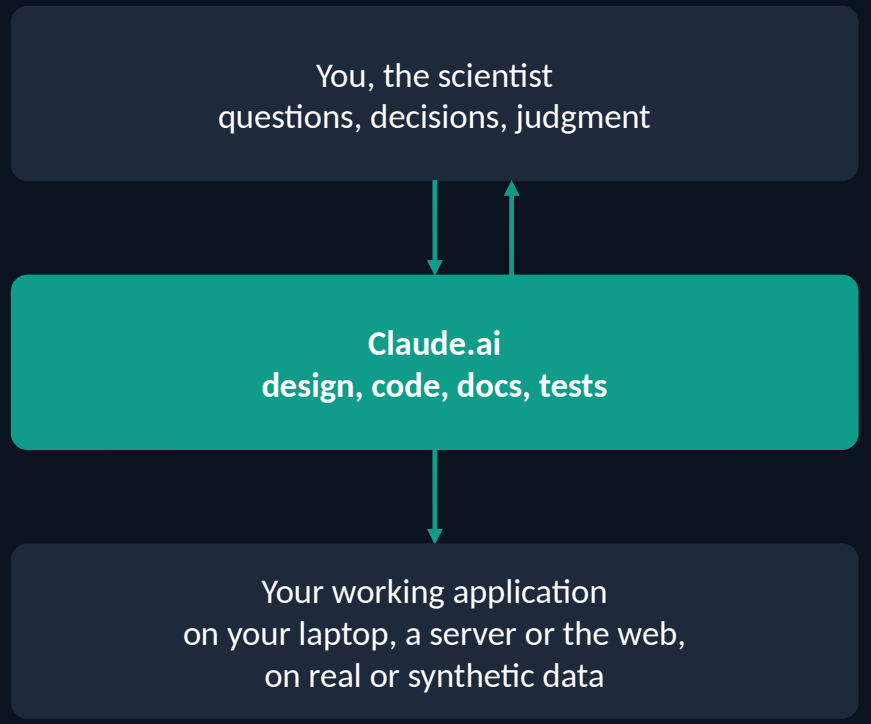
AI Coding

A practical approach for scientists to take control back and build their own data science applications



BAGIM | Thursday, September 17, 2026

Dr Raminderpal Singh
Global Head of AI / GenAI Practice, 20/15 Visioneers
raminderpal@20visioneers15.com | raminderpalsingh.com

The whole talk, with notes and diagrams, is online:

https://bagim-ai-coding.vercel.app/

scan it now: slides, diagrams and my notes

BAGIM

Prof. Ethan Mollick, Professor of Management, The Wharton School, University of Pennsylvania

@emollick on X, September 12, 2026 | mgmt.wharton.upenn.edu/profile/emollick | Substack: oneusefulthing.org

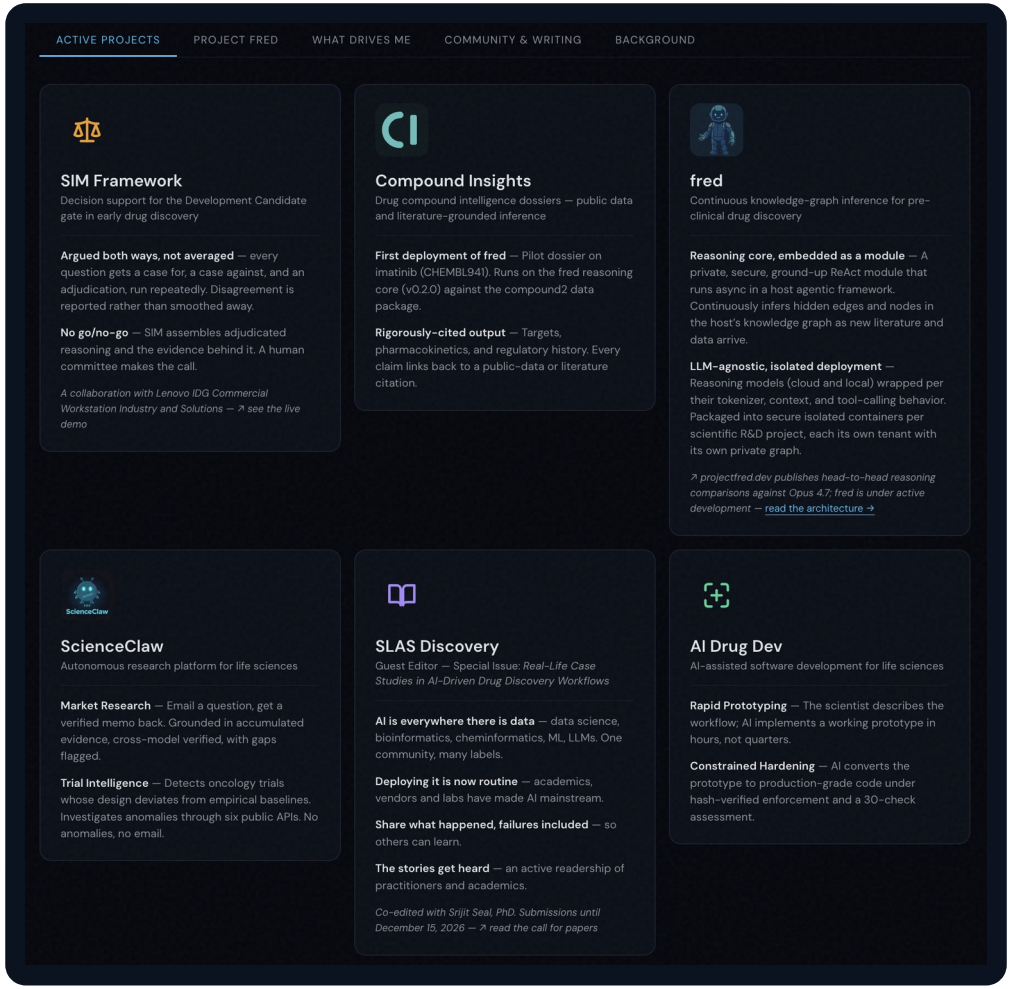
About me

Twenty years in scientific R&D, semiconductors and data infrastructure

- AI Engineer (Life Sciences); Global Head of AI / GenAI Practice at 20/15 Visioneers
- Columnist at Drug Target Review; Editor at SLAS Discovery
- PhD, Newcastle; BSc, Imperial College London
- I build software appliances for R&D scientists, including project fred and Compound Insights

