MVPCOMPANION

THE ESSENTIAL GUIDE

From an uncertain app idea to a focused, testable and buildable MVP

A practical guide to progressive clarification, product decisions, and learning

Preface

Before You Vibe-Code

Building an app is easier to start than it used to be. Deciding what should be built is still difficult. A prototype can make an idea feel concrete before the problem, intended user, or expected value is clear. Speed helps only when it moves the right decision forward.

When those questions remain vague, speed can create a false sense of progress. A team may build faster while learning very little about whether the product matters. The result can be a polished solution to a problem that was never clearly established.

MVPCompanion offers a more deliberate starting point. It treats product definition as a sequence of connected decisions made under uncertainty. The aim is to turn an initial idea into a focused, testable, buildable MVP specification—without pretending that a specification removes uncertainty or guarantees success.

This is not an argument against building. It is an argument for making building purposeful. Sometimes the right next step is a prototype. Sometimes it is a conversation with a potential user, a comparison with existing alternatives, or a small test of a critical assumption. The method helps make that choice explicit.

AI can support this work by organizing rough thoughts, surfacing missing assumptions, comparing alternatives, drafting candidate requirements, and checking consistency. It can also make unsupported claims sound convincing. Treat its output as material to inspect, not as evidence. The responsibility for decisions and their justification remains with you.

This short guide is designed to be used, not simply read. Bring an idea, question it, and record what you know, what you assume, and what remains open. The goal is not to complete every worksheet. It is to make the next important product decision better informed. It is a concise introduction, not a substitute for the complete canonical MVPCompanion book or a claim that every canonical chapter requirement and artifact is covered in full. Use it to understand the method's logic and begin applying it; consult the full method specification and develop the required artifacts before treating an MVP specification as complete.

A USEFUL RULE Use AI to improve the quality of your thinking — not to hide uncertainty or outsource product judgment.

Chapter 0 — Introduction

From an Idea to Something Worth Building

CORE QUESTION: What is MVPCompanion, and how will it help me turn an idea into a coherent, testable MVP?

LEARNING OBJECTIVE: Understand the roles of the Method, 15-stage Process, Artifacts, cross-cutting lenses, and supporting workstreams.

MVPCompanion is a method for progressively clarifying a product idea. It connects the problem, the people affected, the outcome they seek, the Job they are trying to accomplish, and the product decisions intended to support that outcome. The work produces a coherent MVP specification rather than a feature list assembled too early.

The method has three connected layers. The Method defines the reasoning principles and decision logic. The Process organizes that reasoning into 15 stages. The Artifacts capture the outputs needed to explain, test, and connect decisions. Together, they help make product reasoning visible and reviewable.

Figure 0.1 shows how the Method, 15-stage Process and Artifacts work together; the Integrated MVP Specification aggregates their outputs and is not a sixteenth stage.

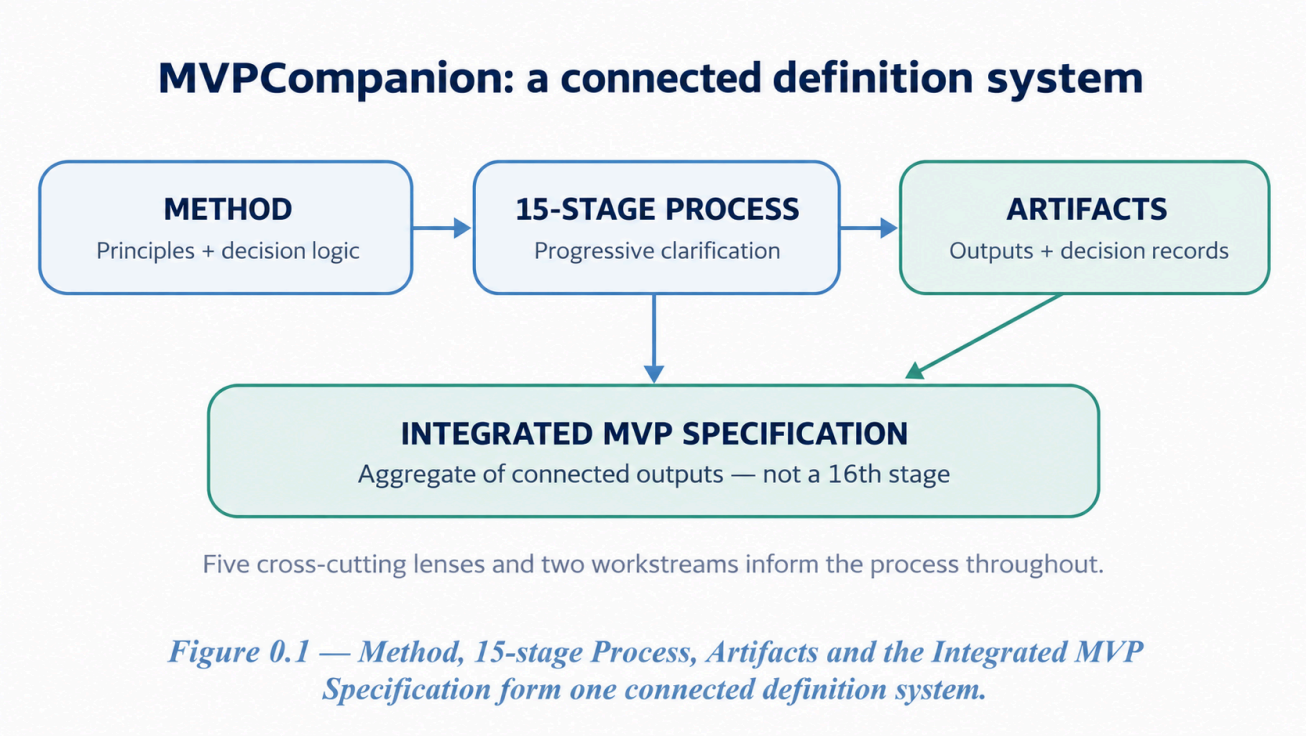


Figure 0.1 — Method, 15-stage Process, Artifacts and the Integrated MVP Specification form one connected definition system.

The 15-stage process

Stage	Question
01 — Idea	What could this be?
02 — Problem	What real problem exists?
03 — User	Who experiences the problem?
04 — Value	What outcome matters, and why this product?
05 — Product	What is this product fundamentally?
06 — Context & Lifecycle	How does the central product object change over time?
07 — Core Jobs	What must users accomplish?
08 — Scope	Which capabilities are candidates for the MVP?
09 — Experience	How does the user reach value?
10 — Information Architecture	Where does information and functionality belong?
11 — Data Model	What information must exist and how is it related?
12 — Requirements	What exactly must the product do?
13 — Learning	What evidence will help us know whether it works?
14 — Technical Boundaries	What technical decisions are necessary to build it?
15 — MVP Boundary	What is included in this MVP, and what is deliberately excluded?

The order matters because decisions depend on one another. It is difficult to define a useful feature before understanding the problem and user; it is difficult to write meaningful acceptance criteria before deciding what the product must do; and it is difficult to choose metrics before clarifying what outcome the MVP is meant to support. Later work may expose a weak earlier assumption, so revisit earlier decisions when new evidence warrants it.

Five lenses, not five more stages

Use five cross-cutting lenses throughout the process. Desirability asks whether the proposed product addresses a meaningful user need. Differentiation asks why someone might choose it over existing alternatives. Viability asks whether a sustainable value exchange is plausible. Feasibility asks whether the product can be delivered within relevant technical and operational constraints. Learning asks what the MVP should help you find out.

The lenses are recurring questions, not a second sequence of stages. A proposal may look attractive through one lens and weak through another. Use the tension to identify what needs to be clarified, tested, narrowed, or deliberately left unresolved.

Two cross-cutting workstreams

Market Discovery and Business Model thinking run alongside the stages. Market Discovery examines users, alternatives, context, and evidence that may change product decisions. Business Model thinking examines who receives value, who uses the product, who may pay, and what would need to be true for the exchange to be viable. These are cross-cutting workstreams, not additional stages.

How to read and use this book

Each Part explains a connected set of decisions, returns to the Master Case, and ends with an output or readiness check. Work with your own idea as you read. Keep assumptions visible, and treat every output as a working artifact that may change when stronger evidence becomes available.

Evidence notation depends on the source context. In the 15-Stage Specification, E means Evidence, H means Hypothesis, D means Derived Insight, and O means Open Question. In the Master Case Evidence Matrix, retain its labels: E = Evidence, H = Hypothesis, D = Derived Reasoning, C = Decision, R = Result/Status, and O = Open Evidence Gap. Do not merge these conventions silently. A decision is not a result, and a planned test is not evidence that the test occurred.

ORIENTATION EXERCISE Write one sentence describing your idea, the outcome you hope it creates, what you currently know, what you are assuming, and the most important question you cannot yet answer. This is a Starting Point Note, not the Stage 1 Idea Brief.

MVPCompanion does not promise that every idea can become a successful business, or that a specification eliminates uncertainty. It offers a disciplined way to expose assumptions, connect decisions, and identify what should be tested next. The quality of the result depends on the quality of the inputs, the honesty of the reasoning, and the evidence gathered.

The journey begins with a simple discipline: do not confuse an idea with a product, a plan with evidence, or a connected set of documents with a validated decision. Make the reasoning visible, then use it to decide what to do next.

PART I — THINK DIFFERENTLY

Treat the MVP as a coherent value-and-learning system.

CORE QUESTION: What makes an MVP worth building and testing?

LEARNING OBJECTIVE: Understand the difference between an idea, a product hypothesis, and a coherent MVP; use progressive clarification and the five lenses.

1. An idea is not yet a product

An idea often arrives as a solution-shaped sentence: an app that tracks something, recommends something, automates a task, or helps people coordinate. That sentence is a useful starting point, but it leaves key questions unanswered. Who has the problem? In what situation? What do they do today? What outcome would matter? Why might they choose this solution over the alternatives?

Treat the idea as a hypothesis, not a conclusion. A useful hypothesis is explicit enough to challenge. It names a possible user, a possible need, and a possible benefit without implying that any of them has been validated. The task at this stage is not to make the idea sound impressive; it is to make it possible to investigate.

Separate known information, assumptions, and open questions. Known information is supported by something you can identify. An assumption may be reasonable, but it has not yet been established. An open question names what remains unresolved. This distinction matters because a polished document can make an assumption look like a fact simply by presenting it confidently.

Use Figure 1.1 when framing an idea: label what is supported, what is assumed, and what remains open.

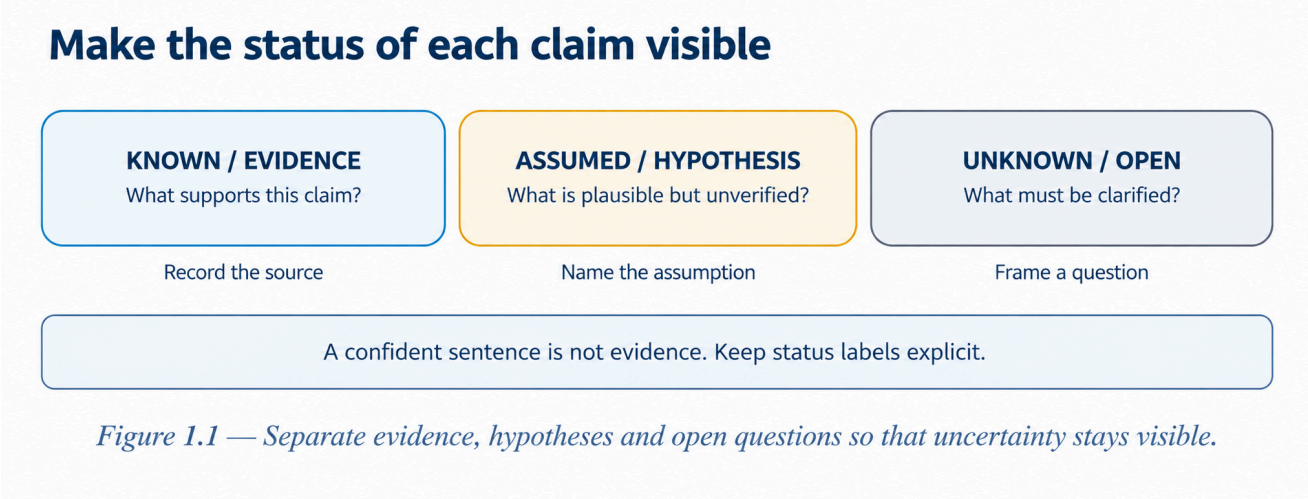


Figure 1.1 — Separate evidence, hypotheses and open questions so that uncertainty stays visible.

2. What an MVP really is

A minimum viable product is not merely a smaller product or a short feature list. It is the smallest coherent solution intended to deliver a meaningful part of the value and support a relevant learning objective. A feature can be small and still be unnecessary; a feature can be easy to build and still contribute little to the core promise.

Apply two tests. The value test asks whether the product can help the intended user accomplish something that matters. The learning test asks whether using or testing it can produce evidence that informs an important decision. A credible MVP connects both. Minimality alone is not enough if the

product cannot deliver meaningful value; data collection alone is not enough if the experience is too incoherent for the findings to be useful. Figure 2.1 distinguishes a coherent value-and-learning MVP from proposals that deliver only one or neither.

