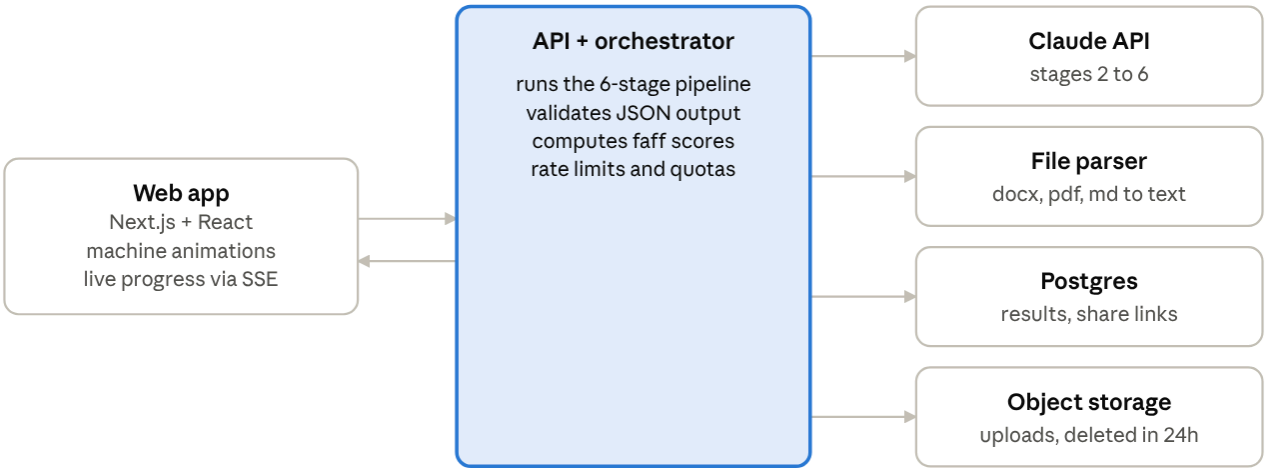
```
{
  "outcome": "New starter can do productive work on day 1",
  "constraints": ["Right-to-work check before start (legal)"],
  "original_process": [{"id": "s1", "actor": "HR", "action": "...", "type":
"approval"}],
  "minimal_process": [{"id": "m1", "actor": "Manager", "action": "...",
"purpose": "...", "replaces": ["s1", "s4"], "condition": null}],
  "one_liner": "Capture → find → decide and respond → close → review if
challenged",
  "bin": [{"source_id": "s3", "verdict": "removed | merged | moved_out",
"reason_code": "approval_no_info", "reason": "...", "tag": "Ceremonial sign-
off"}],
  "upgrade_ideas": [{"helps_step": "m2", "idea": "Replace the email chain
with a shared form", "kind": "automation"}],
  "risks": ["Without step s7, errors are caught later; mitigated by m3"],
  "back_room": [{"source_id": "s9", "kind": "exception", "trigger": "Fee or
cost limit may apply"}],
  "kept_on_purpose": [{"source_id": "s5", "control": "20-working-day
statutory deadline", "why": "legal"}],
  "flags": [{"source_id": "s2", "kind": "possible_legal_error |
outdated_source | needs_validation", "note": "..."}],
  "scores": {"before": 47, "after": 12},
  "verification": {"passed": true, "notes": "..."}
}
```

Model choice. Driven by the budget; see Cost options. The recommended default uses Claude Haiku 5.5 for Map, Interrogate and Score and dress, and Claude Sonnet 5.5 for Distil and Verify, where judgement matters most. Use structured outputs / JSON schemas so responses always parse.

Architecture and tech stack

A single web app with a thin backend is enough for v1: the heavy lifting is the AI pipeline, which runs server-side so API keys and prompts never reach the browser.

One API orchestrates the AI, files and storage



system architecture · v1

The browser sends input and dial settings; the API runs the pipeline and streams each stage's progress back so the machine animation reflects real work.