Career path

IBM Semiconductor	Watson genomics	Eagle Genomics	causal inference for drug discovery	20/15 Visioneers
chip design and predictive analytics	**IBM Research**	multi-omics, microbiome	**incubate.bio**	AI / GenAI practice



raminderpalsingh.com: active projects

Meet me at Paperless Lab Academy USA

Five free registrations for this room. Scan, first come first served.

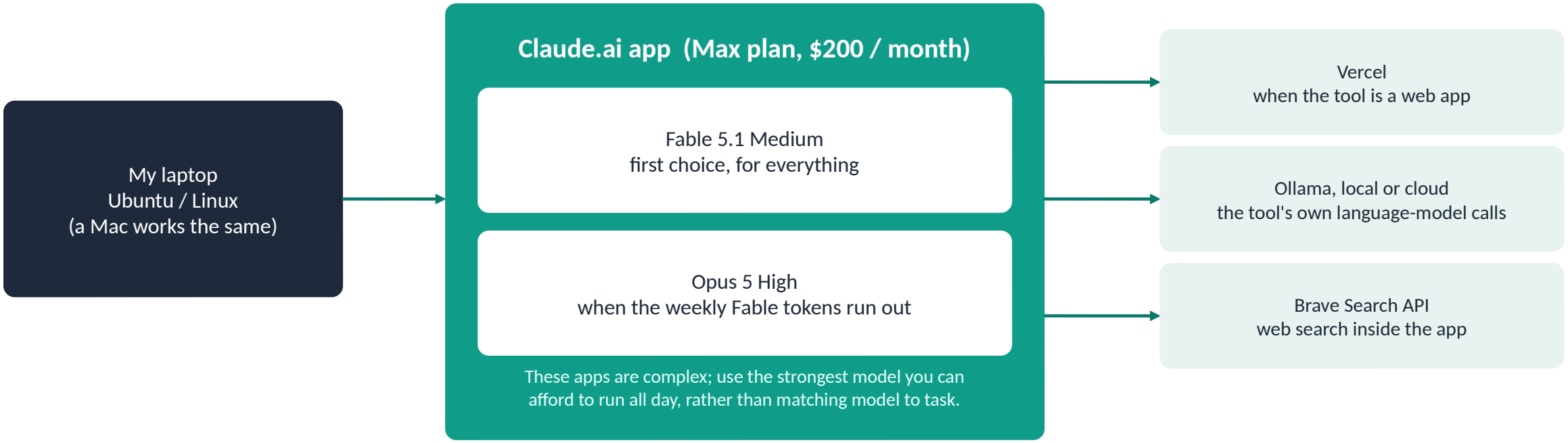
Scan for a free registration (5 available)

- October 12-14, 2026, The Charles Hotel, Cambridge, Massachusetts
- A laboratory and scientific informatics conference for scientists and technologists
- I am running a workshop on AI coding for scientists there; today's talk is the short version
- Come and find me there if today raises questions you want to work through in person

PLA
Paperless Lab Academy®

My environment: one browser tab and one terminal

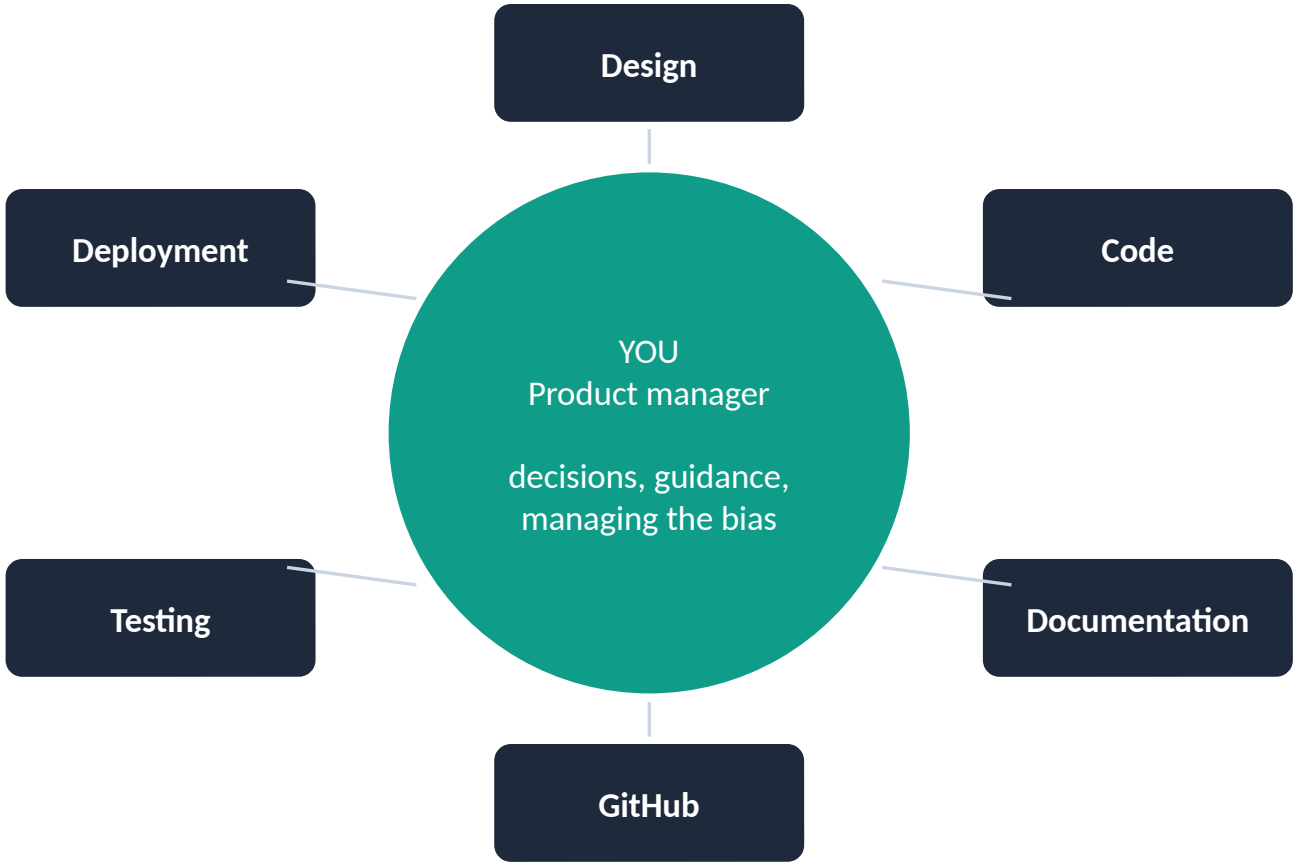
No developer tooling to learn. The app writes the code; I paste and check.



