Signal to Strategy

Session 1 — Automate the Signal

Working guide · Wednesday September 9, 2026 · 12:00–1:30 ET Heather Gawel + Babajide Okusanya · Momentum Product Co.

What you'll walk away with

A signal intake running on your own customer inputs, and your AI augmentation audit started. Not a framework you'll get to later — something that ran today, on your data.

Everything here is yours to keep and reuse. The prompts are written so you can copy them straight out and change the bracketed parts.

Before we start

What you need

A paid AI subscription — Claude or ChatGPT. The six steps work in both. We run the live build in **Cowork**, Claude's desktop tool that works directly with files on your computer — no terminal, no copy-pasting transcripts. If you're on Claude, get the desktop app open before we start. **On ChatGPT?** You'll run the same six steps with files uploaded to a Project instead of a folder on disk. Steps 1 through 5 transfer cleanly. Step 6 becomes a saved prompt or a Project instruction rather than a folder you point at.

One pile of customer signal you already have and never get through. A folder of call recordings, a ticket export, last quarter's NPS, a Granola folder you haven't opened. It doesn't need to be clean.

One real decision you're carrying. You'll use it next week and it helps to have it in mind today.

If your work laptop won't let you get to real data, that's common and it isn't a problem. Say so in the chat. We'll give you the Launchpad pack and you'll run the same workflow on that.

No customer PII. You don't need it, and it slows you down.

The premise

Building got cheap. Everyone in this room can ship a prototype this afternoon that would have taken a quarter three years ago. That's real, and it isn't an edge — it's true for everyone you compete with too.

What didn't get cheap is deciding. Which problem, whose problem, worth what, and can you defend it in a room.

So the four weeks go **Signal** → **Decision** → **Strategy**:

WEEK	WHAT YOU BUILD
1	Automate the signal — your own customer and market inputs, in one intake loop
2	Stress-test the bet — validation prompts for a real decision, tuned per stakeholder
3	Defend the call — ROI measured the way your org already defines success
4	Close the loop — in-market monitoring on the tools you already run

The rule for all four: we build on **your** product, **your** stack, **your** data.

Part 1 — Where your signal comes from

Before anything else, three things in the chat:

The signal source you actually use

The one that's piling up and you never get through

The last decision you made mostly on gut

Write your own answers here so you have them for the hands-on block:

I actually use	
Piling up, never get through	
Last gut decision	

Most rooms sort into five buckets: **conversations** (interviews, sales and CS calls, recordings), **complaints** (tickets, escalations), **numbers** (usage, funnels), **surveys** (NPS, CSAT, churn reasons), and **ambient** (community, reviews, competitor noise).

Almost everyone's number two is a pile of conversations. Unstructured, high value, and nobody has time. That's the pile that got cheap to process this year, and it's the one we're pointing at today.

Part 2 — The AI augmentation audit

Why this comes before the build: you can't automate signal well if the same instinct has already quietly automated your judgment.

The frame

Twelve product jobs. Score each one twice.

Can AI do this? 1 = no. 3 = a solid first draft you'd build real work on top of. 5 = better than you.

Should AI do this? 1 = never delegate the decision. 5 = you barely need to be in the room.

Plot them against each other:

	SHOULD: LOW	SHOULD: HIGH
Can: high	**Trap** — it'll do it, and you'll lose the thing you're accountable for	**Automate** — hand it over Monday
Can: low	**Leave it** — not now, revisit	**Queue** — you would, the tools aren't there yet

The subtests — what stops this being a vibe

Should — the hard half

Judgment depth — how much of this is a call rather than a task?

Reversibility — what does a wrong answer cost, and can you undo it?

Accountability — whose name is on this in a room full of executives?

Presence — does the value come from a human actually being there?

Can — the easy half

Input verifiability — can you check whether the answer is right?

Context — does it need things that live only in your head or your org?

Volume — is there enough of it that a machine actually helps?

What we found

The first time we scored our own twelve jobs, all twelve came back **automate**. That wasn't a goldmine. That was us being generous on *should*, because we wanted the answer to be yes.

Rescored honestly, the pattern has held across nearly every team since:

Automate — insight synthesis, requirements and specs, prototyping, GTM and launch, performance monitoring. Production work. High volume, reversible, and nobody signs their name to a research repo.

