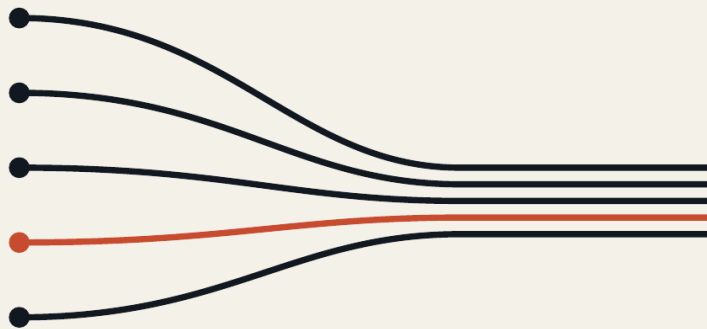


MAGNETIC

From Idea to Working Product

A Five-Step Method to Research, Build, Delegate, Simplify,
and Improve with AI



- 1 RESEARCH
- 2 BUILD
- 3 DELEGATE
- 4 SIMPLIFY
- 5 IMPROVE

AMINE LARHRIB

ZILLION LABS

Magnetic — Free Reading Sample

Amine Larhrib

BookGist Summary

Magnetic — Free Reading Sample

by Amine Larhrib

Genre: Business & Entrepreneurship

Key Takeaways

- Start with a specific person, task, and moment of difficulty—not a broad product vision or a technology you are eager to use.
- Research the last real occurrence of a task: ask what happened, inspect a redacted artifact when possible, and understand the current workaround before proposing a solution.
- Distinguish observations from interpretations and assumptions. One customer's account can suggest an experiment, but it does not prove a market or business.
- Treat existing alternatives—such as email templates, checklists, or manual help—as serious competitors, and identify what they already do well.
- Define a narrow first product around a complete useful path, including its inputs, output, human decision, and ability to retrieve the result later.
- Make product commitments earn one another: research should justify a build, a working product should justify delegation, and observed experience should justify automation or further investment.
- Keep authority and boundaries explicit. BriefDesk, for example, can draft a brief and surface missing information without pricing work, accepting contract terms, or promising delivery.
- A successful demonstration is not yet a dependable product; users also need saved records, recoverability, clear handling of failures, and understandable data practices.

- Stopping, simplifying, or deciding not to automate is a valid outcome. Progress means making a better decision, not adding features or agents.

Who Should Read This Book

Magnetic is aimed at independent founders, consultants, builders, and small teams who want to turn an AI-enabled idea into a focused product without assuming that a prototype is already a business. It is especially useful for readers who need a practical way to interview potential users, define a small first version, test a complete workflow, and make uncertainty visible. Readers should be willing to inspect evidence and seek specialist help when a project involves sensitive data, regulated decisions, or consequential infrastructure. The sample's approach is more operational and decision-focused than a book devoted only to rapid prototyping or prompt techniques: it connects user research to building, delegation, simplification, and ongoing improvement. Its strongest immediate value is helping readers avoid premature scope and false confidence. The supplied text includes the opening, the five-step overview, and the first research chapter; the later steps are introduced but not developed in the excerpt.

Chapter Summaries

The Five Steps at a Glance

Magnetic organizes product development into five connected decisions: Research, Build, Delegate, Simplify, and Improve. Research produces a focused brief grounded in relevant evidence; Build creates a useful workflow that can be tested and recovered; Delegate defines bounded work for an agent—or records the choice to keep the work manual; Simplify turns predictable decisions into rules while preserving an exception path; and Improve uses feedback to decide what should change next. The steps are a learning sequence rather than a rigid checklist. Feedback can send a team back to an earlier decision, and no one is required to add an agent just to complete the process. The book's recurring example, BriefDesk, turns client requests into draft briefs, makes missing information visible, and preserves the version a person approves. The sample presents these connected outputs as a way to make commitments and uncertainty easier to inspect.

