LEANSpark

THE AI-NATIVE PLAYBOOKS · Nº 4

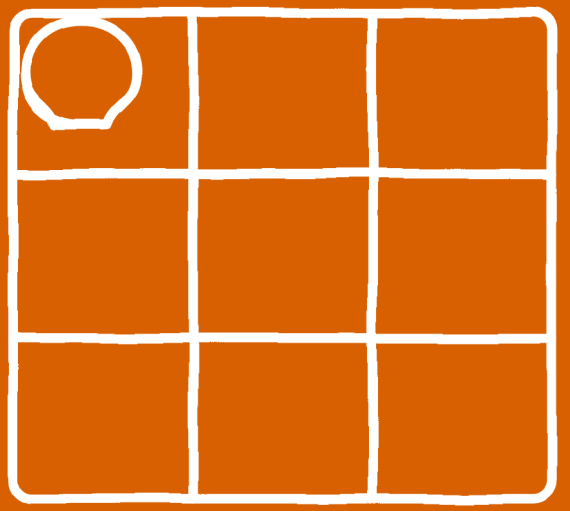
Is Your Idea Worth Pursuing?

Pressure-test any idea in an afternoon — before you write a line of code

LEANSTACK · Ash Maurya

leanspark.ai

Contents



CHAPTER 1

The million-or-billion-dollar question

The answer costs you the most expensive thing you own — time. There are three ways to find it, and only one is both fast and safe.

leanspark.ai

CHAPTER 1

The million-or-billion-dollar question

Every founder starts in the same place: an idea you've been circling — for weeks, maybe for years — and a quiet voice asking, *is this the one, or am I about to spend years of my life finding out the hard way?*

That question — "is my idea worth pursuing?" — is the million-or-billion-dollar question, because the answer costs you the most expensive thing you own. Not money. Time. The years you put into this are years you don't get back. So the real question underneath it is: *is this worth two or three years of my life?*

There are exactly three approaches founders use to answer it. I've used all three. Two of them cost me dearly.

Approach one: build first. You're convinced, so you go build it. Nine to twelve months heads-down, then another six trying to find customers for the thing you already made. By the time you learn whether anyone wanted it, you've burned eighteen months and most of your runway. It's the riskiest, most wasteful approach there is. *Life is too short to spend a year building something nobody wants.*

Approach two: customer research first. Talk to customers before you build. Weeks of interviews, then an MVP based on what you heard. Genuinely better than approach one — but it has two problems. It's still slow. And the dangerous one: *interviews can hand you a false positive.* People are nice. They'll say they'd "totally use that," and you'll walk away certain when all you collected was politeness.

> **I've been there — CloudFire.** Early on I built three products nobody wanted. CloudFire taught me the most, because I did everything approach two says to do: I interviewed, I built an MVP, and it *worked* — real, paying customers. Then I did the math I'd been avoiding. It cost me about twelve dollars to acquire a customer worth about four. Twelve in, four out. Desirable… and completely broke. That's the trap: you can have a desirable product, real customers, and a model that still doesn't close.

Approach three: business model first. Before you build, and before you even interview, you lay the whole business model on one page — a Lean Canvas — and stress-test it. You find the cracks while they're still cheap to fix, on paper, before you've spent a single real week on the wrong thing. Designing the model used to take one to two weeks. With LeanSpark it takes one to two hours — which makes approach three *faster* than customer-first and dramatically *safer* than build-first. That's the approach this runbook walks you through.

Get your idea onto one page

The job: turn your one sentence into a testable one-page model — without the blank-page paralysis.

IN LEANSPARK — YOU TYPE

Start an idea (or import a Lean Canvas PDF)

Reads your one sentence — or a Lean Canvas you upload as a PDF — and fills the nine boxes into a structured model you can actually test. You bring the idea; LeanSpark does the transcription.

RETURNS A populated Lean Canvas — problem, customer segments, UVP, revenue — the blueprint the rest of this runbook stress-tests.

VIA MCP — PROMPT ANY AGENT

"Read `business_model://current` . If it's empty, draft a first-pass Lean Canvas from this one sentence — "[your idea here]" — and name the customer segment specifically, never "users.""