Trap — prioritization, strategy and positioning, pricing and packaging, experiment design, stakeholder alignment. Decision work. AI is good enough at these that you won't notice it eating your judgment until you're in a room defending a roadmap you can't explain.

AI will print you a beautiful prioritized roadmap. You still can't stand in front of your exec team and say the model ranked it.

Score it live

The worksheet is at momentumproductco.com/signal-to-strategy/audit — the link is in the chat too. In session, do these three so we can compare — **insight synthesis**, **prioritization**, **strategy and positioning** — then pick one more that's eating your week.

Fill in hours per month and pain. Those two are what turn this from an interesting chart into a decision: your pick is the highest *hours × pain* sitting in the Automate quadrant. Not the most interesting job — the biggest one you lose nothing by handing over.

Before next Wednesday: finish all twelve and post your trap quadrant in the cohort channel.

Part 3 — Live build: signal intake from scratch

Six steps, about 40 minutes. Copy each prompt into Cowork as you go. **The coaching callouts are the judgment moves** — they're what turn these prompts into a real decision rather than just output.

Frameworks at a glance: Opportunity Solution Tree is the **default** everyone runs in Step 4. Jobs-to-Be-Done is the **alternate**. The **designer track** (How Might We + journey map) is a third option. The problem-prioritization matrix, hypothesis-driven brief and segmentation map live under **Other lenses** at the end.

The scenario

You're at **Launchpad**, a B2B onboarding and enablement platform that helps SaaS companies guide customers from signup to first value. Leadership has set a goal: annual logo churn from 14% to under 10% by end of year.

You have three kinds of signal and no time:

```
launchpad-signal-data/
  01_Sarah_Chen_Feb2026.md  …  10_Elena_Vasquez_Mar2026.md   10 interview transcripts
  support_tickets_Q1_2026.csv                               75 tickets
  nps_Q1_2026.csv                                           120 responses
  launchpad_company_overview.md                             the business context
  PROMPTS.md                                                this prompt pack
```

Conversations, complaints and surveys — three different shapes of the same customer.

Step 1 — Set your context (2 min)

Tell Claude who you are, what question you're answering, and how to structure the analysis.

PROMPT · COPY THIS

Tip: if Claude's reply could apply to any SaaS company, your context isn't specific enough.

```
I'm working at Launchpad, a B2B onboarding/enablement platform that helps
SaaS companies guide customers from signup to first value. Our annual logo
churn is 14% and leadership wants it under 10% by end of year. This Cowork
workspace has our Q1 2026 discovery data: interview transcripts, support
tickets, NPS responses, and a company overview.

Help me synthesize this to find the highest-leverage opportunities to reduce
churn. When you analyze it:
1. Find patterns ACROSS all sources, not just within one.
2. Surface contradictions or tensions between segments.
3. Distinguish high-frequency complaints from high-impact ones.
4. Tell me what's notably ABSENT — and whose voice is missing or
   under-represented in this data.
Don't build a framework yet. Read everything first and confirm the goal.
```

The last line is the one doing the work. It stops the model performing analysis before it has read anything. Watch what comes back: it tells you what it has access to, and it flags that the interview sample skews toward people still talking to you. Neither of those was asked for.

Step 2 — Load the data (3 min)

In the starter the data is already here — just point Claude at it. *(On your own data later you'd drop a folder into Cowork or connect Gong / Zoom / Drive so transcripts and CSVs come in directly — no exporting.)*

> PROMPT · COPY THIS
>
> **Tip:** start with about five transcripts and add more only if you need to. Patterns get clearer, not just bigger.

```
Read all the interview transcripts in this workspace, then the support
ticket and NPS CSVs. Orient yourself on what each column means. When you've
read them, tell me what NEW patterns emerge when you combine all three
sources — e.g. "the mobile issue appears in 3 interviews, 10 tickets, and
12 NPS comments."
```

Watch it run commands to open the CSVs rather than working from the column names. That's the difference between reading your data and guessing at it.

Then compress before you go on:

```
First, can you summarize the step 2 findings in 10 bullets.
```

Almost nobody does this step. A wall of analysis is not a finding — ten bullets is what actually goes in front of a stakeholder.