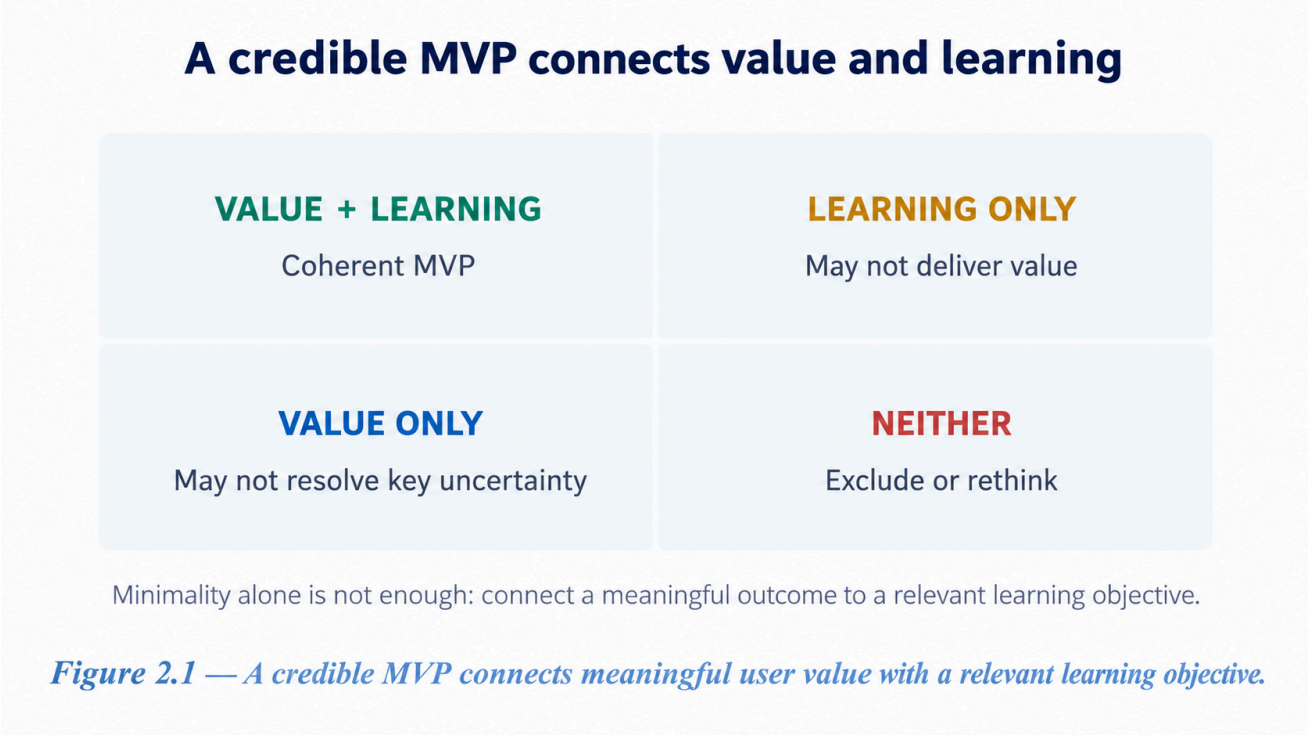


Figure 2.1 — A credible MVP connects meaningful user value with a relevant learning objective.

3. Progressive clarification

MVPCompanion breaks the broad task of product definition into a sequence of focused questions. Each stage has a purpose, a decision to clarify, an output, and a readiness check. The check asks whether the question has been answered clearly enough to proceed — not whether uncertainty has disappeared. Figure 3.1 provides the high-level route from an initial idea to an explicit MVP boundary; the sequence structures reasoning but is not a rigid waterfall.

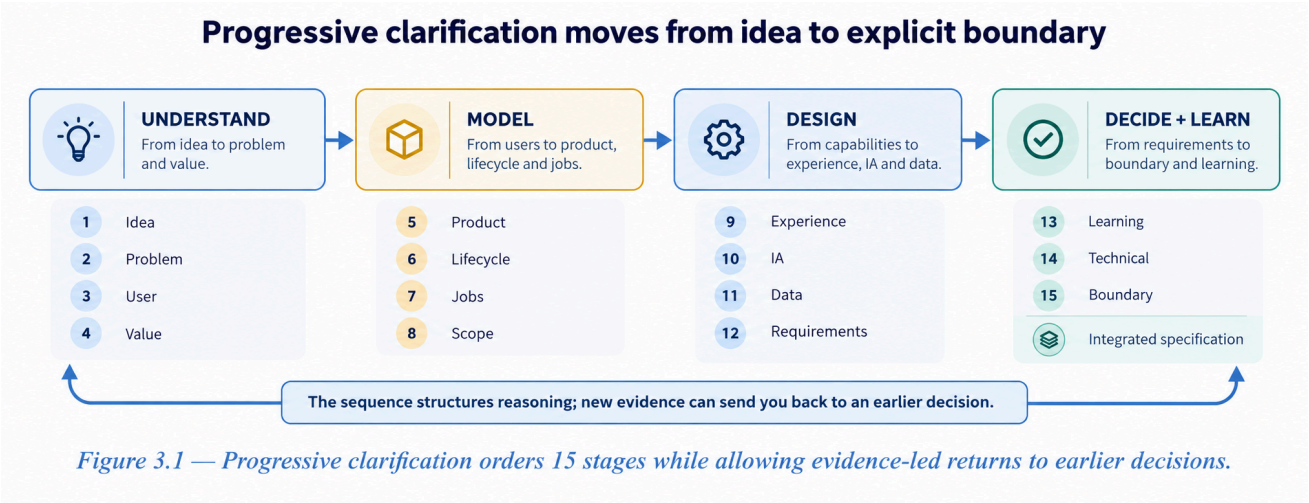


Figure 3.1 — Progressive clarification orders 15 stages while allowing evidence-led returns to earlier decisions.

Progressive clarification is not a rigid waterfall. New evidence may change an earlier hypothesis. The discipline is to make the change explicit, trace its consequences, and avoid carrying contradictory

assumptions into later work. The process orders the reasoning; it does not guarantee that reality will follow the plan.

4. Use the five lenses

Lens	Question to ask
Desirability	Does this address a meaningful problem or outcome for the intended user?
Differentiation	Why might this be preferred over the status quo or alternatives?
Viability	Is there a plausible way to create and capture value sustainably?
Feasibility	Can the proposed behavior be delivered within relevant constraints?
Learning	What uncertainty can this MVP reduce, and what evidence would matter?

Use the five lenses to expose weak points, not to complete a ceremonial checklist. If desirability is uncertain, technical sophistication will not resolve the central question. If the proposal offers no meaningful distinction from the status quo, a longer feature list may increase cost without making the product more compelling. If the learning objective is vague, instrumentation may generate data without producing useful insight. Use Figure 4.1 to review a decision through all five lenses; they are not additional stages.

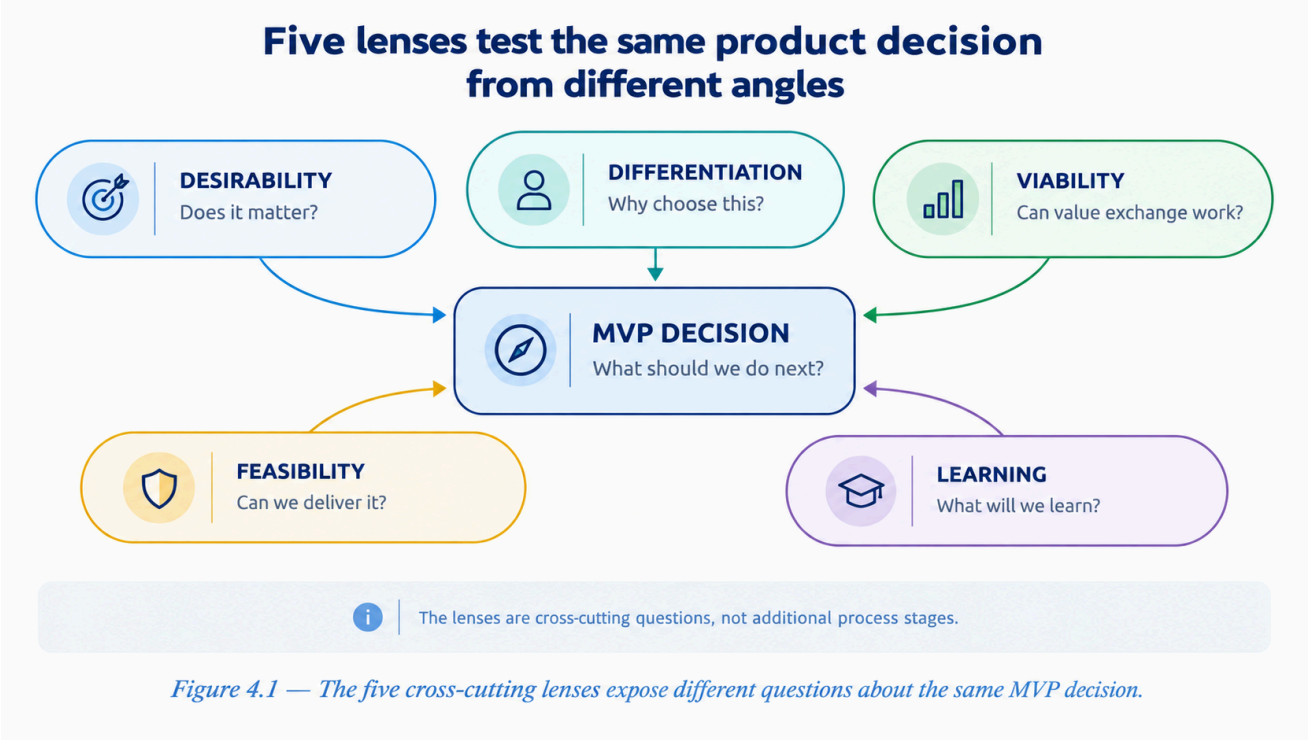


Figure 4.1 — The five cross-cutting lenses expose different questions about the same MVP decision.

Master Case: Related Product Framings and a Focused-MVP Hypothesis

The Master Case concerns a book-related product and brings together several framings found in the available source material: BookScout / Book Tracker or Book Finder, MyBookScout / Personal Book Intelligence, and Matchshelf as a focused-MVP framing. The available evidence does not establish that these names represent a verified chronological sequence, successive releases, or a documented product

history. This guide uses them as related teaching framings where the source material supports them, not as a claim that the product evolved in this order or that its history followed the 15 stages.

The case materials describe a proposed loop: ISBN Scan → Understand → Recommend → Explain → Learn → Improve. ISBN scanning may make library capture easier and reduce the cold-start problem, but it is not the core value by itself. The value hypothesis is that a reader can answer a more useful question: “Which book fits me — and why now?” Whether the proposed loop delivers that value remains to be tested.

EVIDENCE DISCIPLINE The available case package documents an elaborated product hypothesis and strategy. It does not establish that every assumption was historically validated or that the actual development followed the 15 stages in this exact order. The stages are used here as a methodological reconstruction, not as a claim about historical chronology.

Your turn: frame the idea

- Stage 1 — Output 1, Idea Brief: Write the raw idea in one sentence without adding persuasive language. Then record the trigger or context, the provisional user, the hoped-for outcome or benefit, and why the idea may matter.
- Stage 1 — Output 2, Idea Hypothesis: State provisionally who may experience what problem or situation, what outcome they may seek, and why the proposed idea might help. Keep it explicitly testable; do not present it as a validated fact.
- Separate what is known from assumptions and open questions. For each important claim, note whether it is evidence, hypothesis, derived reasoning, or an open gap; do not inflate confidence to make the idea sound stronger.
- Identify which of the five lenses currently contains the greatest uncertainty and what evidence could most usefully reduce it.

Stage 1 — Idea Ready check: The Idea Brief and Idea Hypothesis state the idea, trigger/context, provisional user and hoped-for outcome; major assumptions and open questions are visible. Do not treat this as validation.

OUTPUT AND GATE Output: an initial Idea Hypothesis. Gate: can you describe what the idea might be without presenting it as validated? If not, keep clarifying before adding features.

PART II — UNDERSTAND THE OPPORTUNITY

Clarify the problem before committing to the solution.

CORE QUESTION: What problem matters, for whom, and why might this product be chosen?

LEARNING OBJECTIVE: Produce a clear opportunity definition connecting problem, user, alternatives, outcome, value, and unresolved assumptions.

5. Capture the idea without polishing away uncertainty

Stage 1 records the idea as it currently exists: what prompted it, who it may serve, and what benefit it might provide. Resist the urge to resolve every uncertainty immediately. The Idea Brief preserves the original proposition; the Idea Hypothesis makes its uncertain parts visible. Keep both concise enough to revisit as you learn.

6. Define the problem without naming the app

A proposed solution can make the underlying problem harder to see. Test the problem statement by describing the situation without naming the app. Specify who is affected, when the problem occurs, what the person does today, where friction appears, and what consequence follows. Distinguish the observable problem from your explanation of why it occurs. Figure 5.1 illustrates the relationship discussed in this section.