Introduction: The Demo Is Only the Beginning

The introduction contrasts a polished demonstration with the practical problems that appear when someone tries to use a product for real: an output may invent details, repeated actions may create duplicate records, saved work may be hard to find, and users may ask privacy

questions the builder cannot answer. AI can accelerate ideation, specification, interface design, and coding, but that speed can also make unresolved assumptions look settled. Magnetic's answer is to connect the decisions that turn an idea into something useful and operable. A studio request illustrates how to narrow the work. Rather than building an expansive AI workspace, the proposed first task is to convert a client request into a draft brief, preserve the requested deliverables, distinguish different dates, mark an unknown approver, and avoid inventing a budget or delivery promise. The complete path includes submission, a saved draft, visible gaps, human correction and approval, and later retrieval. The introduction emphasizes that delegation and automation should follow understanding, not precede it, and that abandoning an idea or keeping part of the work manual can be a sound result. The example product follows Lina, who runs a small design studio. Her broad idea becomes BriefDesk: a focused tool for turning project requests into structured draft briefs and getting human approval before anything becomes a commitment. Readers are encouraged to choose one project, keep a shared record of evidence and decisions, and begin with the task users cannot reliably complete today.

1. Choose a Problem Small Enough to See

This chapter argues that broad ambitions such as an “intelligent workspace” are difficult to evaluate because they can encompass sales, scheduling, project management, billing, and many other jobs. Lina makes progress by identifying a concrete moment in her studio's work: a new client request arrives, staff assemble a draft brief, ask questions, revise it, and agree on scope. The research task is to understand who performs that work, what triggers it, where it becomes difficult, and what consequences follow. The chapter recommends asking people to walk through their most recent real experience rather than asking whether they like a proposed product. With permission, inspect a redacted input or output, learn what the current process already does well, and record costs in recognizable terms such as delays, rework, checking time, or missed decisions. It also advises keeping claims, sources, observed behavior, and resulting decisions distinct. Statements of interest, trial commitments, payment, successful use, and repeat use are different kinds of evidence and should not be treated as interchangeable proof of demand. Lina's conversation with Noor changes the idea. Noor initially agrees that an assistant sounds useful, but a walkthrough of a recent project reveals a more specific problem: deliverables were added across messages, and two people each assumed the other had approved the copy. The key issue is not simply writing a polished brief; it is exposing missing or conflicting commitments before the studio agrees to deliver. The chapter's practical exercise is to name a decision an interview could change, ask about a recent event and its workaround, and record what was learned, what remains uncertain, and what action should change as a result.

Notable Quotes

"A working product helps a specific person complete a useful task, preserves the result, and can be recovered when something fails."

"The problem is usually a missing sequence of decisions."

"The central discipline of this book is to make each commitment earn the next one."

"Your first competitor may be an email template."

"A person who watches a demonstration has observed a possibility. A person who completes a real task has tested usefulness."

"A first boundary should also say what is outside it."

"Begin with the task your user cannot reliably complete today. That is a better starting point than the tool you are most eager to try."

Full Summary

A client request arrives at a small studio, and someone must turn it into a draft brief, identify what is missing, and preserve the version a person approves. BriefDesk, the book's teaching product, is designed around that specific path. Its purpose is not just to produce text: it must help someone complete a useful task, keep the result, and make that result recoverable if something goes wrong. The five-step Magnetic method connects the decisions needed to make such a product work.

The sequence begins with Research, which asks whether the problem is worth a small test. The intended output is evidence and a focused build brief. Research is not treated as a preliminary formality that can be left behind once coding begins. It gives the build a reason and a boundary, and makes clear what is known and what still needs testing. The question is deliberately modest: not whether an entire market needs a sweeping new platform, but whether a particular task and proposed first product merit a small experiment.

Build asks whether a user can complete the whole path. Its output is a tested, recoverable workflow: something more substantial than a polished screen or an isolated demonstration. A user needs to be able to move through the useful task, and the product needs to preserve the result so that it can be retrieved. The workflow must also be possible to test, pause, and recover. This makes the build a working route through the task rather than a collection of features whose usefulness has yet to be established.