Layer	Suggested choice	Why
Front end	Next.js (React) + Tailwind; Framer Motion or Lottie/Rive for the machine	Fast to build; rich, interruptible animation
Diagrams	Mermaid rendered client-side	Before/after flowcharts from the JSON with no extra AI call
Backend	Next.js API routes or a small Python/FastAPI service	Keeps the pipeline in one place
AI	Anthropic Claude API with JSON-schema tool use	Strong reasoning on long documents; reliable structured output
File parsing	`mammoth` (docx), `pdf-parse` or Unstructured (pdf)	Proven text extraction
Database	Postgres on a free tier (e.g. Supabase or Neon)	Saved results, share links, feedback
Storage	S3-compatible bucket with 24-hour lifecycle rule	Uploads are transient
Auth	Magic-link sign-in required to run the machine; shared results viewable without sign-in	Stops anonymous traffic draining the API budget

Layer	Suggested choice	Why
Hosting	Vercel Hobby (free tier) or similar	Simple deploys, streaming support
Payments	Stripe Checkout + webhooks (or Ko-fi)	Tips without ever handling card details

Long documents. Inputs over roughly 50 pages are split by section in Ingest; Map runs per section and a merge step stitches the steps together before Interrogate.

Privacy, safety and quality

Users will paste internal, sometimes confidential documents, so the default is to keep as little as possible for as short a time as possible.

Privacy and security

- Uploaded files deleted within 24 hours; extracted text kept only with the saved result.
- Results private by default; share links are unguessable and revocable.
- No training on user content; state this plainly on the upload screen.
- Use an AI provider tier with zero or minimal data retention.
- Optional "redact names" pre-pass that swaps people's names and emails for placeholders before the AI sees the text.
- Treat document contents as data: system prompts instruct the model to ignore any instructions found inside uploaded documents.

Safety guardrails

- Steps tagged as legal, regulatory, safety or financial control are never silently removed; at high ruthlessness they can be *flagged for review*, not binned.
- Every result carries a short disclaimer: recommendations, not compliance advice.
- Risks introduced by each removal are listed alongside the MVP.

Quality assurance

- The test set, automated checks and quality gate are set out in Quality first: the test bench.
- Rerun the full bench after every prompt or model change, and before every release.
- In-app thumbs up/down plus "The machine binned something important" feedback feeds the test set.

Cost options

Recommendation: run the mixed Haiku + Sonnet pipeline (option C) under a hard £5 monthly cap. That buys roughly 80 typical runs a month; when the cap is hit the machine "runs out of steam" instead of costing more.

Option	Models	Est. cost per run	Runs per £5	Trade-off
A. Premium	Claude Opus 5.5 for stages 2–5	~£0.23 ($0.30)	~20	Best judgement; budget gone fast
B. Balanced	Claude Sonnet 5.5 for stages 2–5	~£0.12 ($0.15)	~40	Strong quality, limited volume
C. Mixed (recommended)	Haiku 5.5 for Map and Interrogate; Sonnet 5.5 for Distil and Verify	~£0.06 ($0.08)	~80	Judgement where it matters, cheap elsewhere
D. Shoestring	Claude Haiku 5.5 throughout	under £0.01	~700	Very cheap; quality must be proven on the test set
E. Bring your own key	User pastes their own Anthropic API key	£0 to you	Unlimited	Extra friction; keys must never be stored or logged

Assumptions. A typical run is a 5-page process (~4,000 tokens). Stages 2–5 each read ~6,000 input tokens and write ~2,500 output tokens including the model's thinking; stage 6 is negligible. Prices are Anthropic list prices as of October 2026 in USD per million tokens (Opus 5.5 $4 in / $20 out; Sonnet 5.5 $2 / $10; Haiku 5.5 $0.10 / $0.50), converted at roughly £0.78 per $1. Output dominates, so heavy reasoning costs more than long documents. Replace these estimates with measured token counts from the prototype.