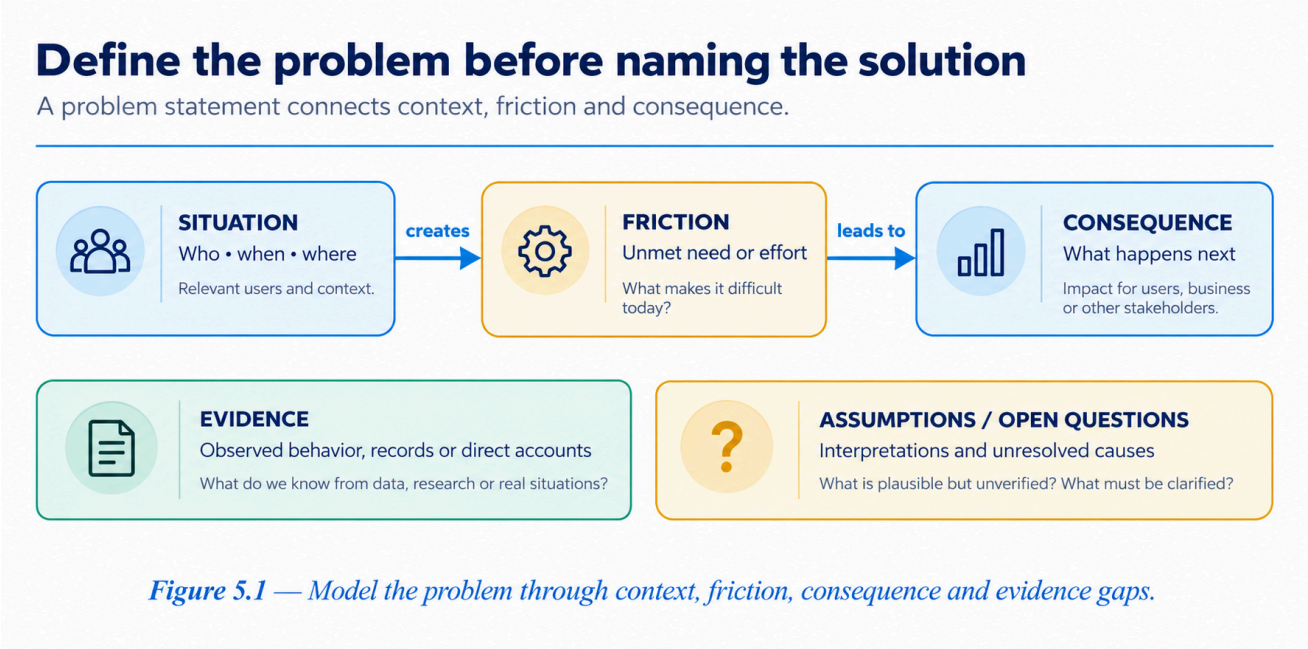


Figure 5.1 — Model the problem through context, friction, consequence and evidence gaps.

Evidence may come from observed behavior, interviews, existing records, usage data, or other relevant material. Early work will often contain gaps; the requirement is not to eliminate every gap before

proceeding, but to label claims honestly. If you have only an assumption, call it a hypothesis. If you infer a conclusion from limited inputs, make the reasoning visible and note what could weaken it.

7. Identify the primary user

"Everyone who reads" is too broad to guide early product decisions. The primary user is the person whose problem and desired outcome will shape the first version. Other people may influence a purchase, pay, supply data, or become users later. Keep these roles separate instead of assuming that user, buyer, and beneficiary are always the same person. Figure 5.2 illustrates the relationship discussed in this section.

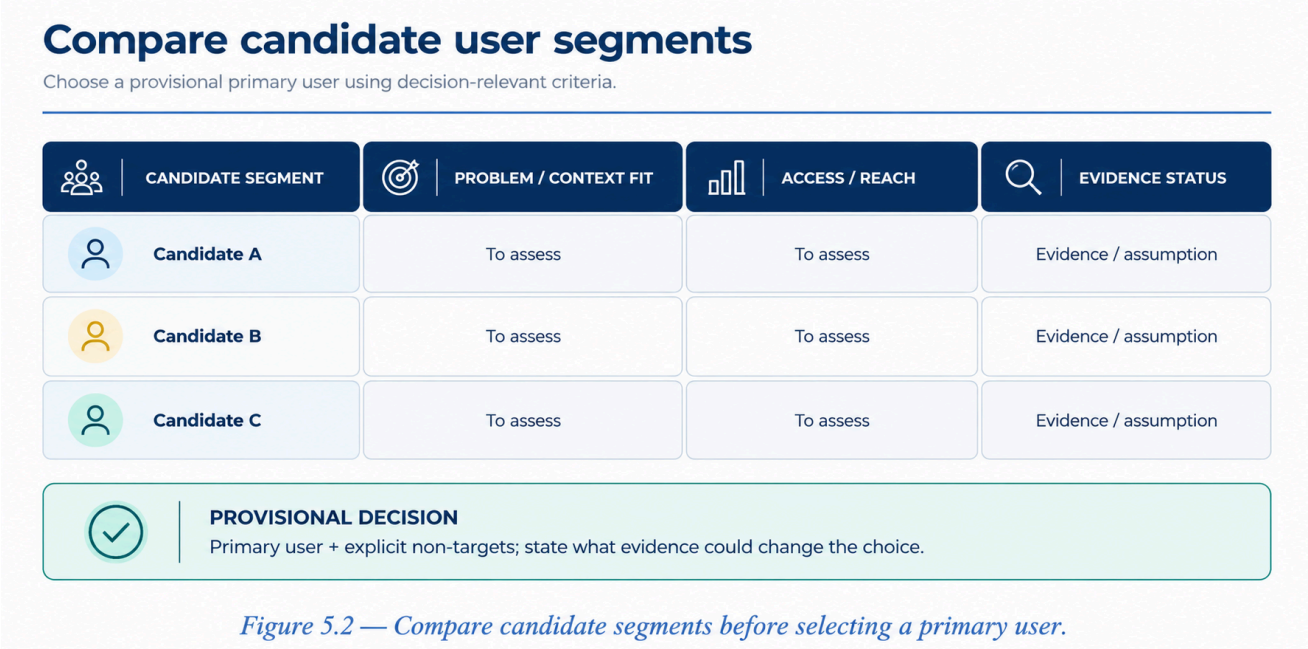


Figure 5.2 — Compare candidate segments before selecting a primary user.

A useful profile focuses on the situation, behavior, constraints, and needs that matter to the problem. Include demographic details only when they affect a product decision. Define who is not in the initial target as well: an explicit boundary helps the team resist adding features for every adjacent audience. Unless supporting evidence exists, present the profile and segment as provisional.

8. Understand the alternatives

When a problem occurs, people already respond somehow. They may use a competing product, ask someone for help, maintain a spreadsheet, browse a marketplace, postpone the decision, or accept the status quo. These behaviors are alternatives even when they are not direct competitors. Figure 5.3 illustrates the relationship discussed in this section.

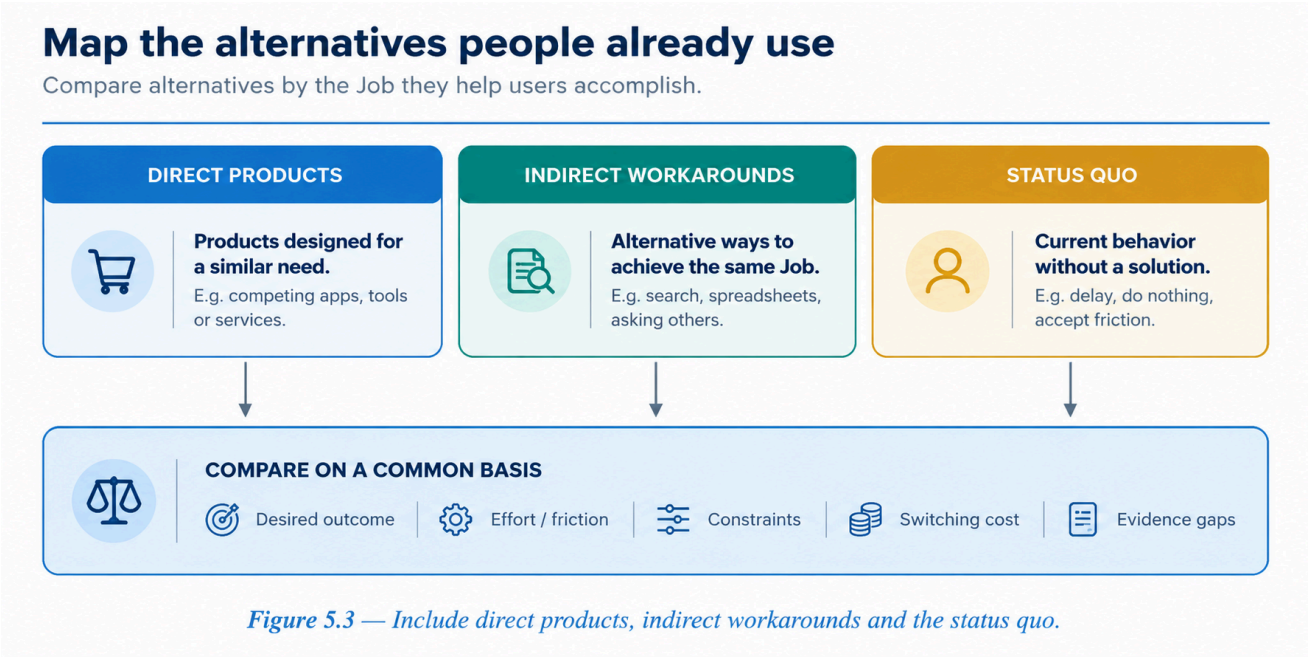


Figure 5.3 — Include direct products, indirect workarounds and the status quo.

Investigate what people do today, what works well enough, where it breaks down, and what would have to change for a new approach to be worth adopting. Market Discovery should inform a product decision, not become a catalogue that sits beside the work. For each research question, ask which hypothesis, scope choice, or next test the answer could change.

9. Define value and differentiation

A value proposition links a particular user and situation to a meaningful outcome, then explains why the proposed product might be preferable to the alternatives. Words such as “intelligent,” “seamless,” and “personalized” are not value by themselves. Explain what they would enable a user to do, and mark the claim as a hypothesis until evidence supports it. Figure 5.4 illustrates the relationship discussed in this section.

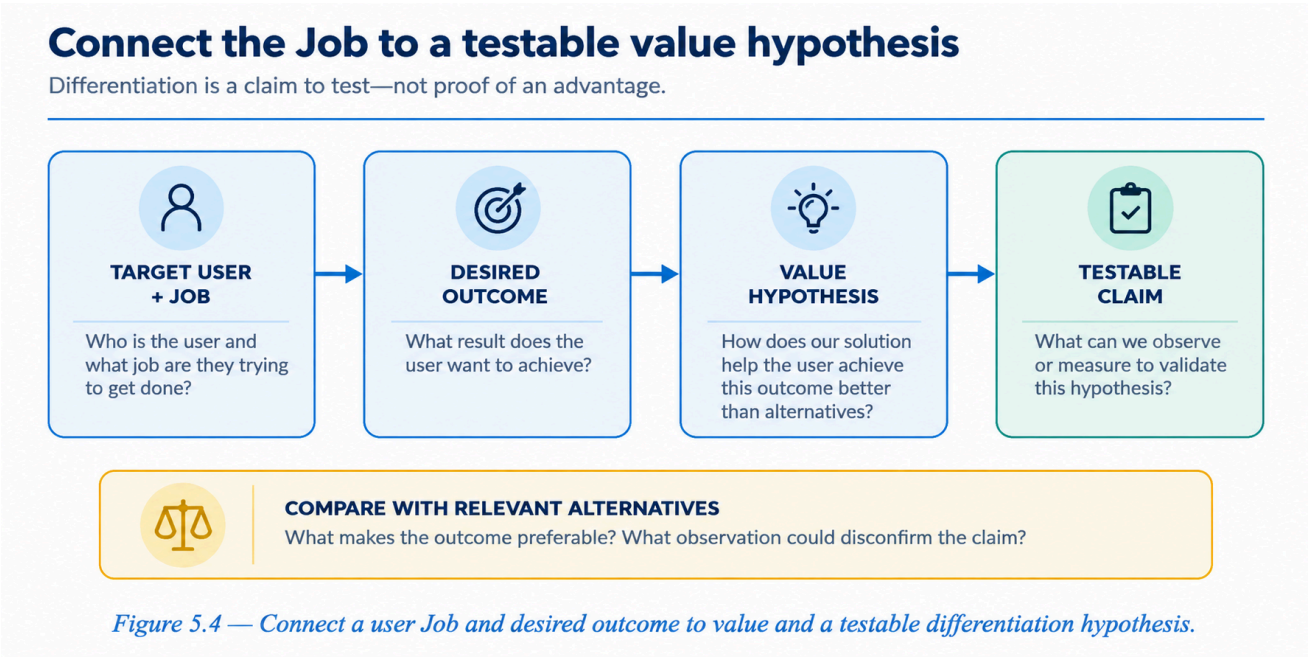


Figure 5.4 — Connect a user Job and desired outcome to value and a testable differentiation hypothesis.

In the Master Case, the problem hypothesis is more specific than "readers need more book discovery." It is framed as a Reader's Paradox: despite having access to books, reviews, rankings, lists, and recommendations, a reader may still find it difficult to decide which book fits their interests and circumstances now. This is a problem framing to investigate, not evidence that the experience is universal or prevalent. The proposed differentiation is that personal context and explainable recommendations could make the decision easier; that remains a hypothesis.

This framing changes the product question. A generic book finder may increase the number of options. A Personal Book Intelligence concept would need to use information about the individual reader, explain why a recommendation may fit, and learn appropriately from feedback. These are design hypotheses, not demonstrated capabilities or validated advantages. The next step is to identify which assumption is most consequential and what evidence could challenge it.

Opportunity brief: a compact working format

Field	Question
Problem	What friction or consequence occurs, for whom, and in what situation?
Primary user	Who experiences this problem most directly?
Current alternatives	What does the user do today, including the status quo?
Desired outcome	What should become easier, better, faster, safer, or more reliable?
Value hypothesis	Why might the proposed product help create that outcome?
Differentiation hypothesis	Why might it be preferred over relevant alternatives?
Evidence	What supports each important statement?
Open questions	What remains uncertain and could change the decision?