CHAPTER 2

Stress-testing is a thought experiment

An architect finds the cracks in the blueprint before pouring concrete. So do you — with thought, because thought is free and experiments are expensive.

leanspark.ai

CHAPTER 2

Stress-testing is a thought experiment

Here's the analogy to keep for the rest of this book. An architect does not pour concrete to find out if the building stands. They run simulations on the blueprint first — load, wind, stress — and find the cracks while the building is still lines on paper, where a crack costs an eraser instead of a demolition crew.

A business model stress-test is the same move: a thought experiment you run quickly *inside the building*, on the blueprint, to find and fix the obvious cracks before you spend a single real week. You're not testing it in the market yet. You're testing whether it could even hold up in principle.

This isn't a startup hack — it's how rigorous thinking works everywhere. You test for *plausibility* first, with thought, because thought is free and real experiments are expensive. The physicist David Deutsch put it best:

> **David Deutsch.** "The overwhelming majority of theories are rejected because they contain bad explanations, not because they fail experimental tests." Most bad ideas aren't killed in the lab — they're killed because, when you reason through them, the explanation doesn't hold. That is exactly the move you make with your business model: before spending real weeks gathering evidence, reason through it hard and throw out the versions that contain bad explanations. Cheaply. With thought.

Every business model travels the same lifecycle: **idea → plausible → problem evidence → solution demand → product activation → product retention.** It starts as an idea — a pure leap of faith. The first move isn't to gather evidence; it's to get the idea to *plausible* — does it hold together as a model at all? That's the thought-experiment stage. Only then do you spend real time earning the empirical stages, each one costing more than the last.

Most founders skip straight from *idea* to building, or from *idea* to interviews — jumping the cheapest, most valuable stage. This runbook does that stage properly: idea → plausible, fast.

> **Which of the seven would you test first?** A full stress-test scores seven dimensions. Four are **core** — Clarity, Desirability, Viability, Feasibility — the must-pass gates; if one is broken, nothing downstream matters. Three are **strategic** — Mission, Defensibility, Timing — which sharpen a model that's already coherent. This runbook runs the four core, in order, because those four are what answer "worth pursuing." The order isn't obvious, and getting it wrong wastes weeks.

Run your first stress-test

The job: get your idea to "plausible" before you spend a single real week on it.

IN LEANSPARK — YOU TYPE

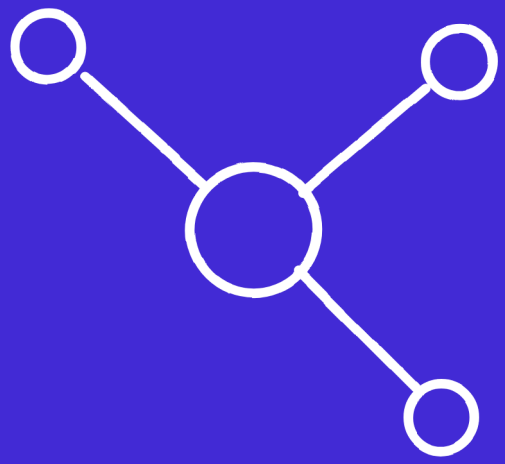
Run my stress-test

Simulates all seven dimensions as a thought experiment and scores each one — flagging which gates are must-pass and exactly where the cracks are — with no market evidence required yet. Your first run is free.

RETURNS A scored board across the seven dimensions, weak gates flagged, so you know which crack to fix first.

VIA MCP — PROMPT ANY AGENT

"Read `business_model://current` and stress-test it as a thought experiment. For each of the four core dimensions — Clarity, Desirability, Viability, Feasibility — say whether it is plausible and name the single biggest crack."



CHAPTER 3

Clarity — is my idea clear and concise?

You can't test three businesses at once — and neither can you build them. The fix for a blurry idea isn't better writing. It's splitting it into separate bets.

CHAPTER 3

Clarity — is my idea clear and concise?

The first core test is **Clarity.** It checks three things: is this *one* coherent model, not three-in-one; is the language *specific* — real segments, not a vague “users”; and are the Lean Canvas boxes filled with something actually testable.