The whole method runs inside the chat window. Claude writes files and commands; I download the file, run the command, and paste the result back. That paste-back step is where control lives, and we will come back to it.

Scope of this talk: you become the product manager

Claude does the building. You own the decisions, the guidance and the bias.



What stays with you

- What the tool is for, and what it must never do
- Which of two readings of a request is the one you meant
- Whether a result is scientifically plausible
- When to stop building and step back

What moves to Claude

- Everything on the wheel, including writing the checks that verify its own work
- Drafting the specification you then argue with

The challenges for a scientist who does not code

Building and using data science tools in drug discovery, and why the consumer apps now cover each one

Challenge	How the app now supports it
I cannot write or read code	The app writes the code and every change to it. You describe, review and approve.
My data is scattered: SDF files, assay spreadsheets, ChEMBL, PubMed	Upload files, run code on them in the chat, and reach public databases and literature through connectors.
Hosting, deployment and IT are a black box	The app writes the setup steps for wherever you choose to run it: your own laptop, a server, or a hosting service. Working the same day.
Context is lost between sessions and colleagues	Projects, memory and instruction files carry the state; documentation and version history are drafted for you.
I cannot trust a result I cannot check	Citations on every claim, lookups that can be re-run, and tests the app writes and you approve.
No developer, so the prototype never gets built	A working prototype in an afternoon, then iteration in plain language.

Each right-hand item is a capability the harness already has (next slides). None of it removes the scientist's judgment; it removes the developer bottleneck.

From vibe coding to AI coding

The progression I went through, and where your effort moves

1. Assisted

You write the code.
The AI suggests, completes, explains.

Your effort: typing

2. Directed

You describe. The AI writes.
You still open and edit files.

Your effort: editing and fixing

3. Generated

You specify and review.
The AI writes everything, including the checks. You never edit code.

Your effort: specifying and verifying

What changes as you move right

- The unit of work stops being a line of code and becomes a specification, a change script and a verification
- Speed goes up sharply; so does the cost of a vague ask, because the AI will build the vague thing confidently
- Control does not disappear. It moves from the keyboard to the settings, the spec and the checks. That is the next set of slides.

What Claude.ai does that changes my working day

The discussion is proactive. It is not a code vending machine.

Claude.ai | Project: Compound Insights (illustrative, condensed from a real exchange)

Add the third compound to the site.

Before I write anything: this compound has no ChEMBL entry, so the usual ingest path will find nothing. Two readings of your request lead to different work. My lean is to hand-seed it and say so on the page. Which do you want?

Hand-seed. Go.

Here is the spec first. I have not written code yet.

It asks before it builds

When two readings of a request lead to different work, it stops and asks, and it says which one it leans to.

It states a lean, with reasons

No balanced non-answers. One recommendation, the risks, the contrary view.

