Great Products Are Like Great Recipes

Most builders skip straight to cooking.

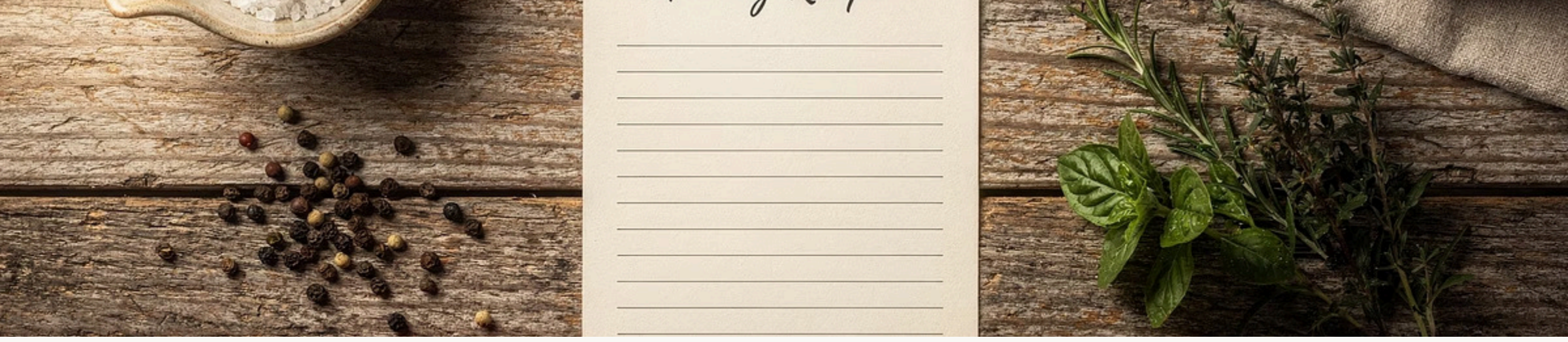
Here's how to get the recipe right first.

Swipe to learn the 6-step process →

THE PROBLEM

They built all weekend. They launched. Nobody ordered. 😬

Most products fail because the builder skipped the recipe. They assumed they knew what people wanted, cooked something nobody ordered, and blamed the market.



THE FRAMEWORK

The 6-Step Recipe 🍽️

01

Find Your Kitchen/ Focus

Pick your space of focus if you have a brioad range of interests

02

Explore Different Menus/ Zoom in

Find what's broken

03

Explore the Hunger and pain

Prove the pain is real

04

Name the "Craving"

Put the pain into words

05

Design the Dish and Recipe

Sketch your solution

06

Check Your Bets and Build

Find what could kill your concept.

These steps are numbered but not strictly linear. Any step can send you back to an earlier one — that's a feature, not a bug.

STEP 1

Find Your "Kitchen" 🔪

Score your broad interest space to converge to one - focus is important. Follow the evidence, not the excitement.

Value Pools

Is money leaking here? is time wasted an issue? is market growing?

Edge

Do you have a secret insight, ingredient or recipe nobody has today?

Why Now?

Is the timing right? are there signals?

Founder Fit

Score high if you already have relationships, a relevant track record, and a channel to reach buyers. Score low if you'd be starting cold.

⚠ If you score low, stop. Either find a stronger kitchen or find a co-founder who raises the score.

STEP 2

Explore Different Menus 📋

You've picked your space. Now find out what's actually broken inside it. Talk to real people, listen hard, and let the patterns tell you where to focus.

1

Talk to real people

Talk to people who are currently experiencing the problem — who are actively dealing with it today. Listen for repeated complaints , frictions etc — and the workarounds they've built. Spreadsheets, WhatsApp groups, manual hacks. Workarounds are the strongest signal that a problem is real.

2

Cluster What You Heard into key themes

Group the problems that came up more than once. For each cluster, note: how often it happens, what it costs them in time or money, and how badly they want it fixed.

3

Pick the Most Painful thing to solve

Score each pain cluster by frequency, cost, and urgency. The highest scorer is your starting hypothesis — cross-check it against your Step 1 scores before committing.

Step 1 chose the kitchen. Step 2 finds what's missing from the menu.

STEP 3

Confirm the Craving 👅

Not every complaint is worth building for. Prove the hunger is real.

Wasted Money

People are already paying for a bad fix.

Bad Fixes

They've built their own workaround. DIY hacks = real hunger.

Frequency

How often does this problem actually happen? A painful, expensive problem that only occurs once a year changes the business case entirely.

> Threshold: if 7 out of 10 people describe the same pain without you naming it first, and at least 3 are already spending time or money on a workaround, the craving is confirmed.

STEP 4

Name the Craving ✍️

Vague hunger = vague recipe. Write it from the user's perspective.

When I am...
The specific moment

I want to...
The outcome they need

But...
What's blocking them

Which makes me feel...
The emotion (or in B2B, the operational consequence — use 'Which means...' instead of 'Which makes me feel...')