Clarity is where most exciting ideas score lowest, and the reason is almost always the same. Take our example founder, Steve, whose one-sentence pitch is “let anyone build AR and VR worlds without writing a line of code — the Oasis, buildable by anyone.” A great sentence — vivid, ambitious, instantly picturable. Which is exactly what makes it dangerous: a sentence this exciting is easy to fall in love with and hard to test.

Look closer at Steve’s canvas and the problem appears. He doesn’t have one business model; he has *three wearing a trenchcoat* — a SaaS subscription business, a marketplace, and an advertising business, stacked on one page. Different customers, different economics, different everything. A stress-test literally can’t score a model that’s three models — and that’s the right answer, because neither can Steve.

> **The fix for clarity isn't to write better. It's to split.** When your canvas hides more than one business, no amount of wordsmithing rescues it. Spawn a separate variant for each coherent model. Now they’re not a debate in your head; they’re separate bets on the board, side by side, each a single testable business model. Spawning a variant *is* placing a bet — you’ve just turned one blurry idea into a few sharp ones.

Split one blurry idea into sharp bets

The job: turn three-models-in-a-trenchcoat into separate, testable bets.

▸ IN LEANSPARK — YOU TYPE

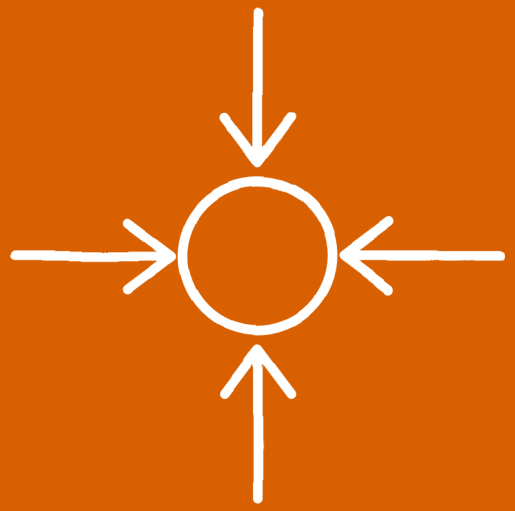
Spawn a variant

Splits a multi-model canvas into one coherent business model per variant — each with its own segment, problem, and economics — so every bet stands on its own and can be scored honestly.

RETURNS One variant per coherent model, side by side on the board. Each variant is a bet you can pursue or kill independently.

▸ VIA MCP — PROMPT ANY AGENT

"Read `business_model://current`. If it stacks more than one revenue model or segment together, propose the split into separate coherent business models — one segment, one problem, one revenue line each."



CHAPTER 4

Desirability — will anyone care?

Desirability isn't 'did someone say they'd buy it.' It's whether one specific customer would fire what they use today and hire yours instead.

CHAPTER 4

Desirability — will anyone care?

The second core test is **Desirability**, and it's where the false positive from Chapter 1 comes due. Desirability is *not* "did someone say they'd buy it." Remember CloudFire — people said yes, people paid, and it still didn't matter. Done right, desirability is about *switching*: is the pain acute enough that a specific customer would actually fire what they use today and hire your thing instead?

The key is what I call the **Innovator's Gift**: *new problems come from old solutions.* You never invent a problem in white space — you find it by looking at what people use *today* and asking what's broken with it. That chains four boxes on the canvas:

- **Customer Segments** — who.
- **Problem** — what's broken for them.
- **Existing Alternatives** — how they solve it *today* (not other startups — the real incumbent, even if it's a spreadsheet or a habit).
- **Unique Value Proposition** — what your solution must beat, by 10×, to be worth the switch.

> **Better is relative.** The trap is pitching your solution in a vacuum — "we're great!" The gift is that the existing alternative hands you the comparison for free. Don't ask "is the CD good," ask "is the CD better than the *cassette* — and at what?" Tangled tapes, no rewinding at a party. Those are known problems, and a UVP built on them is one people will actually switch for.