Your turn: test the opportunity

- Stage 2 — Output, Problem Statement: Describe the affected user, situation, current behavior, friction, and consequence without mentioning the proposed solution. Include current alternatives or workarounds where relevant.
- Supporting record — Evidence / Assumptions: Record current alternatives, workarounds and inaction where relevant. Separate supporting evidence from hypotheses, derived reasoning and open gaps. Record what could weaken the problem hypothesis.
- Stage 3 — Output 1, Primary User Profile: Describe one provisional primary user in the context and behavior relevant to the problem, including important constraints and why this user is the initial priority. Do not imply that the segment is validated.
- Stage 3 — Output 2, Target / Non-Target Segment: State who is included in the initial target segment and who is explicitly not targeted yet. Identify evidence that could confirm or weaken this choice. Treat the segment choice as a hypothesis where evidence is incomplete.
- Stage 4 — Output, Value Proposition: State the desired outcome and value proposition. Describe the expected benefits and formulate differentiation/USP and positioning hypotheses. Compare the proposed value with the status quo and relevant alternatives. Mark unsupported claims as hypotheses.

Stage 2 — Problem Ready check: The problem statement is solution-independent and makes the affected user, context, current behavior, alternatives/workarounds, friction, consequences and uncertainties explicit.

Stage 3 — User Ready check: The Primary User Profile describes the relevant context, behavior and constraints; target and non-target boundaries are explicit, and the choice remains a hypothesis where evidence is incomplete.

Stage 4 — Value Ready check: The desired outcome and value proposition are explicit and do not merely describe a feature list. Expected benefits are stated, the proposed value is compared with the status quo and relevant alternatives, and unsupported differentiation or positioning claims remain hypotheses.

OUTPUTS AND READINESS CHECKS

Stage 2 — Problem: Problem Statement, supported by an explicit Evidence / Assumptions record.
Gate: Can you explain the problem and why it matters without relying on the app idea itself?

Stage 3 — User: Primary User Profile and Target / Non-Target Segment.
Gate: Is the initial user choice explicit and grounded in relevant context, with uncertainty acknowledged where evidence is incomplete?

Stage 4 — Value: Value Proposition, expected benefits, differentiation/USP hypothesis, and positioning hypothesis.

Gate: Is the proposed value explicit, compared with the status quo and relevant alternatives, with unsupported claims identified as hypotheses?

PART III — DEFINE THE PRODUCT

Model what the product is before designing its screens.

CORE QUESTION: What must the product fundamentally represent and help users accomplish?

LEARNING OBJECTIVE: Connect the value proposition to a product model, lifecycle, core jobs, and a plausible business model hypothesis.

10. Model the product

A product model describes the important things the product needs to represent and how they relate. It is not a screen map. Start with the central entity or concept, then identify supporting entities and relationships. Ask which information must persist, which is temporary, who owns it, and what product behavior depends on it. Figure 6.1 illustrates the relationship discussed in this section.

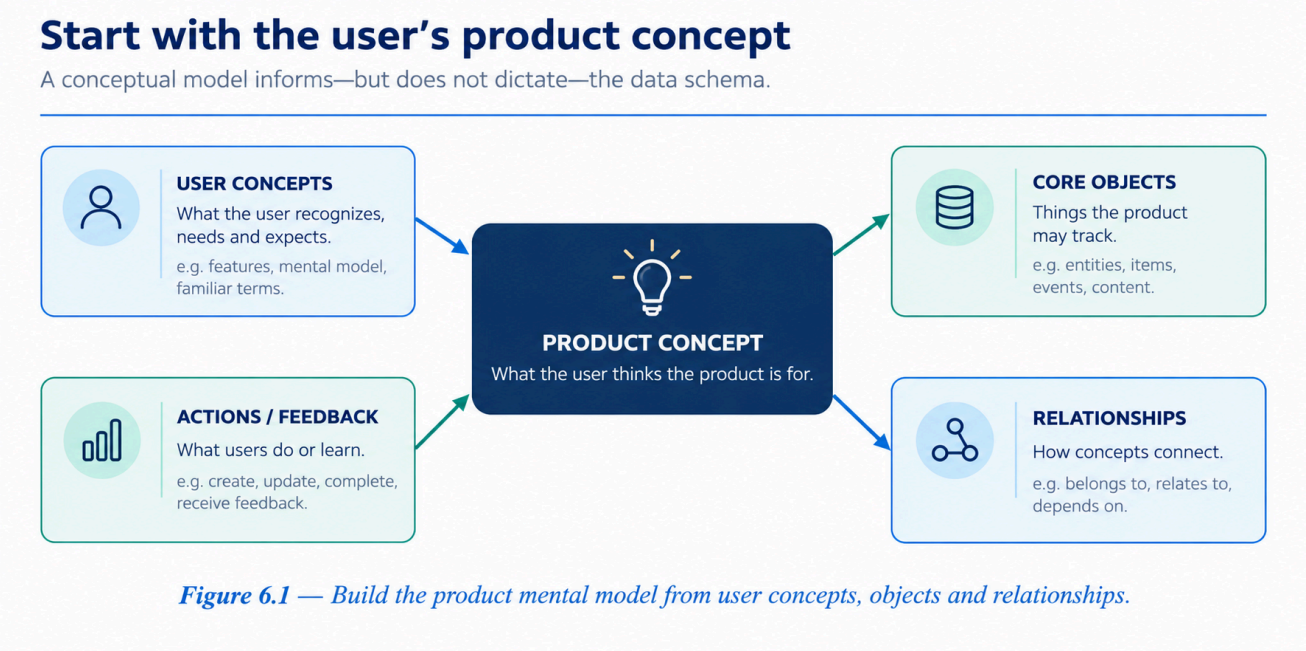


***Figure 6.1** — Build the product mental model from user concepts, objects and relationships.*

In the MyBookScout concept, the reader's personal library is a possible foundation for understanding individual taste, not necessarily the end product. Books, reading status, ratings, feedback, reading behavior, and contextual signals could contribute to that understanding. Include each only if its role is clear and its value justifies the added data and complexity. The case materials do not establish a final implementation schema.

11. Model context and lifecycle

Important product objects change over time. A book might be captured, identified, added to a library, marked as planned, started, finished, rated, or used as feedback. The relevant states depend on the product's purpose. A lifecycle model names meaningful states, transitions between them, and events that trigger change; it should not attempt to enumerate every imaginable state.

Figure 6.2 illustrates the relationship discussed in this section.

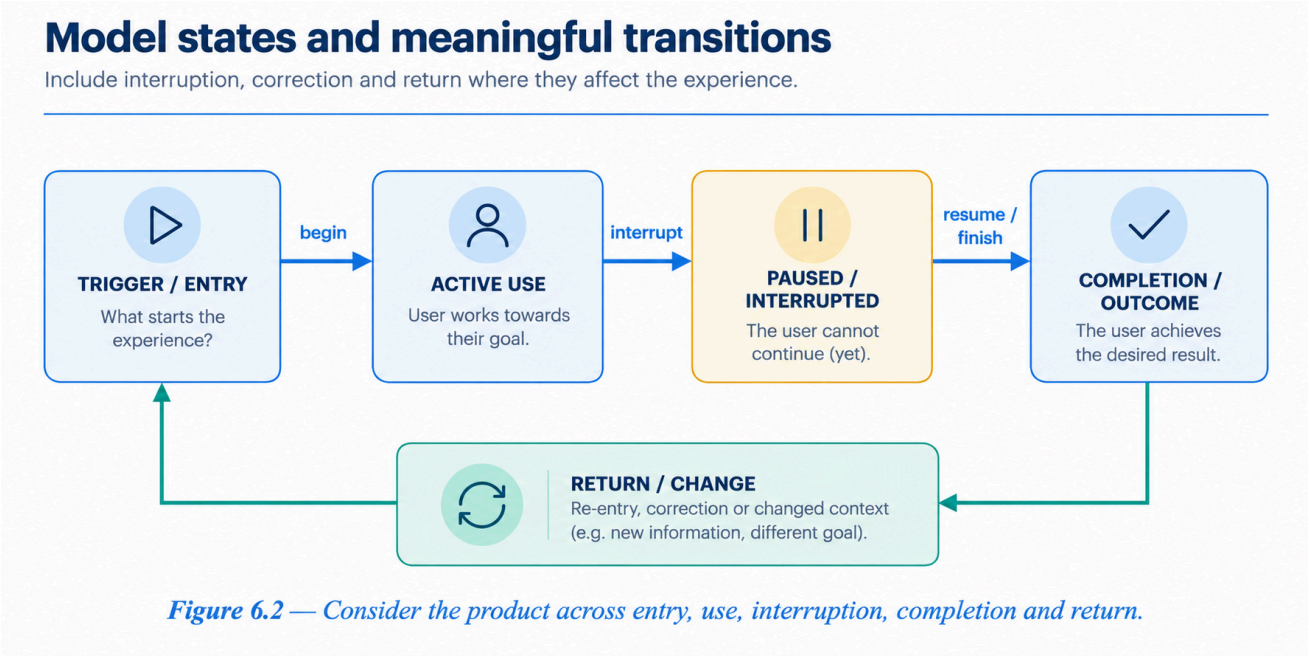


***Figure 6.2** — Consider the product across entry, use, interruption, completion and return.*

Lifecycle questions often reveal requirements that a static entity list misses. Can the same item be added twice? What happens when identification fails? Can a user correct a mistaken match? What changes after a recommendation is dismissed? You do not need to solve every edge case at once. First identify the transitions that affect core value, data integrity, user control, or later learning.

12. Define Jobs

A Job describes what a user is trying to accomplish or make progress on in a particular situation. It is not a feature or interface action. “Scan a book” names a mechanism; “capture my existing library without tedious manual entry” describes the progress the user wants. “Show recommendations” names a system behavior; “choose a book I am likely to value for my current mood or needs” expresses a user-level outcome.

Use Jobs to connect user intent to the capabilities the product may need. A useful Job statement makes the situation or trigger, the progress sought, and the desired outcome clear enough to guide prioritization. Keep the initial set focused: every additional Job can introduce new features, dependencies, and competing demands. Prioritize by relevance to the primary user and outcome, not by how easy a feature sounds to build.

Throughout this guide, *Job* means what a user is trying to accomplish or make progress on in a particular situation. Jobs-to-be-Done (JTBD) may support analysis of user motivation and context, but it is not a separate MVPCompanion stage, lens, or mandatory artifact. MD-001 establishes this reader-facing terminology; its formal incorporation into the frozen Method Baseline remains a separate controlled change.

13. Make the business model hypothesis explicit

Business-model thinking belongs in product definition because a product must create value within real constraints and incentives. Ask who uses the product, who benefits, who might pay, and what they might pay for. These roles can overlap, but do not assume they do. The answers should inform the value proposition and the tests needed before committing to a commercial model.

At this stage, the business model is a hypothesis, not a business plan. A plausible way to capture value does not prove willingness to pay; a proposed price does not establish demand. Record the assumptions behind the model and identify evidence that could confirm, weaken, or reject them. Keep pricing ranges explicitly hypothetical unless actual research or transactions support a stronger claim.

Master Case: from book collection to personal decision support

The Master Case treats the reader's library as an input to personal decision support, rather than as the product's final destination. Its proposed loop is Scan → Understand → Recommend → Explain → Learn → Improve. The collection may provide an initial signal; a developing model of taste may inform recommendations; explanations may help readers judge those recommendations; and feedback may inform later decisions. The materials describe this as a product concept, not a proven operating loop.

The proposed loop exposes dependencies that need testing. If scanning is slow or unreliable, readers may not capture enough of their collection. If the captured library provides too little useful signal, personalization may remain weak. If recommendations lack credible explanations, readers may not trust them. If feedback is not captured or does not improve later recommendations, the learning loop may add complexity without delivering value. Treat these as failure hypotheses to test, not as observed outcomes.

Your turn: model the product

- Stage 5 — Output 1, Product Mental Model: In a few sentences, explain the central product concept and how it is intended to create value.
- Stage 5 — Output 2, Entity Map: Identify the central entity and only the supporting entities the product genuinely needs. Distinguish entities and their relationships from actions, workflows, and screens.
- Stage 6 — Output, Context & Lifecycle Model: Map the relevant context, meaningful states, entry conditions, transitions or triggering events, and recurring loops where relevant. Show how entities participate in relevant lifecycle states and transitions. Include only states that affect core value, behavior, or data integrity.
- Stage 7 — Output, Core Jobs: Draft three to seven candidate Jobs in the user's language. For each Job, state the situation or trigger, the progress sought, and the desired outcome. Prioritize the Jobs and connect the highest-priority Jobs to the primary user and the outcome they seek.
- Cross-cutting business-model check (not a Stage 5–7 output): State who receives value, who uses the product, and who might pay; mark uncertain assumptions. Keep this hypothesis distinct from the Product Mental Model, Entity Map, Context & Lifecycle Model, and prioritized Job set.

Stage 5 — Product Ready check: The Product Mental Model explains the central product concept; the Entity Map distinguishes persistent entities and their relationships from actions, workflows and screens.

Stage 6 — Context & Lifecycle Ready check: Relevant contexts, states, entry conditions and transitions are explicit where they affect core value, behavior or data integrity; recurring loops are included where relevant.

Stage 7 — Core Jobs Ready check: Each prioritized Job connects to a user situation/trigger, progress sought and desired outcome; priorities and secondary Jobs are explicit enough to inform scope.

OUTPUTS AND READINESS CHECKS

Stage 5 — Product: Product Mental Model and Entity Map.
Gate: Does the Product Mental Model explain the central product concept, and does the Entity Map distinguish entities and their relationships from actions, workflows, and screens?