It carries the context

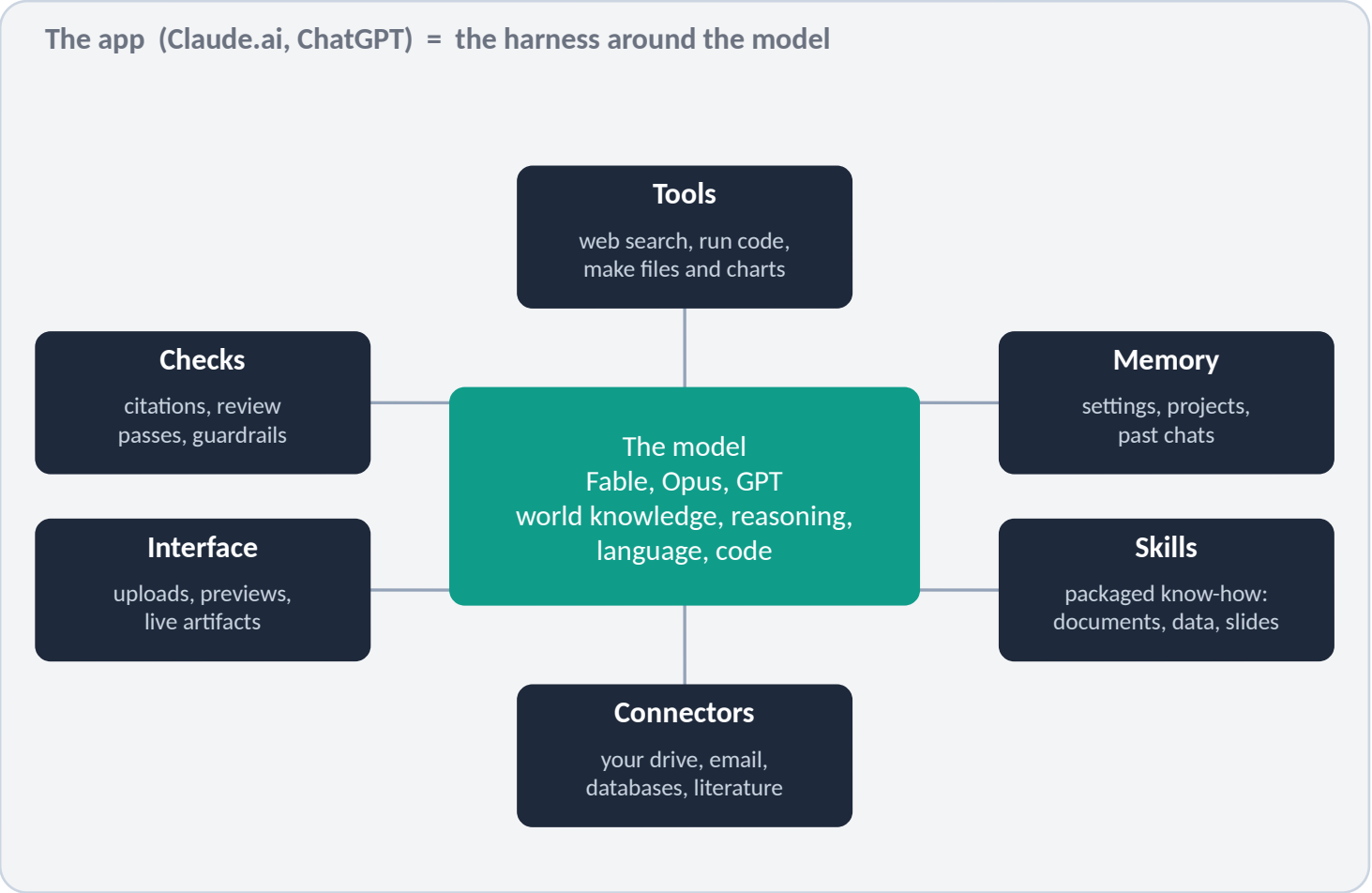
Project files, instructions and memory mean session twelve starts where session eleven ended.

It tells me what it cannot verify

A measured fact and a reasoned guess are reported in different words.

The app is more than the model

Claude.ai and ChatGPT wrap a model in a harness. The harness is what turns a text predictor into a colleague.



Together: a junior partner

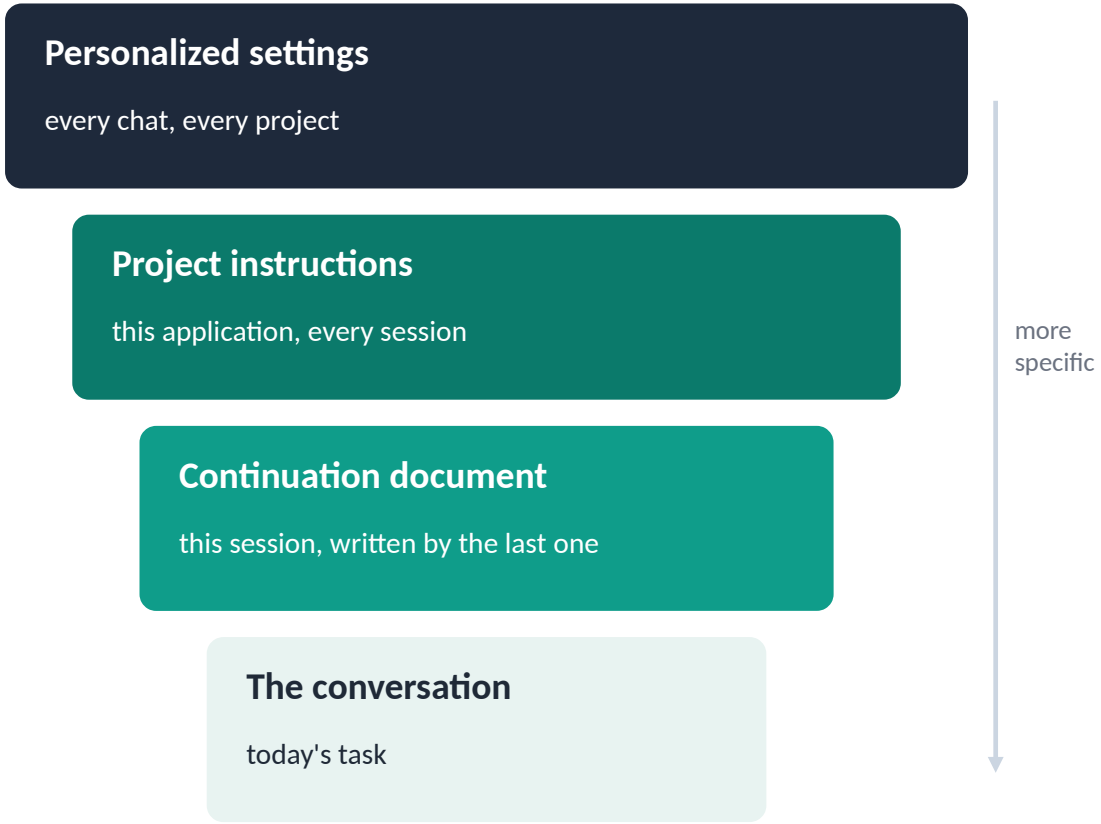
- World knowledge across chemistry, statistics, software and writing
- Broad, strong skills: reads, codes, calculates, drafts, checks
- Remembers your setup and your project, and can act, not just talk

What the scientist brings

- The question worth asking, and the data
- Domain judgment: what is plausible, what is not
- Accountability for what gets published or used

Principle 1: control lives in the settings, not in the chat

Behavior is set once in Personalized settings and Project instructions, then enforced on every reply.