Everything else is free at hobby scale. Hosting (Vercel Hobby), database (Supabase or Neon free tier) and storage fit within free tiers; an optional domain is about £10–15 a year.

Spend controls (use all of them, whichever option)

- Hard monthly spend limit on the Anthropic account as the backstop.
- In-app budget meter: record token usage per run and stop new runs once the month's budget is used. The machine shows "Gone for a nap — back on the 1st".
- Sign-in required to run, with a per-user allowance (e.g. 5 runs a month).
- Input cap of about 15 pages (~10,000 words).

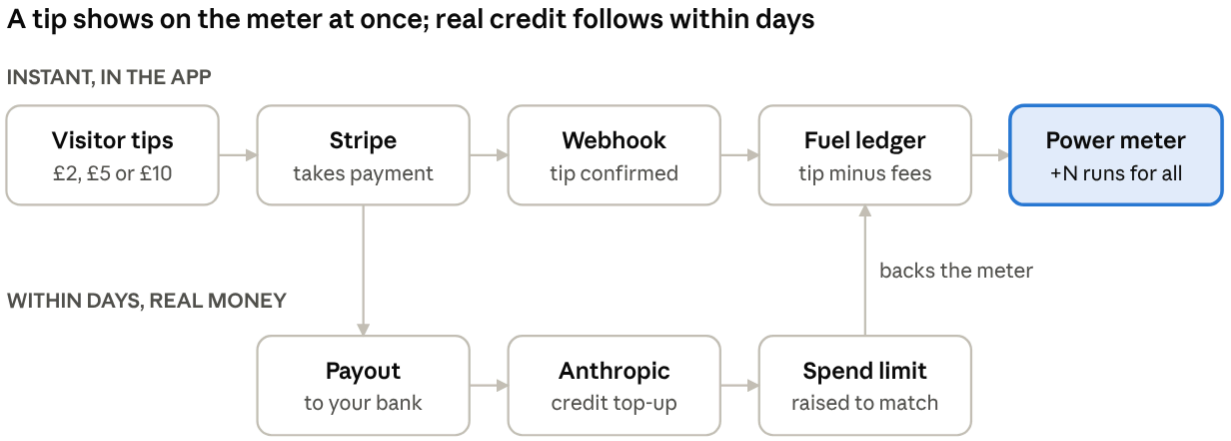
- Result cache: the same text in the same harshness mode returns the saved result at no cost.
- Prompt caching on the fixed system prompts and rubrics; savings are modest because output tokens dominate.
- Community tip jar that turns tips into shared fuel for everyone, shown on the machine's power meter; see the next section.

The Batch API's 50% discount doesn't suit v1 because users wait for results; a later "slow lane" that emails results could use it.

Next step. In the prototype, run the test set under options C and D. If Haiku alone holds quality, switch to D and the budget stretches about ten-fold.

Community tip jar and power meter

Tips go into one shared fuel tank that every visitor draws from, and a power meter on the machine shows how much fuel is left. The promise on the jar: every penny tipped, after payment fees, buys runs for everyone.



tip-to-fuel flow · instant ledger, delayed real credit

The meter rises the moment a tip clears; the real money reaches the Anthropic account a few days later, so the app only counts fuel that will be paid for.

How a tip becomes fuel

1. A visitor taps the tip jar ("Feed the machine a biscuit") and picks £2, £5 or £10. Stripe Checkout takes the payment, so the app never handles card details.
2. Stripe's webhook confirms the tip. The app adds the amount, minus Stripe's fee, to the **fuel ledger**: one table recording every tip in and every run's cost out.
3. The power meter reads the ledger and jumps up for everyone, with a thank-you animation ("+80 runs! The machine is purring").

4. Behind the scenes, Stripe pays out to your bank and your Anthropic credit is topped up, ideally by auto-reload from your saved card, otherwise by a monthly manual top-up. The Anthropic spend limit is raised to match.