Stage 6 — Context & Lifecycle: Context & Lifecycle Model.
Gate: Are the relevant contexts, states, entry conditions and transitions explicit where they affect core value, behavior or data integrity?

Stage 7 — Jobs: Prioritized Core Jobs.
Gate: Does each prioritized Job identify the situation or trigger, progress sought and desired outcome, with priorities clear enough to inform scope?

Cross-cutting check: Record the business-model hypothesis separately, including who receives value, who uses the product and who might pay. Mark uncertain assumptions.

PART IV — SHAPE THE MVP

Derive the smallest coherent experience from the jobs.

CORE QUESTION: What capabilities are necessary for users to reach the intended value?

LEARNING OBJECTIVE: Translate core jobs into capabilities and features, then design a coherent journey, information architecture, and data model.

14. From jobs to capabilities and features

A capability is something the product must be able to do to support a Job. A feature expresses that capability through a more concrete product behavior. Keep the relationship explicit: connect each proposed capability to a prioritized Job and its intended outcome. If the link is unclear, the capability may be unnecessary, premature, or based on an assumption that needs testing. Figure 7.1 illustrates the relationship discussed in this section.

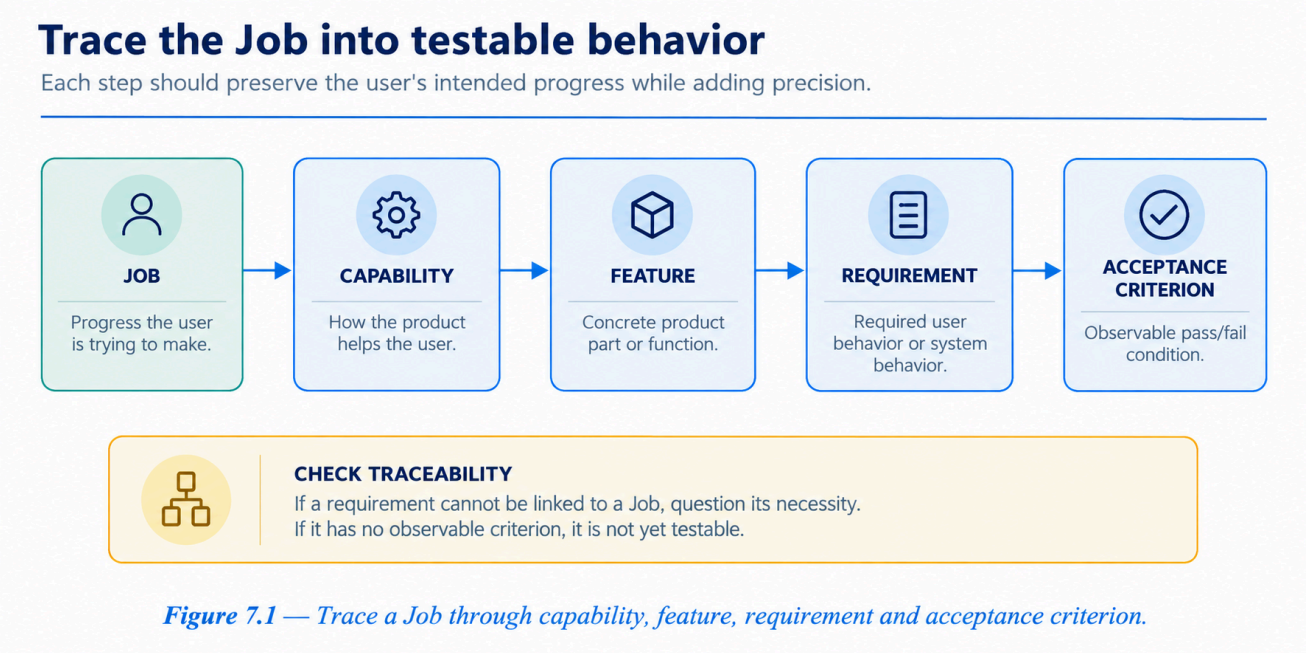


***Figure 7.1** — Trace a Job through capability, feature, requirement and acceptance criterion.*

Here the traceability chain becomes practical: Problem → User → Outcome → Job → Capability → Requirement → Acceptance Criterion → Metric. The chain explains why a decision is being made and how its intended effect could be assessed; it does not imply that every link is equally certain or already validated. Mark hypotheses and derived reasoning where evidence is incomplete.

For Matchshelf, candidate capabilities could include capturing a reader's books, maintaining a personal library, generating a small set of personalized recommendations, explaining why a recommendation might fit, and collecting feedback. These are candidate capabilities consistent with the documented focused-MVP direction, not proof of a finalized feature set. Their inclusion should depend on the prioritized Jobs, evidence, dependencies, and final Stage 15 boundary decision.

15. Design the core experience

The core experience is the shortest coherent journey through which a user can reach the product's intended value. Map the path from entry to the first meaningful outcome. Show what the user does, what

information the system needs, what the system does, and what response the user receives. Keep the journey focused on the outcome rather than on showcasing every available feature. Figure 7.2 illustrates the relationship discussed in this section.

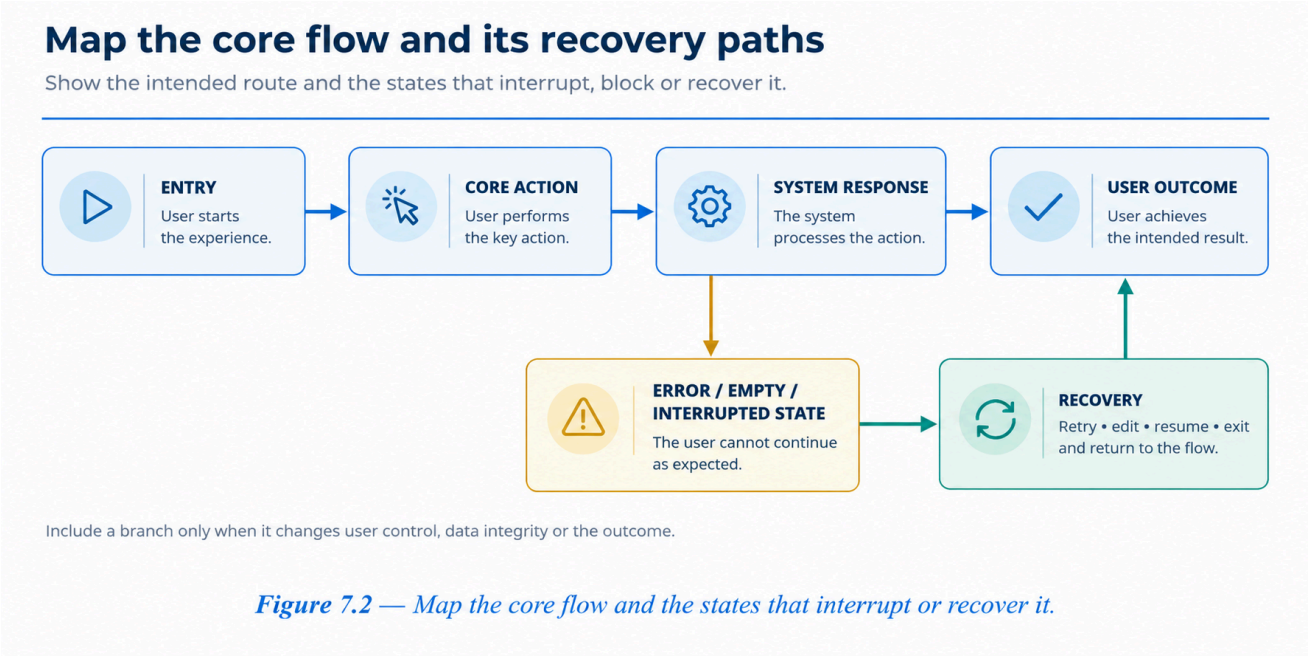


Figure 7.2 — Map the core flow and the states that interrupt or recover it.

For a book-intelligence MVP, one possible journey begins with library capture, moves to an initial understanding of the reader, and ends with a recommendation accompanied by an explanation. Each step must earn its place by enabling the next. If a lengthy questionnaire delays the first useful result, test whether all of its inputs are necessary at that point or can be collected later.

Design the empty state as deliberately as the ideal path. What can the user do before adding books? How does the product respond when a scan fails or the available information is insufficient for a useful recommendation? A focused MVP need not handle every conceivable case, but it must not ignore failure states that interrupt its core promise. Identify the few recovery paths that matter most.

16. Information architecture

Derive information architecture from the product model, prioritized Jobs, provisional MVP scope, and core experience. Name the main destinations or areas, then map the hierarchy of important objects, list/detail relationships, and links needed to complete the core journey. Do not give every capability its own destination by default. Figure 7.3 illustrates the relationship discussed in this section.

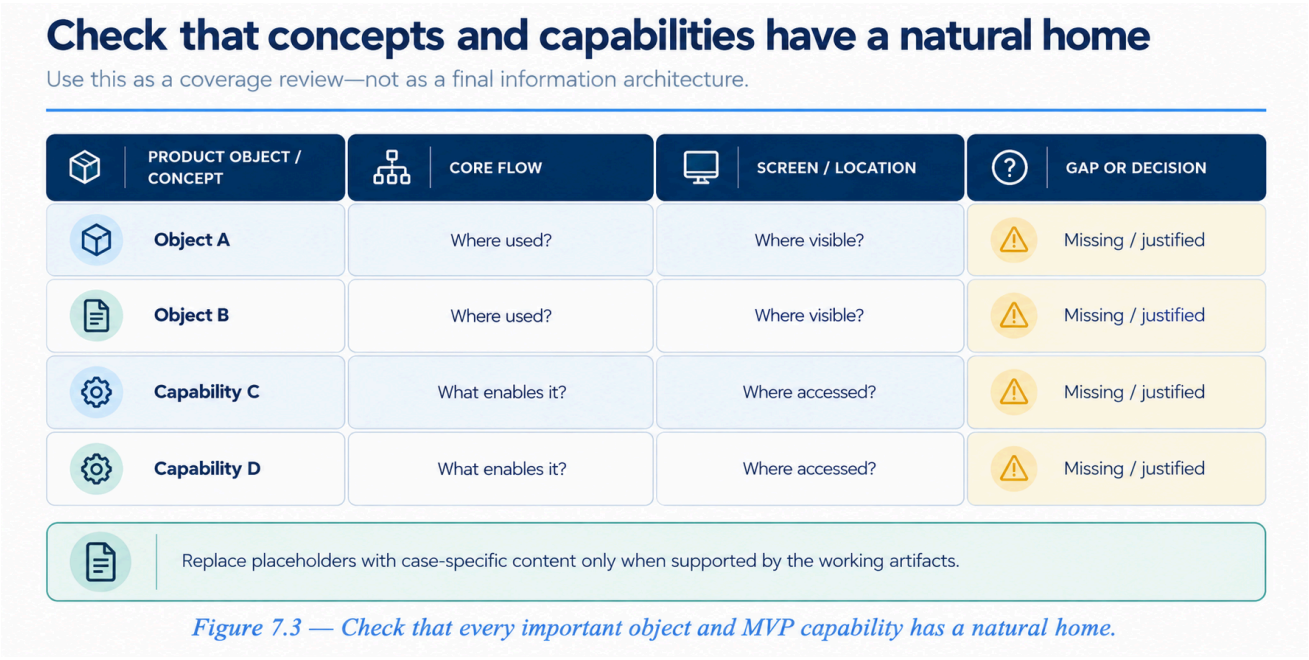


Figure 7.3 — Check that every important object and MVP capability has a natural home.

For each MVP capability, identify where it belongs and how the user reaches it. Look for orphaned capabilities, overloaded top-level navigation, and structures that mirror internal teams or technical layers rather than the user's mental model. The Master Case's Library → Taste → Discovery → Decision → Learning hierarchy is conceptual; it is not evidence of a validated information architecture or navigation model.

Stage output: an Information Architecture + Navigation Model. The IA Ready check is met when the hierarchy and navigation are explicit and every MVP capability has a natural home. This stage depends on Product, Jobs, Scope, and Experience; its decisions inform Data and Requirements.

Information architecture organizes the product's objects and capabilities so users can find and understand them. Derive it from the product model and core experience instead of copying an organizational chart or accumulating navigation items.

A simple product may need only a few destinations. The test is not whether navigation looks sophisticated, but whether the structure reflects the user's mental model and supports the prioritized Jobs. Before adding a section, ask whether it represents a distinct user need or merely exposes an internal implementation detail.

17. Define the data model

Begin with entities and relationships, not a long list of fields. For each entity, specify the minimum required and optional fields, relationship cardinality, relevant status or lifecycle fields, ownership, and timestamps or audit information that materially affect the MVP. Distinguish user-entered information from imported data and system-derived values.Figure 7.4 illustrates the relationship discussed in this section.