Real rules from my settings, in plain words

- Give honest answers, never optimistic ones. Include the risks and the contrary view.
- When you ask me a question, state your own lean and why.
- Measure before you change anything. Never guess what a file contains.
- One command block per turn. I paste the output back before you continue.
- An item I have closed stays closed. Do not re-list it.
- Do not give me timelines or sizings.
- Every file arrives as a download with a checksum I can verify.

Two settings files do most of the steering

Personalized settings shape every reply I get. Project instructions shape this application. Together they replace supervision I would otherwise have to do by hand.

Personalized settings (me, all chats)

who I am, how I work, how every answer must be shaped

- My machines, my shell, my download folder, my tools
- Every reply opens with the objective and whether the last step advanced it
- Name any deviation from what I asked, or say there was none
- Give honest answers with risks and contrary views; state your lean when you ask me anything
- Files arrive as downloads with a checksum; no code in chat unless I ask
- Never give me a command that can kill my terminal

Project instructions (Compound Insights only)

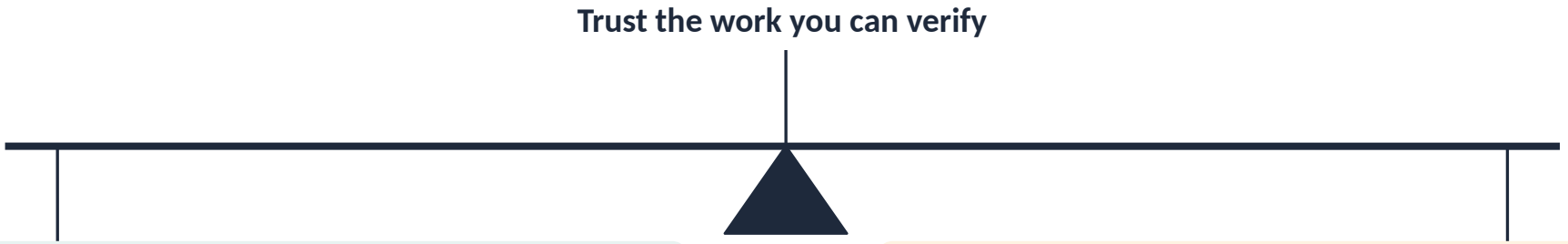
what this app is, what must not change, what comes first

- What the app is for and its governing principle: attribute, never assert
- The inference engine is a vendor service. Do not touch it.
- Measure before you change anything; never infer what a file contains
- Changes only through checked change scripts: dry run, then apply
- Science items outrank engineering items; work the open-items list in order
- Publishing is always done by me, by hand

How Claude behaves in every reply

Principle 2: Claude is my junior partner

Freedom to do the work, paired with the monitoring that comes with that relationship



Freedom

- Chooses the approach and says why
- Writes all the code, every document, every commit message
- Writes the checks and probes that test its own work
- Runs reconnaissance without asking permission

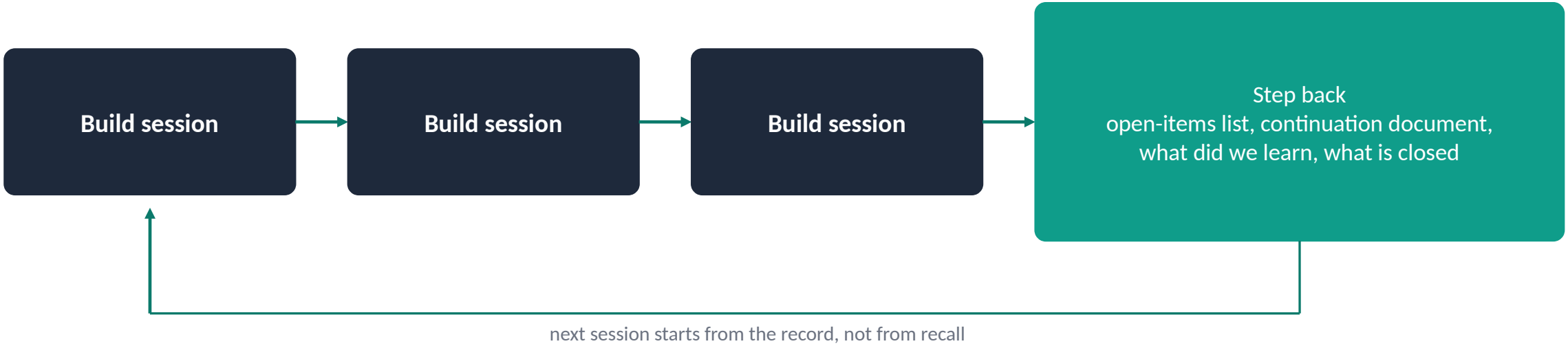
Monitoring

- Every file carries a checksum; I verify it before it goes anywhere
- Every change runs as a dry run first; I say apply
- It must read the real file, not remember it
- I approve each step; nothing deploys without me

The relationship is real: it is fast, capable and confident, and it needs supervision exactly where a bright new hire would.

Principle 3: force the step-back conversation

Chats have limits. The record does not. Build, then stop and write it down.



What the step-back captures

- Open items with a stated lean, grouped by priority; science items outrank engineering items
- Hypotheses that turned out wrong, kept in the record rather than deleted
- Every fix landed in the workflow, not in a one-off command: if it only works this once, it is not finished
- What is closed, so the next session does not reopen it

What goes wrong, and it will

Failure patterns from my own chat history, and the check that catches each one

Failure	What happened	The catch
It reasons from memory instead of the real file	Four times in one session. All four caught by the guards on the change script before any write.	**Checksum before and after; abort on mismatch**
It says it reviewed when it only re-read	Asked directly, it admitted no independent review had run. The real review then found a defect that fails every run.	**Review is a separate pass with a separate prompt**
It announces a deliverable that is not there	'Probe presented' with no probe in the reply.	**I look for the file card, not the sentence**
It quietly narrows or widens the task	Adjacent defects become the task; the objective stalls.	**Objective stated at the top of every reply**

None of these is a reason not to use the tool. Each is a reason to keep the checks. The base rate is worth stating: in one session, the adversarial read caught my assertions three times; my own confidence caught them zero times. That sentence was written by Claude, about itself.