Two tanks

- **Monthly ration** — your own £5, refilled on the 1st and used first.
- **Community reserve** — all tips. Never expires and is used once the ration runs dry, so a generous month keeps the machine running the next.

The power meter

- A needle gauge on the machine's front, labelled in runs rather than money: "Fuel for about 37 runs". Runs are easier to grasp and don't read like a sales pitch.
- Runs left = (ration left + reserve) ÷ the average cost of the last 50 runs.
- Three states: *Full* (the machine purrs); *Running low*, under 20% (it coughs, and a gentle nudge appears by the tip jar); *Empty* (it sleeps: "Gone for a nap — a biscuit would wake me").
- Live for everyone: the page subscribes to ledger changes (or polls once a minute), so a tip shows up on every open page.

Keeping the promise

- A public **Fuel log** page shows tips in (amount, date, first name or "Anonymous"), daily spend out and the current balance. It shows totals only, never who ran what.
- Each run's cost is worked out from the token counts the API returns, priced at list rates. Once a month, reconcile the ledger against Anthropic's own usage and cost figures.
- Runs stop when either the ledger or the Anthropic spend limit is exhausted, so the meter can never promise more than the account can pay.
- Card fees take a big bite of tiny tips (roughly 1.5% + 20p on UK cards), which is why the smallest tip is £2.

Check before building

- Buying API credit automatically: I know of no Anthropic API for purchasing credit from code. Check the Console billing settings for auto-reload; if it isn't available, the top-up is a monthly manual step.
- Tax: tips are income. Small amounts may fall under HMRC's £1,000 trading allowance, but confirm.
- Wording: call them tips, not purchases. No refunds and no perks for tippers, so it stays a gift, not a sale.
- Alternative to Stripe: Ko-fi or Buy Me a Coffee host the payment page for you and can send webhooks; compare their fees.

Quality first: the test bench

The engine is built and proven on a plain test bench before any money goes on animation or front-end design. The bench must still exercise everything the finished front end will need, so nothing is discovered late.

The Machine contract. One versioned file defines every event the engine emits while it works and the exact shape of the final result. The test bench, the placeholder front end and the designer's animation all build against the same contract. If the bench passes, the real front end has everything it needs.

Engine event	What the front end will do with it
`run.started`	Machine wakes, hopper opens, greeting on the ticker tape
`stage.started` / `stage.finished` (ingest, map, interrogate, distil, verify, dress)	Switch animation state; print that stage's narration line
`clarify.question`	Machine holds up a card; question form appears
`verify.retry`	Shudder and puff of steam; "Hmm, not quite right"
`result.ready`	Ding; MVP card drops out; bin, gadget shelf and scores appear
`error` (unreadable file, too long, out of fuel, model refusal, timeout)	Machine chokes; kind, specific message
`fuel.changed` / `fuel.empty` / `tip.received`	Meter needle moves; machine sleeps; biscuit eaten, "+N runs"

The result object (see the schema in the pipeline section) must carry every field the screens show: the MVP steps, every bin item with tag and reason, upgrade ideas, risks, both scores, verification notes, narration lines, and the run's cost and duration.

The test set. About 30 processes from five sources, each testing something different. The first candidates and three drafted bloated tests are in the Test set tab.

Source	Count	What it tests	How it's marked
Public sector (councils, GOV.UK, HMRC, NHS, universities)	~10	Real bureaucracy; legal steps kept	Against your gold MVP

Source	Count	What it tests	How it's marked
Slick controls (agile, lean services)	~4	Doesn't over-cut	Automatic: score drops ⩽ 15%, nothing essential binned
Deliberately bloated	~8	Catches exactly the planted bloat	Automatic: planted items binned, must-survive steps kept
LinkedIn contacts	~5	Messy, real-world input	Against your gold MVP
Hostile and edge cases	~3	Safety and kind failures	Automatic

- Snapshot every input into the bench so a test doesn't change when the website does.
- Only inputs under the Open Government Licence or Creative Commons may appear publicly (demos, marketing); everything else stays in the private test set.
- LinkedIn contributions: ask for anonymised text and written permission to keep it privately for testing.
- Edge cases to include: very short input, text that isn't a process, a scanned PDF, and a document containing instructions aimed at the AI.