Step 3 — Ask for surprises (7 min)

Before any framework, find what you might have missed. **This is the highest-value moment in the whole workflow. Protect it.**

```
Before we build any framework, tell me the most surprising or non-obvious
observations across these three sources — not summaries.
1. What patterns only appear when you cross-reference interviews, tickets,
   and NPS?
2. Where do different customer segments disagree?
3. What's notably ABSENT — and whose voice is missing or under-represented?
4. What's the single most important insight for churn reduction?
```

Push back if the first answer is thin: *"That's a pattern, not a surprise. Tell me something I wouldn't have guessed from reading two transcripts."*

Whose voice is missing? Churned-and-gone customers, the quietly dissatisfied, smaller accounts, non-native speakers. Synthesis that overlooks them makes biased decisions. Ask explicitly who isn't represented before you trust the picture.

Step 4 — Build your framework (10 min)

Opportunity Solution Tree is the default — it produces the most useful output in the time we have. If the **OST skill** is loaded, run it instead of the short prompt below: you'll get user-need framing, experiments rather than solutions, and evidence grounding automatically. Notice the difference between a thin prompt and a real skill. That gap is the whole point.

Opportunity Solution Tree (default)

```
Build an Opportunity Solution Tree for Launchpad's churn-reduction goal.
DESIRED OUTCOME: reduce annual logo churn from 14% to <10%.
Identify 4-6 OPPORTUNITY areas. For each:
- Name it as a USER need, not a company problem.
- Rate evidence: Strong (3+ sources), Moderate (2), or Emerging (1).
- List 2-3 SOLUTIONS framed as experiments with the assumption each tests.
- Note which customer segments care most.
- Include the most compelling supporting quote.
Organize from strongest evidence to weakest.
```

Jobs-to-Be-Done (alternate) — use this if you'd rather understand *why* customers hire or fire the product.

```
Using JTBD, identify the core jobs Launchpad customers hire the product to
do. For each: a job statement ("When [situation], I want to [motivation],
so I can [outcome]"), what "done well" looks like, how well Launchpad
delivers (Strong/Adequate/Failing), which underserved outcomes drive churn
(cite evidence), and which competitor is best positioned to steal the job.
Identify at least 4 distinct jobs.
```

Designer track — design brief / journey map. Turn the synthesis into design inputs instead of an exec summary.

```
Reframe this synthesis as inputs to a design brief. Produce:
1. 2-3 "How Might We" statements grounded in the strongest evidence.
2. A rough current-state onboarding journey map with the key drop-off and
   frustration moments, each with a representative verbatim.
3. The single top unmet user need to design against, and the evidence.
Keep it about the user's experience, not internal metrics.
```

Step 5 — Pressure-test and own it (5 min)

```
Pressure-test this:
1. What's the weakest link? Where is the evidence thinnest?
2. Argue AGAINST the #1 recommendation. What would you say?
3. What additional data would we need before deciding?
Then reformat as a 1-page executive summary for a VP Product meeting:
strategic question, top 3 recommendations ranked by evidence, key risks,
and what to validate next.
```

Don't lose the human. Before you trust a theme, open two or three of the raw verbatims behind it. Averaging 75 tickets into four bullets is useful — and it quietly erases the specific person who was frustrated. Spot-

checking guards against both lost nuance and invented quotes.

Own it. When this goes to the VP, *you* own the recommendation, not Claude. Name the one decision you'd be personally accountable for, and the evidence you'd stake it on.

The one question people forget to ask

At any point, ask it what it actually used:

```
Did you use the analyzing-user-feedback skill for the feedback analysis?
```

In the run we walk through, the answer was: *"No, I didn't. Honest answer."* It had the skill. The skill was relevant. It didn't read it, and it only said so because it was asked.

Then:

```
Just the delta — what changes vs. the current analysis.
```

Leaner than a rewrite, and it shows you the size of the gap between a thin prompt and a real skill.

Step 6 — Make it yours (3 min)

Do one real thing with your own work so the skill leaves the room with you:

Open a new Cowork space for your own product.

Adapt the Step 1 context prompt to **your** product — who you serve and what decision you're making. No customer PII needed.

Point Cowork at one folder of your real discovery data, or connect Gong / Zoom / Drive — then run Step 3 (surprises) on it.