Now look at a canvas like Steve's through that lens. Four segments all "want" no-code AR/VR. That broad, easy appeal is exactly the smell of a false positive — *when everyone kind of wants it, usually no one wants it badly enough to switch.* Desirability forces the real question: which *one* segment feels this pain badly enough to switch *now*? Not four maybes. One yes.

Find the one segment that would switch

The job: replace "everyone kind of wants it" with one segment that would actually fire its current alternative.

IN LEANSPARK — YOU TYPE

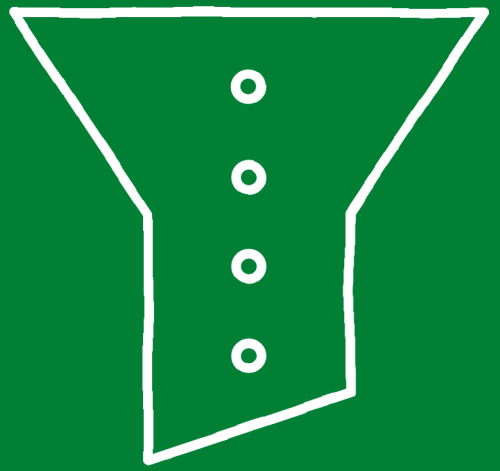
Test desirability

Chains Customer Segments → Problem → Existing Alternatives → the UVP that has to beat them, then pressure-tests whether the pain is acute enough for a real switch — flagging the broad, shallow appeal that signals a false positive.

RETURNS Per segment: the true existing alternative, whether your UVP is 10× better, and which single segment has switch-grade pain.

VIA MCP — PROMPT ANY AGENT

"Read `business_model://current` and `competitive://current`. For each segment, name the true existing alternative (what they use today, not other startups), the known problem, and whether the UVP is 10× better. Flag any segment whose desire is broad but shallow."



CHAPTER 5

Viability — is my idea worth pursuing?

The dimension that quietly kills real, customer-having businesses. Does the math close to your goal? Answer it in thirty seconds, on paper.

CHAPTER 5

Viability — is my idea worth pursuing?

This is the big one — the dimension that killed CloudFire, and the one I'd run first if I could run only one. **Viability** asks the question desirability can't: *does the math close to the goal?* You can be desirable and still broke — twelve in, four out.

You answer it with a back-of-the-envelope Fermi estimate — the **Rapid Viability Test**, in three steps.

Step one — goal sizing. Your Minimum Success Criteria (MSC): the annual recurring revenue that makes this worth your three years. Think in powers of ten. Level one, **$100K** — enough to quit your day job. Level two, **$1M** — a small company. Level three, **$10M** — a VC-backable business. Level four, **$100M** — a unicorn. Steve is after Level three: $10M.

Step two — customer sizing. Your average revenue per customer per year, rounded to the nearest power of ten. This picks your *animal*, and your animal picks your whole go-to-market:

- $10 **flies** — you need a million customers; you'd better be viral.
- $100 **mice** — a hundred thousand; product-led.
- $1,000 **rabbits** — ten thousand; inside sales.
- $10,000 **deer** — a thousand; field sales.
- $100,000 **elephants** — a hundred; enterprise.
- $1M **whales** — ten named accounts and multi-year sales cycles.

Steve, at $500/month (~$6K/year), is a **deer.**

Step three — market sizing. Customers needed = goal ÷ revenue-per-customer. Steve: $10M ÷ ~$10K ≈ **1,000 customers** by year three. Now the real question isn't "is that a lot" — it's *is the market big enough to supply a thousand of them, and can you reach them at that price?* If the number lands somewhere your market can't fill, that's a "no" you just got for free, in thirty seconds.

> **Desirable ≠ big enough.** Viability is the most common place a real, working, customer-having business quietly fails. A segment can want your product badly and still be too small — or too expensive to reach — to ever reach the goal. This is where "worth pursuing" grows teeth: it's where *desirable* meets *big enough*.

Size the bet against your goal

The job: find out in thirty seconds whether the math can even close to your goal.

IN LEANSPARK — YOU TYPE

Run the viability check

Runs the Rapid Viability Test — MSC goal ÷ revenue-per-customer = customers needed — rounds your price to a power of ten to find your go-to-market animal, and asks whether each segment's market can supply that many. A Fermi estimate, not a spreadsheet.