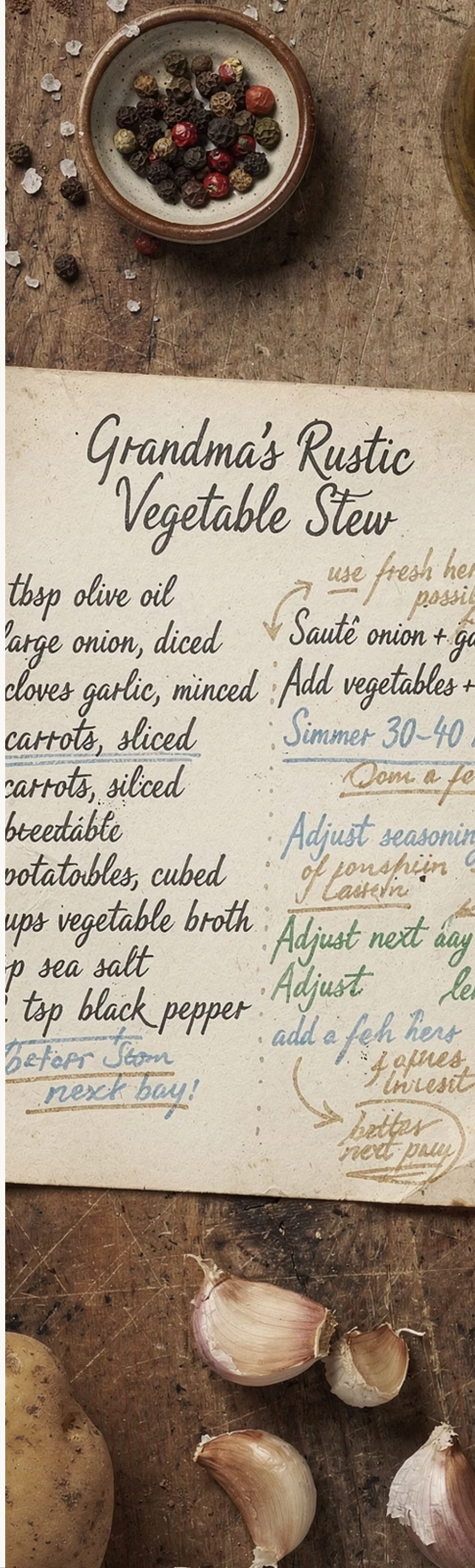
So I...
The bad fix they resort to

Consumer example:

> *"When I get home at 6pm... I want a healthy meal in 30 mins... But the fridge is half empty... Which makes me feel like a bad parent... So I order pizza."*

B2B example:

> *"When I'm reviewing my team's pipeline on Monday morning... I want to see which deals are at risk... But our CRM data is three days stale... Which means I can't flag at-risk deals until it's too late... So I spend an hour calling each rep individually."*



STEP 5 — PART 1

Design the Recipe 👩‍🍳

You're designing a web app or service that solves this better than anything out there. Don't jump straight to building — taste first, then sketch.

Steal Like a Chef

Look at how completely different industries handle the same core need. If your user needs to make a fast decision with incomplete data, look at how ER triage, trading desks, and sports coaches do it. Borrow the best bits, then make it your own.

Then Make It Your Own

Sketch 6–8 rough ideas. For each one: who uses it, what they do, what they get. No polish, just explore.

Gate: don't move to mapping the user flow until you've generated at least 6 concepts and picked the 2 strongest to develop. Narrowing too early is how you end up building the first idea, not the best one.

STEP 5 — PART 2

Map the Dining Experience 🗺️

A user flow is the experience your diner has from the moment they walk in to the moment they decide to come back.

They Walk In

What's the first thing they see? What's the vibe? What do they do first?

They Order

What does the diner ask for? A search, a choice, a preference? This is where they tell your app what they came for.

The Kitchen Delivers

They don't see how it's made — they just get the result. What does your product process, generate, match, or show?

They Eat

Did it satisfy the craving? Was it what they expected? A result, a decision, a saved thing, a next step?

They Come Back

What triggers their return? A notification, a new need, a habit? If there's no return trigger, you have a tool, not a product.

- [ ] Each step should feel obvious. If a diner has to think too hard, they won't come back. This is your core happy path — the shortest route to value for a first-time diner. Edge cases and returning-visitor flows come later.

STEP 5 — PART 3

Now Describe the Whole Thing 💡

Before you build, you need to be able to describe four things clearly. If any of these are fuzzy, the build will be too.

The Product

What is it — an app, a tool, a service? What does it do in one plain sentence?

The User

Who specifically is this for? What do they want, and what's getting in their way right now?

How It Works

What does the user actually do inside the app? What does the app do back? Keep it to the core interaction — not a feature list.

The Outcome

Write one sentence: 'Before, they [old way]. After, they [new way].' If you can't fill in both halves with something specific and measurable, the value isn't clear enough yet.

- [ ] If a friend asked what you're building, could you answer all four in under a minute? The Product, The User, How It Works, The Outcome. That's the test.

STEP 6

What Could Kill the Dish? 🔥

Every recipe has assumptions (future bets) baked in. Find out which ones could kill the dish before you open a restaurant.