Delegate follows only when there is work that might reasonably be assigned. The question is which repeated work can be bounded. Its possible output is a role contract that defines the delegated job, or an explicit decision to keep the work manual. Delegation is therefore an option to evaluate, not a box that every project must tick. A product can move through the

method without adding an agent. If the work cannot yet be described and judged, keeping it manual remains a legitimate choice.

Simplify asks which decisions have become predictable. Its output is a design with rules for routine cases and a route for exceptions. The step looks for a clearer, more manageable way to operate the work once experience has made some parts understandable. It does not assume that every task should remain an agent's responsibility: routine cases can be governed by rules, while exceptions retain a path for judgment. Simplification is a distinct decision, based on what the work has shown, rather than a reason to automate merely because automation is available.

Improve asks what the next evidence should change. It leaves the project with one decision, an owner, and a review point. Feedback is not simply gathered and admired; it should lead to a considered choice about what changes next. The owner and review point make that choice inspectable, instead of letting feedback accumulate without affecting the product. Improvement can also return the work to an earlier decision. If an observation weakens an assumption, the team may need to revisit the brief, workflow, delegated role, or operating rules rather than pressing forward along the sequence.

The five outputs are connected: a focused brief, a tested workflow, a bounded delegated role or an explicit choice to remain manual, a simpler operating design, and a feedback decision. Each makes the next commitment easier to inspect. Research provides the basis for building; a working product makes it possible to judge repeated work; observed work can then inform delegation and simplification; and feedback can guide the next change. The method is meant to keep those commitments linked, rather than allowing research, building, and automation to become disconnected experiments.

The order is a main path, not a rule that a project must follow mechanically. Feedback can send the team back to any earlier step, and no stage requires manufacturing work just to claim completion. Someone whose task or intended user remains unclear should begin with Research. Someone who already has relevant evidence but only a mockup can begin with Build. If a working product creates repeated work that its operator can already judge, Delegate may be the useful entry point. If automation is difficult to supervise, Simplify is the place to examine the design. And if feedback is being collected without any clear account of what it changes, Improve offers the appropriate starting question.

Even when starting in the middle, it helps to understand how the outputs connect. The diagrams support that reading: solid arrows indicate the next action, while dashed arrows mark a return for more information or revision. Numbered steps show each item's place in the method, and labels carry the meaning, so the diagrams remain usable in black and white. The structure is meant to be worked with, not merely read. A person can predict the next decision, compare that prediction with a worked response, and complete the corresponding step for a

project of their own. The examples follow BriefDesk, whose path turns a client request into a draft brief, exposes missing information, and preserves the version a person approves. Its structure can be adapted to another task, while the decisions remain tied to that task's actual work.

A polished demonstration can make the distance between an idea and a product seem to vanish. The screen has a title, a clean form, and a button that appears to do exactly what you asked. You enter the example you have used all afternoon, and a result appears. Then you try different information. The result sounds plausible but includes a detail no one supplied. You press the button again and create two records. You close the page, return, and cannot find either one. A potential user asks whether the information is private, and you do not know where it is stored.

These are ordinary questions, but they arrive after the exciting part. A convincing demo has to become something useful, and the transition exposes decisions that the demo can hide. AI can help explore an idea, write a specification, build an interface, generate code, and draft the work around a product. It can also make uncertainty look settled too soon. A clear answer may conceal a weak assumption; a polished interface, a missing operation; an agent that appears busy, an expensive loop.

The difficulty is not usually a shortage of tools. It is a missing sequence of decisions. Each commitment should rest on something learned in the work before it: evidence should support a focused build; a working path should make it possible to judge what could be delegated; observed cases should reveal which tasks are stable enough to simplify; and feedback should give a reason to change or invest further. The aim is to make uncertainty visible before it becomes expensive, not to pretend it can be eliminated.

That discipline also gives you permission to stop. You may find that an existing spreadsheet is more useful than the product you imagined, or that the product has value while its proposed automation does not. The first version might work best as a service with a modest software component. Those findings are useful even if they redirect or end an attractive idea. The point is to make a better decision, not to manufacture progress toward a predetermined launch.