RETURNS Your animal, the customers-needed number, and a viable / too-small read per segment against your goal.

VIA MCP — PROMPT ANY AGENT

"Read `business_model://current` and `offer://current`. Set the MSC goal, compute customers-needed = goal ÷ annual-revenue-per-customer, round price to a power of ten to find the go-to-market animal, and judge whether each segment's market is big enough to supply that many."

CHAPTER 6

Feasibility — can I build it?

Feasibility isn't 'can I code it.' It's 'can I execute the whole model' — reasoned backwards from the goal. And the better you build, the more this dimension can fool you.

CHAPTER 6

Feasibility — can I build it?

The last core test is **Feasibility**, and it comes with a twist. Feasibility isn't really "can I write the code." It's *can I execute the whole business model* — and you reason about it **backwards: Timeline → Team → Solution.** Start from the goal, then the team it takes, then the smallest solution. Three checks: a **10X Traction Roadmap** (a staged path to your MSC), a **staged go-to-market** (ten customers, then a hundred, then a thousand), and **Demo-Sell-Build** (validate the promise before you build the thing).

Here's the picture of the roadmap: you don't leap to $10M — you climb to it in powers of ten. Month three: your *first paying customer* — the validation milestone, before you've built much at all. Year one: ten customers, $100K. Year two: a hundred customers, $1M. Year three: a thousand customers, $10M — your MSC. Each step is 10× the last. For Steve the deer, that's literally 10, then 100, then 1,000 deer — the roadmap and the animal line up. The discipline is to *play the hockey stick* deliberately — hand-pick your first ten, systematize the next hundred, scale to the thousand — instead of praying for a launch-day spike.

> **A green feasibility score is not a green light.** Can Steve build no-code AR/VR? Honestly — yeah. It's hard, but buildable, and with AI it's more buildable every month. So feasibility scores *fine* — and that is exactly the danger. For most of startup history "can I build it" was THE risk. That era is over. AI made building cheap, which made it the most *tempting* place to start — and the better you are at building, the easier it is to **build the wrong thing beautifully.** Feasibility passing is the test confirming your risk is somewhere else entirely.

Reality-check the build

The job: reason backwards from the goal to the smallest thing that proves the promise.

IN LEANSPARK — YOU TYPE

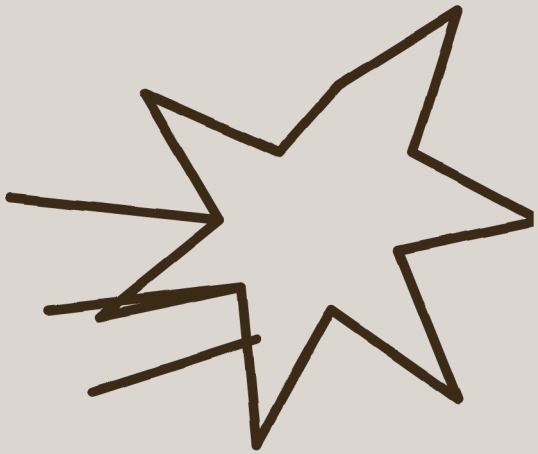
Plan feasibility

Builds a 10X Traction Roadmap (first customer → 10 → 100 → 1,000) staged backwards from your MSC, plus a Demo-Sell-Build plan so you validate the promise before building the product — scoring feasibility against the model, not just the code.

RETURNS A staged roadmap to your goal and the smallest solution that proves it — with the real risk surfaced when feasibility comes back green.

VIA MCP — PROMPT ANY AGENT

"Read `business_model://current` . Build a 10X traction roadmap from first paying customer → 10 → 100 → 1,000, staged to the MSC, and describe the smallest Demo-Sell-Build test that validates the promise before building."



CHAPTER 7

Place your bet

Worth pursuing isn't a feeling — it's a bet you can see. One idea, its directions, a score on each, and a chip pushed forward on one.