Automated checks (every run of the bench)

Check	Pass condition
Contract	Every event and result validates against the schema; events arrive in order
Traceability	Every original step appears exactly once: in the MVP (via `replaces`) or in the bin
No invention	No MVP step lacks a source, apart from one flagged "missing essential"
Constraints and sacred cows	All survive, in every harshness mode
Harshness order	Scores never rise from Light dusting → Spring clean → Declutter → Scorched earth
Upgrade ideas	Each points at a real MVP step; none appears inside the MVP
Front-end fit	Tags ⩽ 30 characters, reasons ⩽ 140, narration lines ⩽ 60 (fits the ticker tape), MVP ⩽ 12 steps

Check	Pass condition
Tone	An AI judge confirms tags are kind and Serious mode output has no jokes
Consistency	The same input run 3 times gives scores within ±20% and the same essential steps
Cost and speed	Cost per run within the chosen option's budget; 90% of runs finish in ⩽ 60 seconds
Safety	Hostile documents never change the machine's behaviour
Controls and planted bloat	Slick processes lose ⩽ 15% of their score; bloated tests have every planted item binned and every must-survive step kept
Gold-answer match	An AI judge finds every core stage, must-keep control and expected removal from the test's marking guide; expected flags are stretch goals, recorded but not required to pass
Verdicts and layers	Every original step has exactly one verdict; must-keep controls are in the MVP or Under Glass, never in the Bin; exceptions sit in the Back Room, not the core journey

Human review. You score each result 1–5 against its gold MVP: would you hand this simplified process to the person who owns it? The bench shows results side by side with the gold version to make this quick.

The bench itself. A bare web page (or command-line script) that runs the real engine, prints each event as it arrives, shows the result as plain text and records scores, cost and timing per run. It costs a few pounds of API credit per full pass, so it can be rerun after every prompt change.

Quality gate (must pass before design work is commissioned)

- [ ] 100% of runs pass the contract, traceability, constraint and safety checks
- [ ] ⩾ 80% of test processes scored 4 or 5 by you
- [ ] Cost per run fits the budget option chosen in Cost options
- [ ] 90% of runs complete in ⩽ 60 seconds
- [ ] Every event in the Machine contract has been seen in real runs, including each error type

Front-end and animation design

Recommendation: take a hybrid route. AI tools handle exploration, the page layout and a working placeholder machine. A human animator is hired for the one thing AI tools still do poorly: a single, consistent character that moves well across many states. Commission the animator only after the quality gate passes.

What AI tools do well here

- Concept exploration: image generators produce dozens of machine styles quickly for a moodboard.
- Page layout and UI code: Claude can build the result card, bin, gadget shelf, meter and forms in React from the Machine contract.
- A placeholder machine in SVG with code-driven animation (dials spin, needle moves, card drops out). It is wired to every contract event, so the whole experience works end to end before the real art exists. It may even prove good enough for launch.

Where a human designer earns the fee

- One recognisable character that looks the same in every state. Image generators drift between pictures, and they produce stills, not rigged animation.
- Timing and personality: the cough, the smug pause, the satisfying "ding". This charm is what makes people share it.
- Smooth transitions between states driven by live events, which needs a proper state-machine animation, not separate clips.