Consider a studio request: "We need a landing page and three launch emails. The event is on 18 November. Please send a quote next week. We have not decided who approves the copy." An unfocused instruction would be to build an AI platform for managing creative projects. A focused one would ask for a draft brief that preserves the requested deliverables, keeps the event date separate from the requested quote date, marks the approver as unknown, and does not invent a budget or promise delivery.

The difference is already testable on paper. A useful output should contain two known deliverables, two distinct time references, and an unresolved approval decision. If the system

invents an approver or treats the quote date as a delivery deadline, the failure is clear and specific. No platform needs to be chosen before that test.

The product can then be described around a complete path. Someone submits the request; a draft is saved; missing information is shown; a person corrects it; approval preserves the agreed version; and someone can retrieve that version the next day. The output is not just convincing text on a screen. The process includes the inputs, the human decision, and a record that survives the immediate interaction. Only after a person can judge a draft does it make sense to consider delegating that work. Routine tasks may later become rules, while errors and user decisions help determine what should change.

Lina, a fictional founder who runs a small design studio, provides the recurring example. Prospective clients send project requests through emails, notes, and forms. Someone must turn those fragments into a brief that the studio and the client can approve. Missing deadlines and vague deliverables create repeated clarification work. Lina's initial idea is an AI workspace for creative businesses, but the useful product becomes much smaller: BriefDesk turns a project request into a structured draft, shows what is missing, and obtains human approval before the brief becomes a commitment.

BriefDesk is deliberately ordinary: it has inputs, an output, a human decision, and a reason to be used again. That makes it possible to examine what a product actually does without pretending to solve an entire industry at once. Its examples are teaching scenarios, not customer outcomes or forecasts. The numbers are invented for instruction, and the mistakes are included so they can be inspected.

The method is intended for independent founders, consultants, builders, and small teams using AI to turn a practical idea into something people can use. No one needs to be an experienced programmer to understand the decisions, but builders must be willing to inspect results, learn basic concepts, and seek appropriate help when the consequences exceed their ability to evaluate the work. A fixed number of days cannot guarantee a viable business, adding AI cannot create demand, and a solo builder cannot safely launch every kind of product.

The work can begin wherever the uncertainty lies. A product with reliable software but chaotic operations may need clearer roles and a record of costs. A busy automation system with little customer demand may need to return to learning what people need. In either case, the decisions and evidence should remain together, so that each exercise feeds the next choice rather than becoming another unused document.

Choose one project and keep a single working record of its evidence, decisions, and open questions. The resulting brief, checks, deployment record, delegated role or decision to remain manual, cost ledger, and feedback routine should serve the product's next decisions.

The starting point is not the tool most tempting to try, but the task a specific person cannot reliably complete today.

For Lina, the moment of difficulty comes when a new client request arrives at her small design studio. Details may be scattered across emails, notes, and forms; someone must turn them into a draft brief, ask about what is missing, revise it, and establish what the work includes. Her first idea, an intelligent workspace for creative businesses, gives her no clear boundary. It could encompass sales, scheduling, project management, design review, billing, or file storage. Choosing the moment when a request becomes a brief turns an appealing but sprawling idea into a task she can investigate.

The investigation begins with someone who recently performed that task. Rather than asking whether the person likes a proposed product, Lina asks them to reconstruct the last time it happened: what triggered the work, what information arrived, what they did first, where they waited, what they copied or checked, and what happened afterward. The sequence matters because annoyance alone does not establish a reason to change. A calm account may reveal costly errors, while an emphatic complaint may have little practical consequence.

When permission allows, Lina asks to examine a redacted example of the input and output. A general claim that briefs are messy is less useful than seeing a delivery date in an email conflict with an outdated date on a form. Artifacts help distinguish what occurred from what someone remembers or infers. Lina also keeps the roles and settings of interviewees distinct: a freelance designer, procurement manager, and studio owner may use the word “brief” for different kinds of work. Blending their accounts into one imagined customer risks producing a tool that serves none of them well.