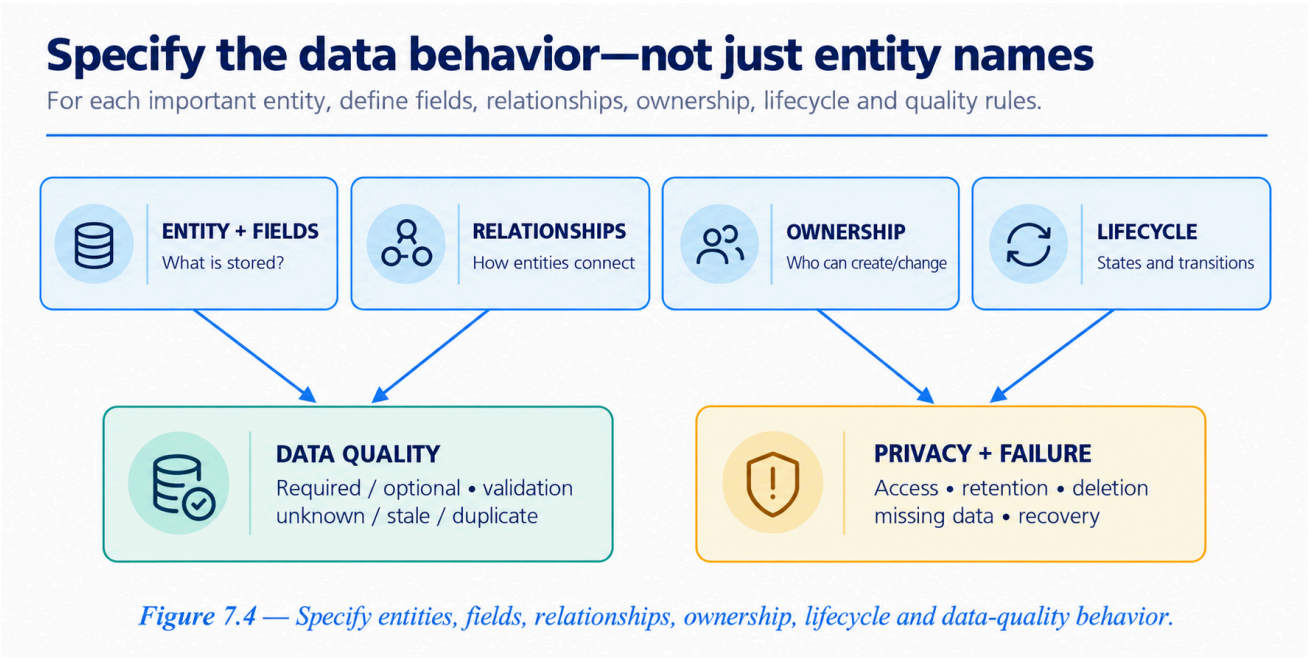


Figure 7.4 — Specify entities, fields, relationships, ownership, lifecycle and data-quality behavior.

For every field or relationship, identify the Job, capability, requirement, or learning question that depends on it. Decide how the product handles missing, inaccurate, duplicated, stale, or corrected data. Where an external source is required, verify that it can supply the information needed under relevant access, quality, and usage constraints; do not assume availability.

Stage output: an MVP Entity/Data Model. The Data Ready check is met when essential entities, fields, relationships, ownership, and lifecycle data are explicit. This stage depends on Product, Context & Lifecycle, Jobs, Scope, Experience, and Information Architecture; it informs Requirements and Technical Boundaries.

The data model describes the information the product must store or derive, how it relates, and the ownership and lifecycle rules that matter. Start from the product model and requirements. Do not collect or store information simply because it might become useful someday.

For each field, ask what behavior or decision depends on it. Is it entered by the user, supplied by an external source, or derived by the system? Can it be missing or wrong? Can the user correct it? What happens when the underlying object changes? Data quality deserves particular attention when the product's value depends on interpreting personal information or generating recommendations.

A traceability example

Link	Illustrative statement for the case
Problem	A reader has many plausible books and struggles to choose what fits now. (H)
User	A reader who wants help making a personal next-book decision. (H)
Outcome	Choose a suitable next book with greater confidence. (H)
Job	Decide which book is worth reading next in the current context. (D/H)
Capability	Generate a small set of personalized recommendations and explain the fit. (D)
Requirement	The system presents a recommendation and a concise explanation based on available reader and book information. (Proposed)
Acceptance criterion	Given a user with sufficient library data, when recommendations are requested, the system displays at least one candidate and a human-readable reason; insufficient-data behavior is also defined. (Proposed)
Metric	Measure recommendation engagement and user-rated usefulness, with the interpretation and collection method defined in advance. (Proposed)

This table is illustrative, not evidence that the case has validated the problem or that the exact requirement has been implemented. It shows how a proposed behavior can be traced backward to a user outcome and forward to an observable test.

Verification note: this traceability table is a single-capability illustration spanning problem through metric; it is not a complete worked example for Stages 8–15. The capability, requirement, acceptance criterion, and metric are proposed/derived, not observed results. It does not by itself demonstrate a complete MVP Feature Map, Out-of-Scope List, Core User Flows, Screen Inventory, IA/Navigation Model, Entity/Data Model, complete requirements set, technical boundary decision, or final MVP Boundary Statement.

Your turn: derive the MVP

Example coverage note: the preceding traceability example demonstrates the logic of one proposed capability only. Use the exercise below to produce the distinct Stage 8–11 artifacts; do not treat the table as evidence that those complete artifacts already exist or have been validated.

- Stage 8 — Output 1, MVP Feature Map: For each prioritized Job, name the outcome and minimum capability needed. For every proposed MVP capability, record its supporting Job, rationale, priority (Must / Should / Later / Out), dependencies, and any material assumption. Treat this as the first scope proposal, not the final MVP boundary.
- Stage 8 — Output 2, Out-of-Scope List: Name the capabilities or features excluded from the current MVP and why. Record important dependencies or conditions that could change the decision. Check that every included capability supports a Job or an explicit learning objective.
- Stage 9 — Output 1, Core User Flows: Map entry → first meaningful value → core outcome. Include essential empty, error, permission, insufficient-data, and recovery states where relevant. Show what the user does, what the system needs, and how the product responds.
- Stage 9 — Output 2, Screen Inventory: Derive a screen/view inventory from the flows; identify the purpose of each view and which flow/state reaches it.
- Stage 10 — Output 1, Information Architecture and Output 2, Navigation Model: show top-level destinations, object hierarchy, list/detail relationships, cross-links, and a natural home and route for every MVP capability. Check for orphaned capabilities.
- Stage 11 — Output, MVP Entity/Data Model: define essential entities, required/optional fields, relationships/cardinality, ownership, lifecycle, and whether data is manually entered, imported, or computed. For relevant fields, state what happens when data is missing, inaccurate, duplicated, or stale, and which behavior or decision depends on the field.

Stage 8 — Scope Ready check: The MVP Feature Map traces every included capability to a prioritized Job or explicit learning objective; dependencies and assumptions are visible; the Out-of-Scope List is explicit. This remains a provisional scope proposal until Stage 15.

Stage 9 — Experience Ready check: Core User Flows reach the intended outcome and include important empty, error, permission, insufficient-data, and recovery states where relevant; the Screen Inventory follows from the flows rather than driving them.

Stage 10 — IA Ready check: The Information Architecture and Navigation Model are both explicit; every MVP capability and important object has a natural home and a route through the core journey; orphaned capabilities and unnecessary destinations have been challenged.

Stage 11 — Data Ready check: The MVP Entity/Data Model defines the essential entities, fields, relationships, ownership and lifecycle rules, and relevant data sources and quality handling. Each material field or relationship has a clear behavioral or decision rationale. The model is not presented as a final implementation schema.

OUTPUTS AND READINESS CHECKS

Stage 8 — Scope: MVP Feature Map and Out-of-Scope List.
Gate: Can every included capability be traced to a prioritized Job or explicit learning objective, with dependencies and material assumptions visible?

Stage 9 — Experience: Core User Flows and Screen Inventory.
Gate: Do the flows reach the intended outcome and account for important exceptional states, with the screen inventory derived from the flows?

Stage 10 — Information Architecture: Information Architecture and Navigation Model.
Gate: Does every MVP capability and important object have a natural home and a route through the core journey, without orphaned capabilities or unnecessary destinations?

Stage 11 — Data: MVP Entity/Data Model.
Gate: Are essential entities, fields, relationships, ownership, lifecycle and relevant data-quality rules explicit, with a clear rationale for material data requirements?

PART V — MAKE IT BUILDABLE

Specify behavior so another person does not have to guess.

CORE QUESTION: What exactly must be built, and how will we learn whether it works?

LEARNING OBJECTIVE: Write observable requirements and acceptance criteria, justify AI/integrations, and connect hypotheses to metrics and learning decisions.

18. Write testable requirements

A requirement describes product behavior precisely enough that another person can implement and assess it without inventing missing definitions. “Make recommendations smart” is not testable. State the context, inputs, expected behavior, outputs, and constraints that matter to the user-visible result. *Figure 8.1 illustrates the relationship discussed in this section.*

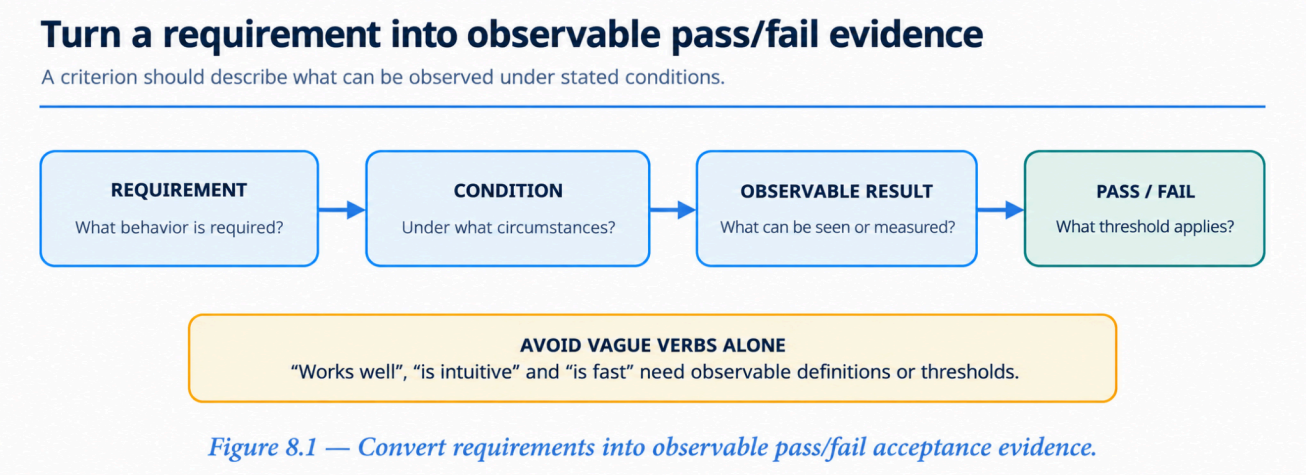


Figure 8.1 — Convert requirements into observable pass/fail acceptance evidence.

Requirements should specify the behavior the product must deliver, not dictate implementation details without a reason. State what must happen and under which conditions. Technical design can then choose an appropriate way to deliver that behavior within the agreed technical boundaries and constraints.

Acceptance criteria make a requirement observable and checkable. Include the normal path and relevant conditions involving missing or invalid inputs, insufficient information, permissions, or external failures. “Works well” only restates the goal. A more useful criterion describes what the user can do and how the system responds when, for example, a book cannot be identified.

19. Justify AI and integrations

Use AI and external integrations only when they support a real Job, a defined product need, or a meaningful learning objective. They can add dependencies, failure modes, costs, latency, data-handling obligations, and uncertainty. Their presence is not evidence of product quality and should not be justified by novelty or marketing appeal.

For each proposed AI or integration component, specify its input, expected output, user-visible behavior, likely failure modes, and fallback. Decide how the product responds when a service is

unavailable, returns poor-quality data, or cannot produce a confident answer. Make uncertainty visible whenever it affects the user's decision.

The AI Companion used during specification work should be held to the same standard. It can draft candidate requirements, generate edge cases, and challenge ambiguity, but its output is not automatically correct or evidence-backed. A human must verify that each requirement reflects an actual product decision and does not introduce unsupported assumptions.

20. Define learning hypotheses and metrics

An MVP can support learning, but collecting metrics does not automatically produce it. Start with a question whose answer could change a decision. State the hypothesis, identify evidence that would strengthen or weaken it, and decide in advance what action different outcomes might justify. Figure 8.2 illustrates the relationship discussed in this section.

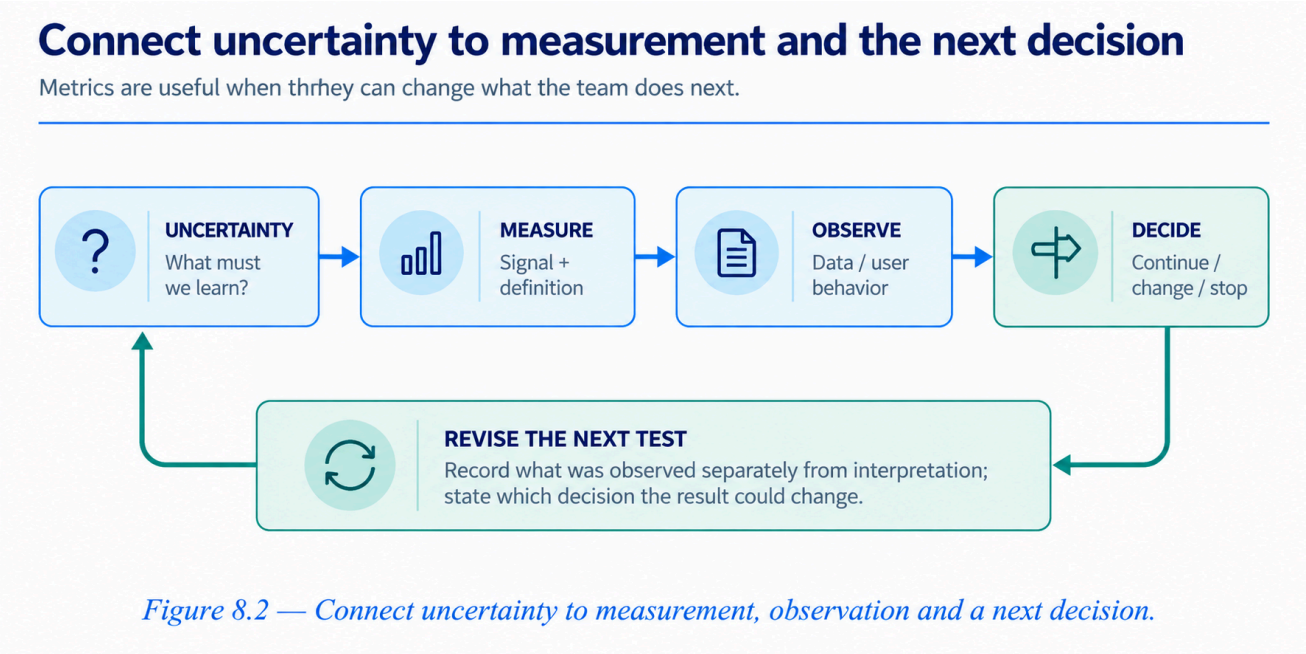


Figure 8.2 — Connect uncertainty to measurement, observation and a next decision.