PROMPT · COPY THIS

Your Monday trigger: what discovery data do you already have sitting unsynthesized — a Granola folder, a Zendesk export, last quarter's NPS? Pick one and run this on it.

Other lenses (appendix)

Optional alternatives to the default OST. Use one if it fits the decision you actually need to make.

Problem prioritization matrix

```
Create a problem prioritization matrix. List every problem, then classify:
- FREQUENCY: High (15+ data points), Medium (5–14), Low (1–4)
- SEVERITY: Critical (caused churn), High, Medium, Low
Place each in: DO FIRST / PLAN NEXT / MONITOR / DEPRIORITIZE.
```

Hypothesis-driven product brief

```
Write 3–5 churn-reduction hypotheses. For each:
- "We believe [building X] for [segment Y] will [achieve Z]."
- Evidence FOR and AGAINST
- How we'd measure success
- Confidence: High / Medium / Low (be honest)
- Next step: Build / Test / Research further
```

Customer segmentation + needs map

```
Identify 2–3 segments defined by relationship to the product, not plan tier.
For each: describe, approximate size (% of base), NPS range, top 3 unmet
needs with evidence, churn risk and primary driver, what would retain them,
and a representative quote. Then: which segment should Launchpad prioritize?
```

What the Launchpad data actually holds

Don't read this before the session — it's the answer key. Every number below was counted from the files, not estimated.

The theme that's everywhere and nowhere. Self-serve flow editing is named in **all ten interviews** and drives **fifteen tickets** — every single Feature Request in the file. Search 120 NPS responses for "self-serve" or "builder" and you get **one hit — and it's positive**, about a different feature entirely. Meanwhile mobile gets **10 NPS mentions** and the analytics/reporting/ROI theme gets **13**. Read the survey on its own and the biggest problem in the business does not exist.

The reason is simple: nobody writes *"I want a self-serve builder"* in an NPS box. **Exactly one** person describes the pain at all, and they describe it as a symptom — NPS003, Starter tier, score 6: *"Every change requires submitting a ticket."* That's the whole lesson. The words people use in a survey are not the words in your roadmap, and a keyword search will never bridge that.

The account that breaks your instinct. Vantage HR files **8 of the 15 self-serve tickets in the entire dataset** — more than half of them, and the loudest complainer by a distance. Vantage HR is also **one of the happiest accounts in the data: NPS +72.7, average score 9.0.** Enterprise, and going nowhere. (Only TerraOps scores higher, at +80.) Complaint volume tracks *engagement*, not risk.

The account you should worry about is **NovaCRM: NPS −90.** They file ten tickets like everyone else, but only two are about self-serve, and they say almost nothing in the survey. Quiet and unhappy beats loud and unhappy every time.

Ticket counts are a trap in this dataset. Seven companies file 10 or 11 tickets each. Five are tied at 11. Any "loudest account" claim built on raw ticket volume is measuring nothing. Concentration is the real signal, not volume.

The single strongest number in the data. NPS by tier: **Starter −50** across 30 responses, with **zero promoters** — not one score above 8. Growth **+4**. Enterprise **+73**. The blended score is +14, which is what the company overview reports, so the split is sound. One number, and it tells you where the churn is coming from.

The churn arithmetic. Starter is 34 accounts, 25 of them churned — a **74% churn rate**, and **62.5% of all churned accounts**. Those are different claims; say the one you mean. Growth churns at 41%, Enterprise at 6%.

The CSM correlation, stated correctly. 80% of retained accounts had a dedicated CSM. Only 10% of churned accounts did. It's the strongest signal in the overview — and it's correlation, not causation. CSM coverage is limited to Growth and Enterprise, which are also the tiers that churn least anyway. Say that out loud; it's exactly what Step 5 is meant to catch.

Whose voice is missing. Ten interviews, and only **five are current paying customers**. Two are Launchpad's own staff. One churned, one was a lost deal, one never bought. **Not a single interview states a plan tier** — so the tier that churns at 74% never speaks in the interviews at all. It appears only as thirty one-line NPS comments, and you can't confirm which tier any interviewee was on.