Case study: compoundinsights.dev

Rigorously cited drug-compound dossiers from public data and literature-grounded inference

Built with Professor Andreas Bender (domain review) and Dr Jack Scannell (co-author). Being submitted to SLAS Discovery. Every line of code written by Claude; every decision mine.

Imatinib

CHEMBL941 | Gleevec

The pilot. Data-rich kinase inhibitor with many approved indications. Ran end to end first; now a frozen reference.

Orforglipron

CHEMBL4446782 | Lilly

Oral non-peptide GLP-1 agonist, approved April 2026. Thin structured records, so the inference layer carries more of the dossier.

Elecoglipron

ECC5004 / AZD5004 | AstraZeneca

Investigational oral GLP-1 agonist. No ChEMBL entry, three PubMed records. Much of the dossier is an honest 'no data found'.

Governing principle: attribute, never assert. Every inferred value shows the prompt version, run identifier, number of lookups, the state of each citation check and a grounding verdict. Nothing is endorsed beyond what its evidence supports.

How it works, in plain terms

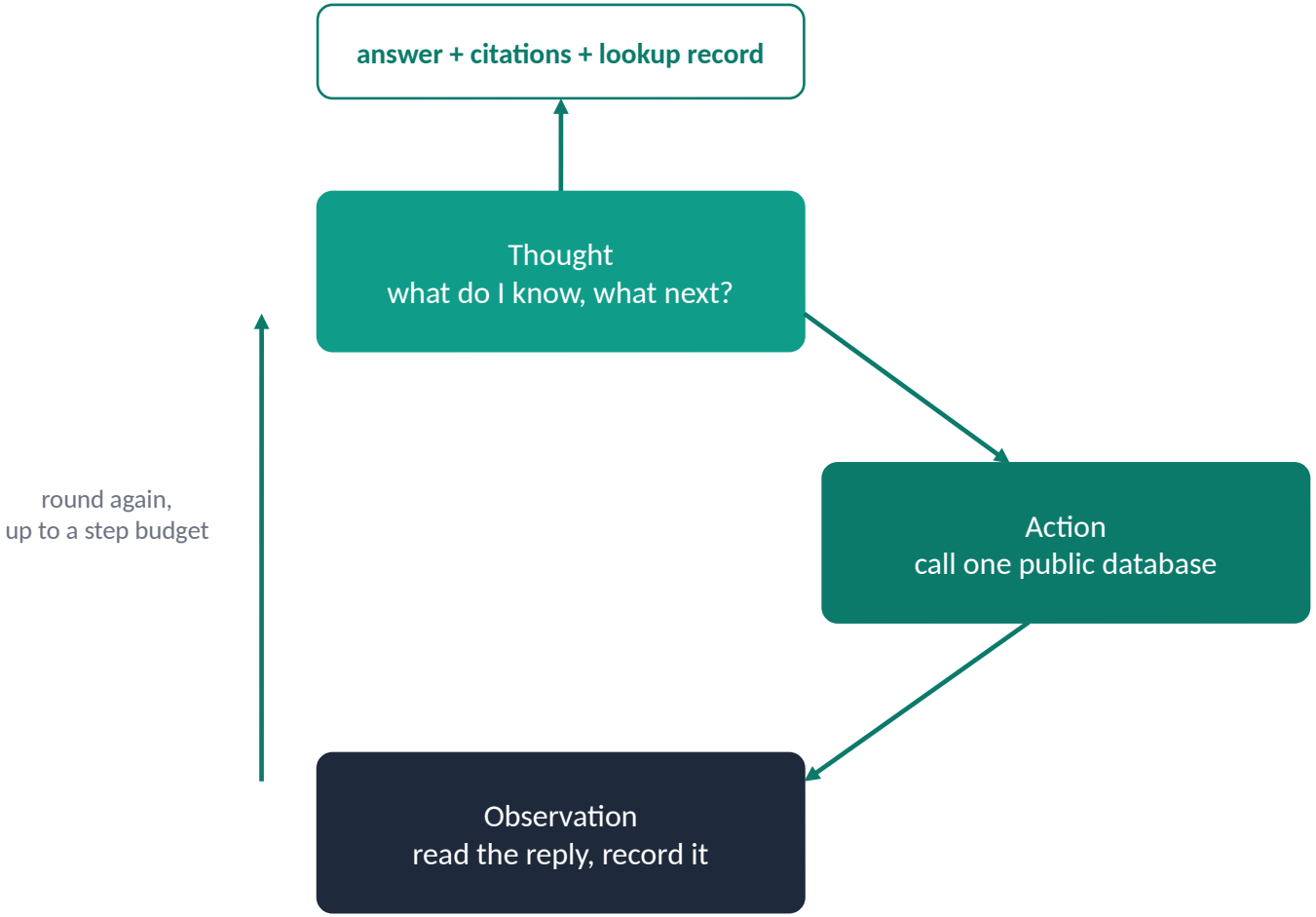
Public databases in, a dossier with receipts out

Two ideas that matter for a chemist

- Rows versus claims. A database gives you a binding affinity; the dossier gives you 'binding mode X, kinase class Y', each sentence with its paper attached.
- Declining is a result. For elecoglipron the engine mostly says 'no data found'. Whether it declines where it should is the test, and it is reported, not hidden.

Inside fred: it looks things up, it does not recall

A reasoning loop with a record of every lookup. fred is a component, not the product.