Choose metrics that relate to the intended behavior and outcome. A scan count may show that people use the capture mechanism, but it does not establish that recommendations are useful. A recommendation click may signal interest without proving satisfaction or a better decision. Interpret each signal in context and, where needed, combine behavior data with direct feedback or other evidence.

Define the metric before seeing the result whenever practical. Specify the event or signal, unit of analysis, collection point, interpretation limits, and decision it is intended to inform. Avoid universal thresholds without a basis. What counts as meaningful evidence depends on the hypothesis, context, sample, risk, and decision at stake.

Master Case: what would count as learning?

For Matchshelf, one learning question could be: can the available library and interaction signals support recommendations that readers consider relevant enough to act on? A related hypothesis is that readers with sufficiently informative libraries will find at least some explained recommendations useful. This is a testable hypothesis, not a documented result.

Potential signals include completion of library capture, inspection of recommendations, saving or acting on a recommendation, and a user's assessment of its fit. None of these signals alone proves the product's broader value. A test plan should specify which uncertainty each signal addresses, what limitations affect its interpretation, and what decision would follow from different results.

Your turn: make one capability testable

- Stage 12 — Output 1, Functional Requirements: Choose one in-scope capability to demonstrate the method. Write a requirement that specifies context, inputs, behavior, outputs, and relevant constraints. The complete Stage 12 artifact must cover all in-scope functional requirements, not only this example capability.
- Stage 12 — Output 2, Acceptance Criteria: Write observable criteria for the normal path and at least one important failure, validation, permission, integration, or insufficient-data state where relevant. Criteria must be testable and must not merely restate the desired outcome.
- Stage 13 — Output 1, Learning Questions: State the uncertainty or hypothesis this capability is meant to address, and what decision could change depending on the evidence.
- Stage 13 — Output 2, Success Metrics: Choose an observable metric or evidence source. Define the event/signal, unit of analysis, collection point, interpretation limits, and the decision different results could inform. Keep planned tests separate from observations, inferences, and results/status.

Stage 12 — Requirements Ready check: Functional Requirements and Acceptance Criteria cover the in-scope behavior at a level another person can implement and assess without guessing; relevant validation, permissions, integrations, constraints and failure states are addressed.

Stage 13 — Learning Ready check: Critical Learning Questions are connected to explicit hypotheses, Success Metrics or evidence sources, interpretation limits, and possible decisions. Planned evidence is not described as an observed result.

OUTPUTS AND READINESS CHECKS

Stage 12 — Requirements: Functional Requirements and Acceptance Criteria.
Gate: Could another person implement and assess the in-scope behavior without guessing, including relevant constraints and exceptional states?

Stage 13 — Learning: Learning Questions and Success Metrics.
Gate: Are the critical uncertainties connected to observable evidence, interpretation limits and decisions that could change? Are planned tests clearly distinguished from observations and results?

21. Define technical boundaries

Technical boundaries identify what must be decided to make the MVP buildable and estimable. They may include platform, data storage, authentication, external services, AI dependencies, security-sensitive behavior, and constraints imposed by existing systems. Include a boundary when it affects scope, risk, cost, or a responsible implementation decision. Figure 9.1 illustrates the relationship discussed in this section.

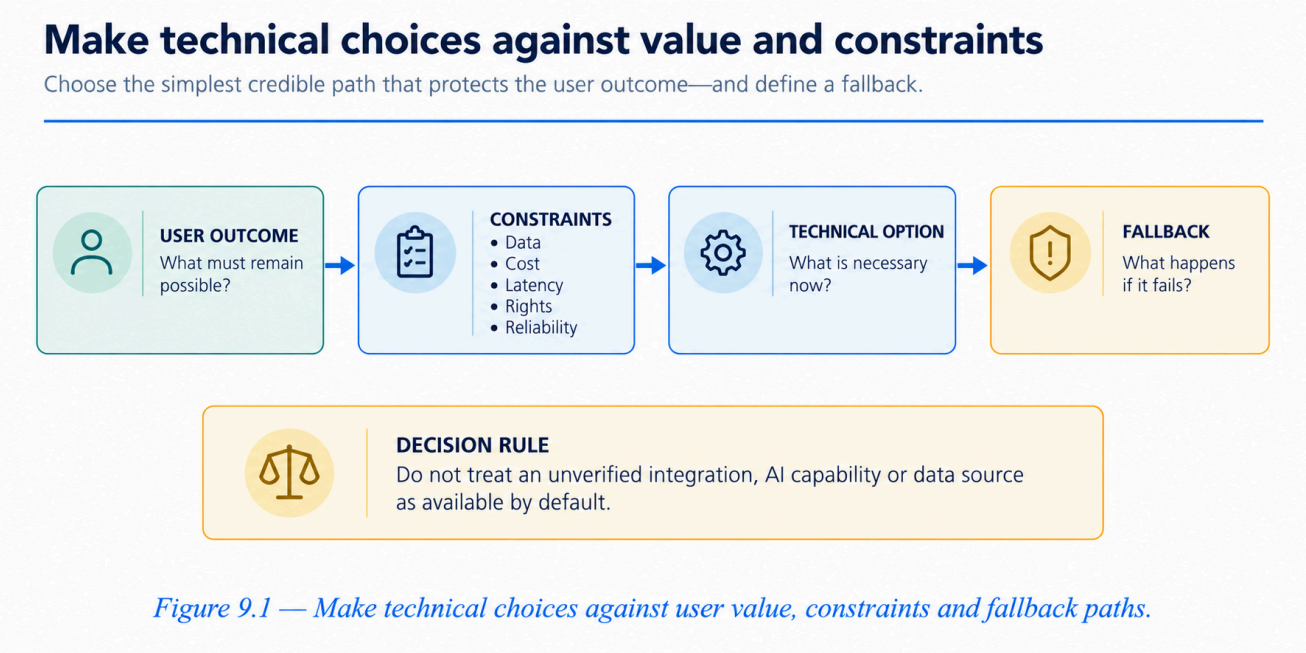


Figure 9.1 — Make technical choices against user value, constraints and fallback paths.

A technical boundary is not a complete future architecture. Specify what must be known to build and assess the MVP, what remains uncertain, and what can safely wait. Designing too far ahead creates cost and commitment before critical product uncertainties are reduced; leaving essential constraints vague creates avoidable implementation risk.

Record technical assumptions and risks that could change the product decision or the build plan. If a requirement depends on an external data source, establish whether the source can provide the needed information under relevant access, quality, rights, and usage constraints. If a model or service is involved, define what happens when it is slow, unavailable, expensive, or wrong.

PART VI — DRAW THE MVP BOUNDARY

Choose what to build now—and what to leave out.

CORE QUESTION: What is the smallest coherent product that can deliver value and support learning?

LEARNING OBJECTIVE: Define technical boundaries, make explicit scope decisions, assemble the Integrated MVP Specification, and assess readiness.

22. Decide what belongs

The final MVP boundary is a product decision, not simply a project-management exercise. At Stage 15, revisit the earlier scope proposal against the core promise, primary user, prioritized Jobs, end-to-end experience, data needs, requirements, learning objectives, technical boundaries, dependencies, and unresolved risks. The question is whether the pieces form a coherent, testable first product. Figure 9.2 illustrates the relationship discussed in this section.

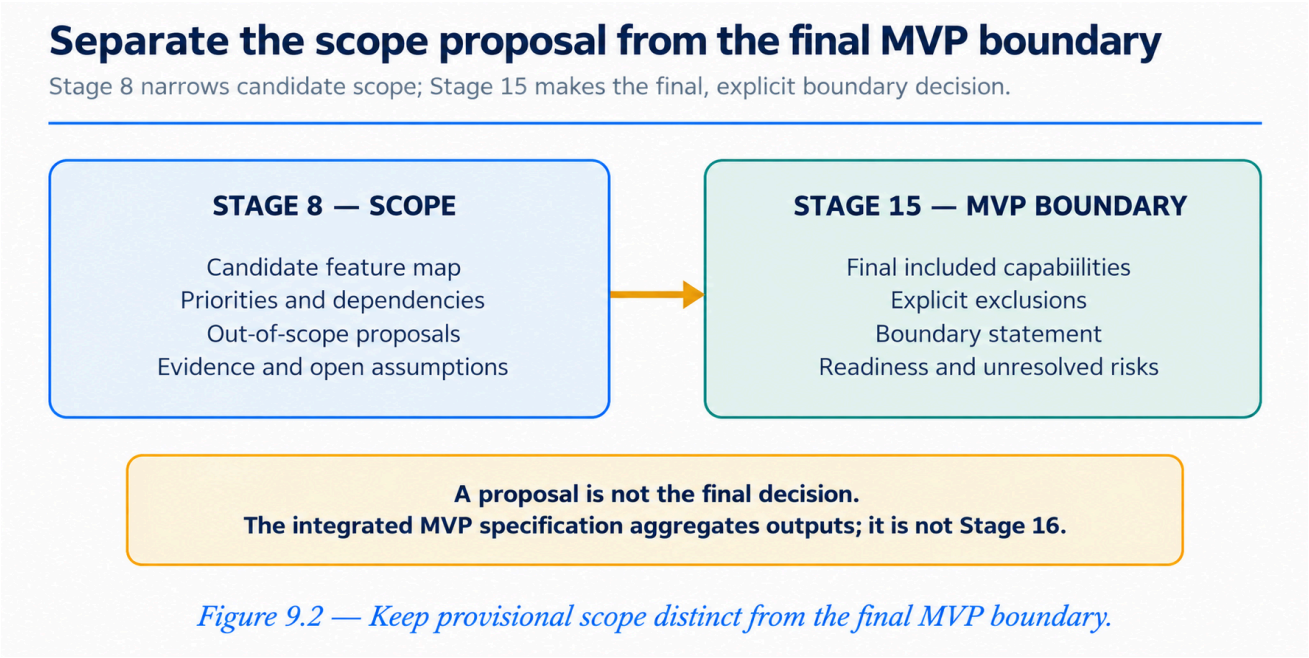


Figure 9.2 — Keep provisional scope distinct from the final MVP boundary.

Use Must, Should, Later, and Out to make scope decisions explicit. Must means necessary for the current coherent MVP. Should means useful but not essential to the first version. Later means deliberately deferred. Out means excluded from the current scope. These labels help structure discussion; they do not replace a rationale tied to user value, dependencies, risk, and learning.

Keep the two scope decisions distinct. Stage 8 — Scope creates the MVP Feature Map and Out-of-Scope List from prioritized Jobs, the product model, lifecycle, feasibility, and learning needs. Its readiness check asks whether each included capability has a clear reason and whether exclusions and dependencies are visible. This is a proposal, not the final commitment.

Stage 15 — MVP Boundary integrates and challenges the outputs of Stages 1–14: the problem and user, value hypothesis, Jobs, capabilities, experience, information architecture, data model, requirements, learning plan, technical boundaries, dependencies, and risks. It resolves conflicts where possible and records the decisions and uncertainties that remain. The Integrated MVP Specification is the combined output of the process, not an additional stage.

23. Make exclusions explicit

Every included feature consumes design, implementation, testing, and maintenance effort. Additional capabilities can also add dependencies, edge cases, and failure modes. Explicit exclusions protect the core experience from expanding into a collection of individually attractive but collectively distracting features.

For Matchshelf, the case materials describe a proposed focused MVP around the personalization concept and foundational library, onboarding, status, rating, and feedback behavior. They also identify social or community features, messaging, and other broader capabilities as exclusions or later possibilities. Treat this as documented scope direction, not proof that the product was built or that every listed capability was finalized.

24. Assemble the Integrated MVP Specification

The Integrated MVP Specification brings decisions from all 15 stages into one coherent, reviewable package. It should not be a loose collection of worksheets. It should make the reasoning and dependencies visible, preserve evidence status, and expose contradictions or unresolved choices rather than concealing them behind a polished document.

A compact specification should connect the problem, primary user, market and alternatives, value and differentiation hypotheses, product model, lifecycle, Jobs, scope, core flows, information architecture, capabilities and features, data model, requirements, acceptance criteria, learning questions, metrics, technical boundaries, dependencies, exclusions, and open questions. The exact presentation can vary, but no essential decision should be impossible to find or trace.

Preserve the status of each important statement in the integrated specification. Evidence supports a claim; a hypothesis remains to be tested; a derived insight follows from stated inputs and reasoning; a decision records a choice; a result or status records what actually happened; and an open gap names what is unresolved. Keep the source-specific notation conventions distinct. A planned test is not a result, and a decision is not proof that the underlying assumption is true.