The alternatives already in use deserve serious attention. For Lina's studios, these might include copying an old brief, asking a project manager to prepare one, using a form, or sending clarification emails. An email template may be familiar, cost nothing extra, and embody years of judgment. Lina needs to learn what each workaround does well before deciding what to replace. If old briefs already provide useful language and the real difficulty is noticing missing information, then the first product may be a missing-information review, not a longer generated document. The cost of the current process should be described in terms users recognize—checking time, delay, repeated questions, rework, or missed decisions. Estimates that have not been measured should be labeled as estimates.

Lina must also keep enthusiasm separate from evidence. Someone saying they would use a product is a weak signal. Showing a recurring problem is stronger; agreeing to try a prototype on the next suitable task creates a concrete learning opportunity. A paid pilot can be stronger still, but only in light of what the buyer understood and what was promised. A trial commitment indicates willingness to test, not lasting demand. Payment records a purchase under particular conditions, not successful use or a profitable business. Even a successful task does

not by itself prove repeat use. For each important claim, Lina records its source, the behavior observed, and the decision it might affect, marking interpretations and gaps rather than disguising them as facts.

Access to suitable first users is part of the product's design problem. Lina needs a route to people who actually perform the work, not merely a market description or an adviser who recommends software. She can map users, managers, and potential introducers separately, and note why each person might engage. A former client, a relevant professional community, or a partner facing the same handoff problem may provide a legitimate route, but access to a community does not grant permission for repeated promotion.

A useful research request is specific and modest. Lina might ask someone involved in preparing studio briefs to spend fifteen minutes walking through a recent example, explaining that a redacted description is enough and that the conversation is about understanding the current process. The request names the task, time, privacy boundary, and purpose without leading with a demonstration. If nobody responds, Lina should examine whether the request reached the right role, described a recognizable task, or asked too much. Silence across credible routes says something about her ability to reach the market, even if it does not prove that the problem is absent.

A pilot invitation should be separate from research outreach. It must explain what works, what remains manual, what information is needed, what the user receives, who handles failures, and when the trial ends. Lina should ask for a concrete next step, such as a scheduled session or an agreed example. A mailing-list signup is not evidence that the product delivers value. If the offer is paid, its terms should make clear whether the customer is buying a human-reviewed service or unattended software. After the trial, Lina should ask what the user did with the output: watching a demonstration, completing a real task, and returning when the task recurs are different levels of evidence.

The first boundary for BriefDesk is a sentence about a particular user and task: small design studios need to turn incoming project requests into draft briefs and identify missing information before agreeing to delivery. That statement remains a set of assumptions to test, but it specifies a user, an output, and a consequential moment. The limits matter too. BriefDesk can draft a brief and reveal what is missing; it will not price the project, accept contractual terms, or promise a delivery date. A human retains those decisions, keeping the product's authority understandable.

Lina's conversation with Noor, a studio project lead in this teaching scenario, shows why these distinctions matter. Lina initially asks whether Noor would use an assistant that turns client messages into project briefs. Noor says yes, but the answer describes a hypothetical future, not the work Noor needs done. Lina then asks for the last request that took longer than expected. Noor recalls a launch page requested in one email, three launch emails added

later, and two people who each assumed the other had approved the copy. The first brief was well formatted; the delay arose from resolving contradictions after work had begun.

Noor cannot share the original client conversation, so the two reconstruct its sequence on paper. Lina marks which details Noor recalls and which could be checked later. The finding changes the idea: the task is not simply to write a good brief, but to make missing or conflicting commitments visible before the studio agrees to deliver. A generic writing assistant could produce smoother prose while making that problem harder to spot.

Lina records three different statements. Noor described a project in which inconsistent requests delayed agreement; that is the observation, though it is recalled rather than independently verified. Surfacing uncertainty earlier might prevent such delays; that is an interpretation. Other reachable studio leads might face the problem often enough to adopt a separate tool; that is an assumption. One conversation can guide an experiment, not establish a market. Lina's next task is to speak with other relevant people, learn whether a checklist or another existing workaround already handles the issue, and ask about their own recent examples. For each conversation, she should decide in advance what question the answer could change, then leave the feature list closed until she understands what actually happened.