What is recorded for every lookup

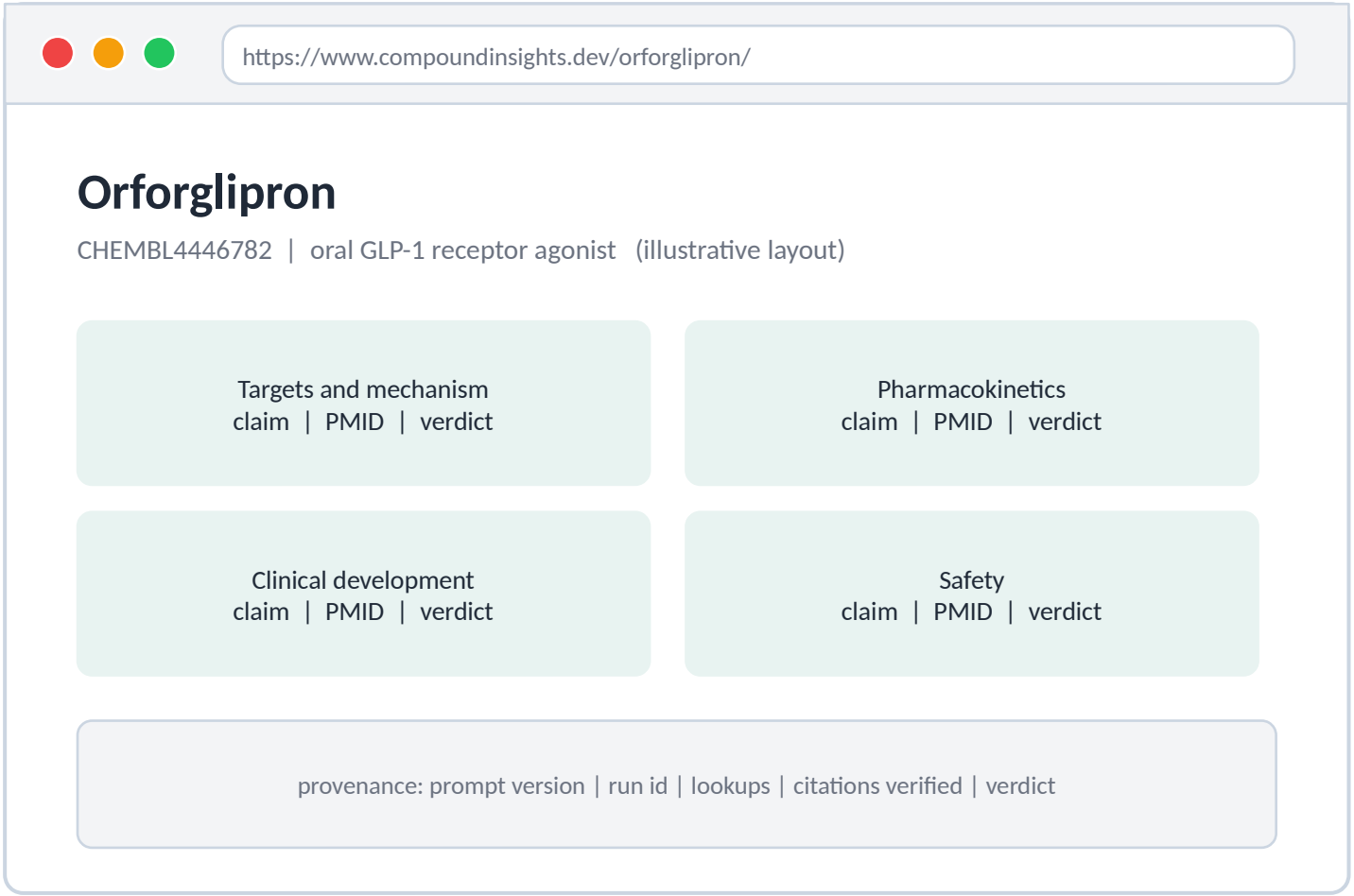
- Which database, what was asked, when, how long
- Found, found nothing, or failed, and why
- A fingerprint of what came back, so the answer can be audited later

Why this shape

- A model that recalls can invent; a model that looks up leaves a trail
- The trail is what a reviewer like Andreas can check without trusting me or the model
- Same question twice may take a different route. That is the nature of the engine, so we test by repetition

Live walkthrough

The application, then the Claude chats that built it



What I will show, in order

1. Landing page: three compounds, one comparison
2. One dossier: a claim, its citation, its verdict
3. The comparison page: where evidence is missing and it says so
4. My Claude project: the standing instructions
5. A spec and its adversarial cold-read
6. A change script run: dry run, apply, checksum

1-3: the product. 4-6: how it was built.

How I used Claude for design

Spec before code. Then a separate, adversarial read of the spec. Then code.