1

Your Desirability Bet

Will enough people order this dish? What must be true about their hunger, and why would they choose yours over what they're eating now?

2

Your Feasibility Bet

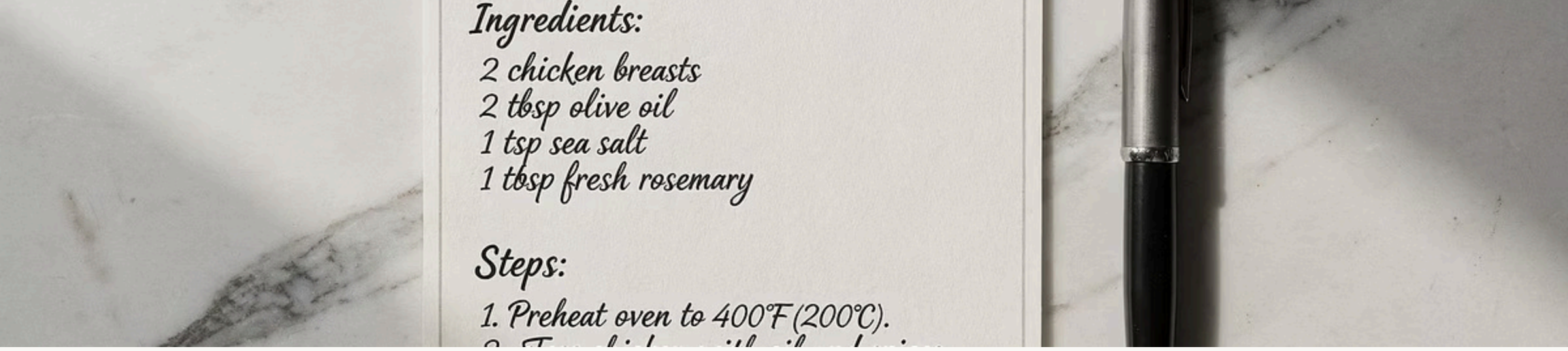
Can you actually cook this? Do you have the skills, the ingredients, and a way to get it to the table?

3

Your Viability Bet

Can you keep the kitchen open? Will people pay enough, often enough, for this to be a business — not just a good meal?

Write your top assumptions that needs to be true for each category. Then rank all three: which one, if you're wrong, kills the idea? Design everything that follows — the Recipe Card and your first version — around answering that one question at the lowest possible cost.



BEFORE YOU BUILD

Write the Recipe Card 📝

This is what you hand to your AI builder — or any builder. Every vague line here becomes an assumption someone else makes for you. Be specific.

1

The Guest

One specific person type with their needs, desires, constraints and attributes. Not 'everyone.'

2

The Dish

One sentence: what it does and for whom.

3

The Dining Experience

Walk In → Eat (in under ___ minutes) → Come Back. What do they see first, how fast do they get what they came for, and what brings them back? Time-to-value is a design constraint, not an afterthought.

4

The Ingredients

Max 3 features. Each must serve the core flow.

5

What's Not on the Menu

Explicitly list what you're NOT building.

Important: If you can't fit it on this card, you're building too much.

CHOOSE YOUR FIRST VERSION

Tasting Menu or Full Restaurant? 🍽️

Your first version answers one question. Pick the format that answers it fastest.

Go back to Step 6 first. Your riskiest assumption determines which format to choose: if the risk is demand, use the Tasting Plate. If it's usability or willingness to pay, use the Pop-Up Kitchen. If you're unsure the solution logic works at all, use the Chef's Table.

🧾 The Tasting Plate

A landing page + waitlist. Tests whether anyone is hungry before you cook and if your dish is what satissfies their hunger.

🍳 The Pop-Up Kitchen

Build the core flow only. Test if people use it and what parts need improving.

👨‍🍳 The Chef's Table

Do it manually for 3–5 real users. No code — just you solving it by hand. This feels slow, but it's where you learn the most — you see the problem up close and discover what your product actually needs to do.

Start with the version that answers your biggest question fastest. You can always add more courses later.

AFTER YOU SHIP

Build → Test → Learn → Repeat 🔄

Shipping is the beginning, not the end. Here's what to do in the first four weeks.

First 2 Weeks: Talk to Users

Talk to at least 5 users. Ask each one: what did you expect to happen, and what actually happened? The gap between those two answers is your most important signal.

First 2 Weeks: Find the Drop-Off

Review where users leave before reaching the core value moment. If more than 60% don't get there, the flow is broken — not the idea. Ship fixes before moving on.

By Week 4: Measure Return

Check how many users come back unprompted. If fewer than 30% return on their own, problem-solution fit isn't there yet. Don't add features — fix the core.

Then: Iterate on the Weakest Signal

Find the single metric furthest from where it needs to be, and fix that first. One thing at a time.

The 60% and 30% figures are starting benchmarks, not universal rules — adjust based on your category and how often the problem occurs. The principle holds: if most users don't reach value, and most who do don't come back, fix the core before adding anything new. Found this useful? Share it with someone who's about to build without a recipe. ♻️