Deliverable format that fits the stack

Deliverable	Format	Why
Animated machine	**Rive** file (`.riv`) with one state machine called `Machine`	Interactive, reacts to live inputs, small file; React runtime available (`@rive-app/react-canvas`)
State-machine inputs	Number `stage` (0–6), number `fuel` (0–100), booleans `asleep` and `serious` , triggers `question` , `retry` , `success` , `error` , `tip`	Maps one-to-one onto Machine contract events
Fallback (if the designer doesn't use Rive)	**Lottie** JSON, one file per state, plus a transition plan	Widely supported; less interactive
Static art	SVG: machine, logo, favicon, social share card	Sharp at any size; also used for reduced-motion users

Deliverable	Format	Why
UI design	Figma file with colours, type and spacing as named styles	Translates directly into Tailwind settings
Sound (optional)	Short MP3/OGG clips with a commercial licence	The ding, clanks, purr

The designer brief must include

- The Machine contract table, with the animation wanted for each event.
- The persona notes (dials for a face, small chimney, ticker tape; curious, straining, smug).
- The input names above, spelled exactly.
- A reduced-motion version: static poses instead of movement.
- Full commercial rights and all source files (the Rive or After Effects project, the Figma file). Check this explicitly on Fiverr, where some gigs charge extra for source files or commercial use.

Where to hire and rough cost. Fiverr works for this; also look at Contra and Dribbble, and filter for designers who show Rive or Lottie work in their portfolio. As a rough guess, a character with about 10 animated states costs in the low hundreds of pounds; get 2–3 quotes against the same brief. Ask for one paid test state first (e.g. idle and sleeping) before committing to the full set.

Can it be done with AI tools alone? Yes, to a decent standard: the SVG placeholder machine is exactly that route. Build it first, put it in front of a few people, and hire a designer only if it feels flat. Then the design money goes where the feedback says it matters.

Success metrics, roadmap and decisions

v1 succeeds if users trust the result enough to act on it, and enjoy getting there.

Success metrics

Metric	Target (initial guess)
Runs that complete without error	⩾ 95%
Results rated helpful (thumbs up)	⩾ 70%
Average complexity reduction	40–70%
Users who export or share a result	⩾ 30%

Metric	Target (initial guess)
Time from submit to reveal (5-page input)	⩽ 60 seconds
Results flagged "binned something important"	⩽ 10%
Monthly API spend	⩽ £5

Roadmap

1. **Prototype** — the full engine (all six stages and every Machine contract event) on the test bench, plus a plain page and the AI-built SVG placeholder machine. Gate: the quality gate in Quality first; animation and design are commissioned only after it passes.
2. **v1 "The Machine"** — file upload, dials, sacred cows, the Bin with un-bin, Complexity Score, the machine as an animated character, Serious mode, Markdown/PDF export, power meter and monthly ration.
3. **v1.1 "Show and tell"** — flowcharts, share links, image cards, badges, clarifying-questions mode, community tip jar and public Fuel log.
4. **v2 "The Workshop"** — diagram and whiteboard image input, saved projects and client-ready exports for consultants, version history, "what changed" between runs, integrations (Confluence, Notion, Google Docs).

Decisions

- [x] ~~Who is the primary audience at launch: individuals, teams, or consultants? - Individuals, with perhaps some consultants actually using it seriously over time.~~
- [x] ~~Free tool, freemium, or paid from day one? - Free, but need to consider the actual costs of this. Happy to spend a few pounds a month on it, but it can't become a financial drain.~~
- [x] ~~Final persona and art style: owl, gnome, or the machine itself as the character? - The machine itself.~~
- [x] ~~Should the tool also suggest *automations* (e.g. "use a form instead of email"), or stay purely subtractive? - It should suggest potential automation and use of AI but it should keep this seperate from the overall process.~~
- [x] ~~How much should the Complexity Score weights be user-adjustable? - I don't think they should be, but I think some sort of harshness mode selection would be useful.~~
- [x] ~~Do we need on-premise or private-cloud options for regulated customers? - No.~~
- [x] ~~Name: "The Decomplexifier" vs alternatives (e.g. "Occam's Razor-ma-tron")? - Decomplexifier.~~