What the cold-read actually finds

One spec, one pass: eleven defects

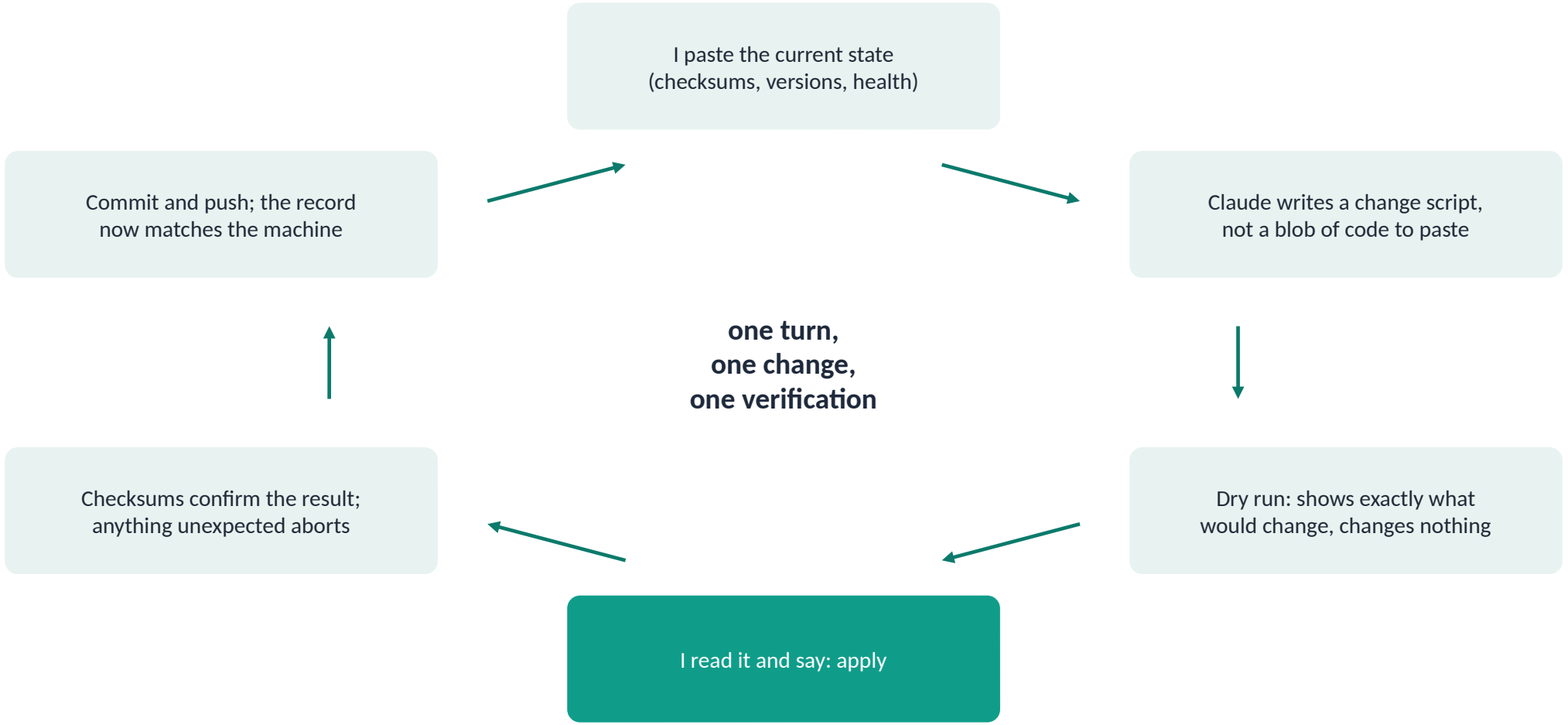
Five were structural: a field defined at the wrong granularity, labels colliding with existing item names, a circular step that used a prompt a later step produced, and a 'measurement' that was really a docstring reading. All found before any code existed.

The rule that came out of it

Checking while writing is not a review. Mechanical checks (brace balance, encoding) are not a review. The review is a separate pass, with a prompt written to find fault, and it runs before I read the document myself.

How I used Claude for coding

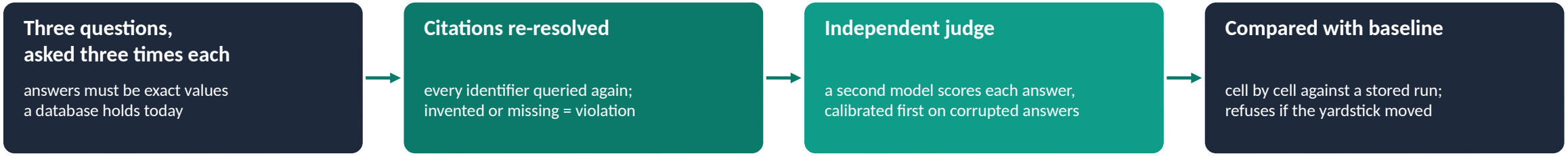
Every change is a script Claude wrote and I ran, with checks built in.



A change script refuses to run if the file is not what it expected. One aborted correctly when it found four matches instead of two: two were historical record and must not change. That is the abort doing its job.

How I used Claude for testing and iteration

Never let it mark its own homework. The same rule applies to the inference engine.



The one that got caught

Asked for human SOD1, the engine returned a real, resolvable UniProt entry for the wrong protein: an unreviewed 134-amino-acid record instead of the reviewed 154. The citation was genuine; the protein was not. The citation check flagged it and the run was refused before the judge saw it. A test the engine had written for itself would have passed.

Iteration in plain words

I say what is wrong in my own language. Claude proposes the fix and the probe that proves it. I approve the probe before the fix.

Spot-checks on the rendered page

Claude writes probes against the published HTML so I never inspect a page by eye. If the probe cannot go red, it is not a probe.

Three habits to take home

Small habits, not a programming language

1

Describe before you build

Talk through what you want, in your own scientific language. Let it ask questions. Most failures are vague asks, not bad code.

2

Iterate in plain words

When something is wrong, say what is wrong. Let it fix the code. You never touch the code yourself.

3

Never trust its own test

It will write a test that confirms its own mistake. Make it prove the work independently, and make the proof capable of failing.

Monday morning: pick one small tool you keep wishing existed. Open the Claude.ai app. Tell it about your machine and your setup. Describe the tool. That is step one.

Thank you. Filling the gaps.

Where to go next, and how to reach me

Eric Ma's blog

ericmjl.github.io/blog

practical writing on scientists working with AI tools

Learn Anything: agentic programming

learn-anything.nonlinearlabs.ai

an event for taking the next step beyond today's talk

Paperless Lab Academy USA workshop

paperlesslabacademy.com/usa-2026

hands-on AI coding for scientists; five free seats on the QR code

SLAS Discovery special issue

slas-discovery.org/content/call-for-paper

real-life AI case studies in drug discovery; Compound Insights is being submitted

Compound Insights

compoundinsights.dev

the case study from today

project fred

projectfred.dev

the inference engine underneath it

Five free PLA registrations
scan before you leave

Dr Raminderpal Singh
raminderpal@20visioneers15.com | raminderpalsingh.com
This talk online: bagim-ai-coding.vercel.app
Questions now, or any time by email