CHAPTER 7

Place your bet

Put it together. Your idea is on a canvas. You've split it into coherent variants, scored each on the four core dimensions — clear, wanted, big enough, buildable — and now comes the decision most founders avoid for months: you **commit.** You pick the one bet to chase first and mark it primary, based on which variant has the sharpest early-adopter pull and the clearest path to the goal.

That's what "worth pursuing" looks like as a decision you can see: an idea, its possible directions, a score on each, and a chip pushed forward on *one.* Not "all four someday." This one, first.

> **A "no" today beats a "yes" you earn in eighteen months.**
> Sometimes the honest read is *reshape it* — or *drop it*. If none of your bets could plausibly reach the goal, that's not failure. That's the most expensive years of your life handed back to you in an afternoon, instead of at the end of a build.

You don't make an idea worth pursuing by believing in it harder. You make it worth pursuing by designing the model, stress-testing it, and betting on the riskiest assumption first — before you build. That's the whole game.

Commit your first bet

The job: pick the one bet to chase first — or hand yourself the years back.

› IN LEANSPARK — YOU TYPE

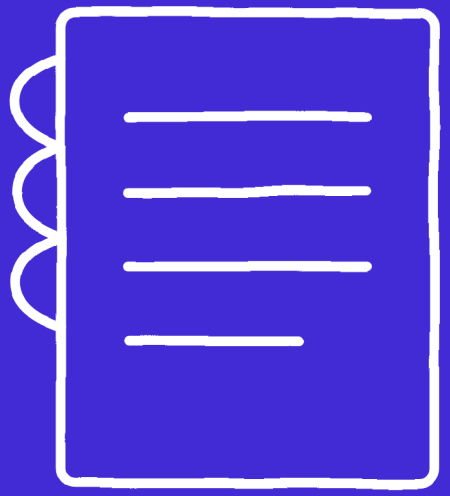
Commit a variant as primary

Ranks your variants on the four core dimensions and lets you commit one as primary — the bet with the sharpest early-adopter pull and clearest path to the goal, with the riskiest assumption named for you to test next.

RETURNS One committed primary bet on the board — the decision "worth pursuing" actually looks like. Or a cheap, honest "no."

› VIA MCP — PROMPT ANY AGENT

"Read `business_model://current` and its variants. Rank them on the four core dimensions and recommend which single bet to commit as primary — and name the riskiest assumption to test first."



CHAPTER 8

Where this goes next

Worth pursuing is the first loop, not the last. Run it on LEANSpark, then go find out if real customers want it.

CHAPTER 8

Where this goes next

Getting an idea from a sentence to *plausible* is move one — the idea → plausible stretch of the lifecycle. The bet you just committed is a *desirability* question: which customer actually wants this. That's exactly what comes next — pressure-testing the bet against real customers with interviews and the forces that make people switch. Worth pursuing is the first loop, not the last.

Run it on LEANSpark

- **Import or start an idea** — a sentence or a Lean Canvas PDF becomes a structured model in minutes.
- **The stress-test** — score the seven dimensions as a thought experiment; your first run is free.
- **Spawn & commit bets** — split blurry ideas into variants, score them, and commit the one to pursue first.
- **The Rapid Viability Test** — size any segment against your MSC in thirty seconds.

Go deeper on the method

- *Running Lean* and **The AI-Native Founder's Playbook** (No. 1 in this series) — the whole journey from idea to traction.
- **How to Talk to Customers** (No. 3) — the next loop: turning a bet into evidence real customers give you.
- **Is There Demand for My Idea?** — the follow-on workshop, where a bet like this one meets real customers.

The stress-test at a glance

TEST	THE QUESTION	THE "NO" IT CATCHES
Clarity	Is it one clear model?	Three businesses in a trenchcoat.
Desirability	Would anyone switch?	Broad appeal, nobody switches (false positive).
Viability	Does the math close to the goal?	Desirable and broke.
Feasibility	Can I execute the model?	A green build hiding the real risk.

LEANSpark

Is Your Idea Worth Pursuing? is part of The AI-Native Playbooks, a series from LEANSTACK / Ash Maurya. LEANSpark stress-tests your idea into a decision — before you build.

leanspark.ai