The absence you can't fix. The interview companies and the NPS/ticket companies **don't overlap at all**. Zero accounts appear in all three sources. So no finding here can be traced through one account from conversation to complaint to score — and any claim that does that is invented. Watch the near-miss: **Velocity HR** (interview) and **Vantage HR** (tickets) are different companies — and **Sarah Chen** is a name in both, as the VP at Velocity HR and as a ticket contact at Vantage HR. Two different people. That's the trap most likely to produce a confident false trace in a live run.

One more thing to know about the files. The tickets CSV has six malformed rows — all Implementation Wait, all missing the subject field. It's in the source data. A careful reader catches it; a naive parse mis-attributes those fields. If it comes up, that's a real lesson about checking your inputs.

Part 4 — Your turn (20 minutes)

You will not finish. Something running on your real inputs is the goal.

Lane A — you have data ready. Point it at your folder or connect the source. Run steps 1, 2 and 3. Aim for one surprise you didn't have this morning.

Lane B — you have one source you can reach in five minutes. A ticket export, one folder of call notes, last quarter's NPS. One source is enough to build the habit. Run steps 1 and 3.

Lane C — you can't get to real data right now. Use the Launchpad pack and change the decision: instead of churn, run it as *what should we build next quarter*. Same folder, completely different answer. That's a real lesson about how much the question shapes the output.

Your first output will be mediocre. The second prompt is where it gets good.

Write down the one thing it surfaced:

Part 5 — Real signal or a loud anecdote?

Run every finding through four tests before it reaches a stakeholder.

Frequency — how many times, out of how many? "Six mentions" means nothing. Six out of forty is a pattern; six out of six hundred is noise.

Spread — how many different *sources*, and how many different segments? One angry enterprise account filing nine tickets is one data point that learned to type.

Consequence — what's attached? Revenue, churn, deal cycle, support cost. A theme with no consequence attached is a preference.

Corroboration — does anything independent agree? Behaviour beating words is the strongest form: they *said* it's fine and the usage says otherwise.

And the override: these are a floor, not a ceiling. One sentence from your largest account can outrank fifteen Starter-tier tickets, and no amount of counting will tell you that. AI counts frequency. You decide what matters — which is exactly the judgment that landed in the trap quadrant an hour earlier.

Run the Launchpad data through them and the ranking inverts. **Self-serve** fails frequency badly in NPS — one comment out of 120 — and passes everything else: all ten interviews, fifteen tickets, and it's the stated reason 17 accounts left for Appcues. **Mobile** passes frequency in NPS with ten mentions and thins out fast: six accounts churned on it against seventeen on self-serve. And **Vantage HR's eight self-serve tickets** fail spread on their own — that's one account filing eight times, and it's one of the happiest accounts in the set at NPS +72.7.

Count mentions and you ship a mobile fix. Weigh consequence and you build the flow editor.

Watch for these

Fabricated specifics — invented numbers, dates, quotes. Ask for the source line.

False recency — an eight-month-old ticket read as current. Include collection dates in your prompt.

Overgeneralising from thin data — three comments aren't a pattern. Say the volume out loud in the prompt.

Missing the absence — ask "what are customers *not* saying that you'd expect them to?"

Before next Wednesday

Finish all twelve jobs in the audit. Post your trap quadrant in the cohort channel.

Get your signal intake running on one real source. Post one surprise it found.

Week 2 — Stress-test the bet. You'll take one real decision you're carrying and try to break it: validation prompts tuned to the people who actually push back on you. Your CEO reads risk. Your CFO reads payback. Your eng lead reads scope. Different lens, different prompt.

Bring a live decision. Bring one you're not sure about — the ones you're certain on make for a boring session.

Where things live

Momentum AI Working Group on Slack — join here: join.slack.com/t/momentumprodu-e4×8005/shared_invite/zt-3ydpfrpfb-Y9NpvNBD_nx9eKbtu1dOnQ 200+ product people, including everyone who's come through our workshops. No pitching, no vendors. You keep access after the course ends.

#maven-cohort-fall2026 — a private channel inside that same workspace, just for this cohort. You're already added; it's there once you accept the Slack invite. Post what's stuck there; don't email.

Slides, prompts and the data pack — momentumproductco.com/signal-to-strategy/session-1

Launchpad is a fictional company created for Momentum Product Co. courses. This guide is for course use only.