25. The readiness gate

Readiness does not mean the product will succeed or that uncertainty has disappeared. It means the definition is clear enough for the next responsible step. Review four dimensions:

- Strategic readiness: the problem, user, value, differentiation, market context, and business model hypothesis are coherent enough to proceed.
- Product readiness: the Jobs, capabilities, core experience, information architecture, data model, and MVP boundary fit together.
- Build readiness: requirements, acceptance criteria, dependencies, technical boundaries, risks, and open questions are clear enough to estimate and implement responsibly.
- Learning readiness: critical hypotheses, evidence sources, metrics, and possible decisions are stated.

Do not let strength in one dimension conceal a weakness in another. A technically precise specification may still address an unimportant problem; an attractive opportunity may still be too ambiguous to build. The readiness decision should name blockers, accepted risks, unresolved questions, and the evidence or action needed next.

Your turn: draw the boundary

- Stage 14 — Output 1, High-Level Architecture + Output 2, Technical Constraints: Record only the material decisions needed for responsible estimation and build discussions: platform, backend/storage, authentication, integrations/AI, security, cost/latency, dependencies, fallback behavior, risks, and open questions as relevant. Avoid speculative implementation detail; distinguish decisions from assumptions.
- Stage 15 — Output 3, Boundary Statement: In a concise statement, define what the MVP will deliver, for whom, through which core journey, and what is deliberately excluded. Assemble the Integrated MVP Specification from the outputs of all 15 stages. It is the aggregate specification, not a Stage 16. Check for contradictions and broken traceability.
- Stage 15 — Output 2, Exclusions: Record each explicit exclusion and its rationale. Distinguish Later from Out, and identify evidence, dependencies, or changed conditions that could justify revisiting the decision.
- Stage 15 — Output 3, Boundary Statement: In a concise statement, define what the MVP will deliver, for whom, through which core journey, and what is deliberately excluded. Assemble the Integrated MVP Specification as the aggregate of the 15 stages—not a Stage 16—and check contradictions and broken traceability.
- Stage 15 — Final readiness review: Assess strategic, product, build, and learning readiness separately. Confirm that the core journey can be built and tested within the stated technical boundaries; record unresolved questions rather than disguising them as decisions.

Stage 14 — Technical Boundaries Ready check: The High-Level Architecture and Technical Constraints capture material build-affecting decisions, dependencies, risks, fallbacks and open questions sufficiently for responsible estimation. They do not pretend to be a complete future architecture.

Stage 15 — MVP Boundary Ready check: Final MVP Scope, Exclusions and Boundary Statement form one coherent decision. Every Must is justified and traceable; the end-to-end journey, failure states, dependencies, technical constraints and learning needs are feasible enough to assess. Stage 15 is an integration and challenge of Stage 8 scope, not a repetition of it.

OUTPUTS AND READINESS CHECKS

Stage 14 — Technical Boundaries: High-Level Architecture and Technical Constraints.
Gate: Are the material build-affecting decisions, dependencies, risks, fallback behaviors and open questions explicit enough for responsible estimation?

Stage 15 — MVP Boundary: Final MVP Scope, Exclusions and Boundary Statement.
Gate: Is the final boundary coherent, with every Must justified and traceable, the end-to-end journey feasible enough to assess, and important dependencies, technical constraints and learning needs made explicit?

Integrated specification: Assemble the outputs of all 15 stages into the Integrated MVP Specification. Record the MVP readiness decision separately, including unresolved questions. The integrated specification is not a Stage 16.

PART VII — BUILD, TEST, LEARN

The specification starts a learning cycle; it does not end one.

CORE QUESTION: How should we move from a defined MVP to evidence and better decisions?

LEARNING OBJECTIVE: Prepare a clear handoff, choose validation methods that fit the uncertainty, and turn evidence into explicit learning and iteration.

26. From specification to build

A build handoff should transfer the decisions already made without pretending that discovery is finished. Give the implementation team the current Integrated MVP Specification, the final scope boundary, testable requirements and acceptance criteria, relevant diagrams, data definitions, dependencies, assumptions, and open questions. Make clear which decisions are fixed for the current MVP and which remain provisional.

Not every open question must be resolved before implementation. Prioritize questions by consequence. An uncertainty that could change the core data model, user promise, or a critical dependency may need resolution first. A question about a deferred enhancement may safely remain open. Record who owns each material question and when it must be revisited.

Sequence the build around dependencies and learning priorities. A vertical slice that exercises the core journey may reveal more than completing each technical layer in isolation. The right sequence depends on the product's risks and constraints; the specification should make those dependencies visible without pretending to dictate every engineering decision.

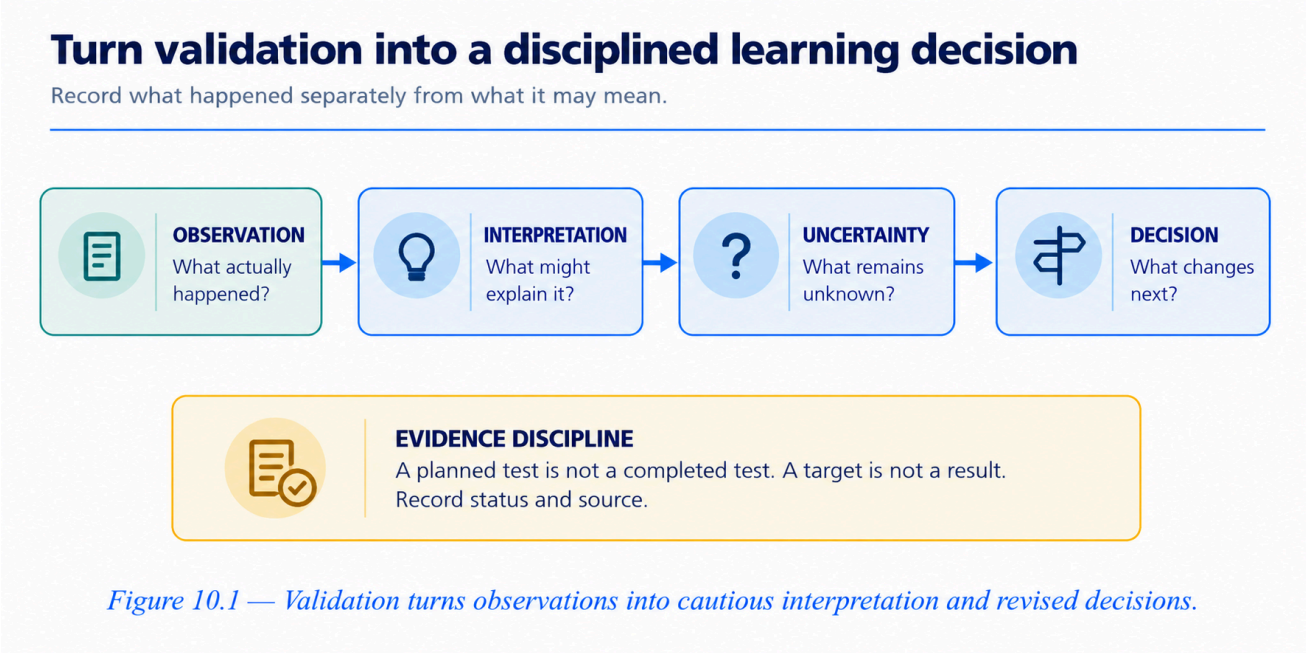


Figure 10.1 — Validation turns observations into cautious interpretation and revised decisions.

27. Validate the MVP

Choose a validation method to match the uncertainty. Interviews or contextual observation may help establish whether a problem is recognized; a prototype may expose usability issues; a working MVP may

be needed to observe behavior in context. No single method answers every question, so state what the chosen method can and cannot establish. *Figure 10.1 illustrates the relationship discussed in this section.*

A landing page, fake door, concierge approach, prototype, or working MVP can each be useful under the right conditions. None is automatically the best choice. Select the least costly method that can produce credible evidence for the decision at hand, while accounting for what it cannot reveal.

Before a test begins, define what will be observed, how it will be collected, how results will be interpreted, and what decision they could inform. Do not change the success definition after seeing the outcome simply to make the test look positive. Unexpected results are useful when they lead to a more accurate explanation or a better next question.

28. Learn and iterate

Evidence does not interpret itself. Separate observations from interpretations and decisions. A small number of users may reveal a serious usability problem without establishing market demand. Positive interview comments may reflect politeness or hypothetical intent rather than actual behavior. State the limits of each inference and seek additional evidence when the decision warrants it.

Use a learning record built around five questions: What did we expect? What did we observe? What can we reasonably infer? What remains uncertain? What decision follows? Then update the relevant hypothesis, specification, or validation plan. Preserve the history so later readers can distinguish the original expectation from what was learned.

The learning loop is Evidence → Learning → Revised Hypothesis → Product Change. It does not require adding features after every test. Learning may justify narrowing scope, changing the target user, revising the value proposition, improving an existing flow, or stopping the effort. A decision not to build can be a valid outcome of disciplined learning.

Master Case: plan the test, do not invent the result

The MyBookScout/Matchshelf case materials describe a planned sequence ending in a 50–100 reader beta intended to examine utility and retention. The available materials do not document completed build work, beta execution, observations, or post-beta decisions. Present the beta as planned, not as a completed validation or a source of results.

A useful validation plan could examine whether readers can capture enough of their library, whether they judge recommendations relevant, whether explanations help them assess fit, and whether feedback changes later recommendations in a useful way. These are proposed questions and signals, not reported findings. Define the sample, method, observation criteria, limitations, and decisions before interpreting results.

Your turn: connect uncertainty to evidence

- Write the most important unresolved hypothesis in one sentence.
- Choose a validation method that can address it and explain what the method cannot prove.
- Define the observation or evidence to collect.
- State what different results would cause you to do next.

OUTPUT AND GATE Output: Build Handoff, Validation Plan, Evidence Set, and Learning/Iteration Record. Gate: does the chosen method produce evidence relevant to the uncertainty, and is the next decision explicit?

PART VIII — AI AS COMPANION

Use AI to sharpen product thinking while keeping judgment human-owned.

CORE QUESTION: How can AI support the process without becoming the decision-maker?

LEARNING OBJECTIVE: Apply AI to clarification, challenge, synthesis, and specification while preserving evidence discipline, traceability, and human accountability.

29. AI as a product-thinking partner

AI can help when the task is well framed. It can organize unstructured notes, suggest missing questions, compare explanations, identify contradictions, draft acceptance criteria, and summarize evidence. Its fluency can also make unsupported claims sound credible. Treat its output as a proposal to inspect, not as a source of truth. Figure 11.1 illustrates the relationship discussed in this section.



Figure 11.1 — Keep verification and human accountability inside the AI-assisted workflow.

Give AI a defined role: clarify, challenge, compare, or review. Avoid asking it simply to "make the idea better," which can encourage premature solution design. A more useful instruction asks it to separate evidence from assumptions, identify broken traceability, surface alternatives, and explain what additional evidence would change the decision.

30. Use AI across the stages

Task	Useful AI role	Human responsibility
Idea	Structure the raw idea and expose ambiguity.	Confirm that the original intent is represented accurately.
Problem	Challenge the problem statement and surface alternatives.	Judge whether evidence supports the problem claim.
User	Compare candidate user groups and identify missing distinctions.	Choose the primary user and accept the trade-offs.
Value	Challenge the value proposition against alternatives.	Decide what value matters and what is sufficiently supported.
Product / lifecycle	Suggest entities, relationships, states, and transitions to review.	Approve the product model and relevant rules.
Jobs / scope	Identify missing Jobs and challenge feature creep.	Choose the Jobs and MVP boundary.
Requirements	Draft testable behavior and edge cases.	Verify that criteria match the intended product decision.
Learning	Suggest hypotheses, measures, and possible interpretations.	Choose evidence standards and make decisions from results.

Treat AI outputs as proposals. Verify factual claims against reliable sources when external information is involved. Label generated hypotheses as hypotheses. Do not accept an AI-generated citation, market estimate, user quote, or case detail without checking that it is real, relevant, and accurately represented. If it cannot be verified, remove it or mark the gap.

A practical AI workflow

A repeatable workflow is more useful than a large prompt collection. Provide the current stage question and artifact; state the role AI should play and the constraints it must respect; request a structured response that separates evidence, assumptions, reasoning, and open questions; then verify the result and record what you accept, reject, or defer.

REUSABLE PROMPT “Review this MVPCompanion artifact for the current stage. Do not invent evidence or decide on my behalf. Separate Evidence (E), Hypothesis (H), Derived Insight (D), and Open Question (O). Identify ambiguity, contradictions, missing traceability, and assumptions that affect the next decision. Suggest questions or revisions, explain why they matter, and mark all suggestions as proposals for human review.”

31. Where human judgment remains

Humans remain accountable for deciding whether the problem matters, whether the user cares, whether the product is worth building, which trade-offs are acceptable, what risks to take, and what evidence is sufficient for a decision. AI can inform those judgments, but it cannot assume accountability for their consequences.

AI can improve a decision process by surfacing alternatives and challenging reasoning. It cannot accept the consequences on behalf of the product owner. A human decision record should state the choice, rationale, evidence considered, risks accepted, alternatives rejected, and questions left open.

Keep the five lenses active

AI can review a proposal through Desirability, Differentiation, Viability, Feasibility, and Learning. Ask it to identify weak reasoning and the evidence needed to strengthen it. Do not treat a generated score as an objective measurement or substitute for user, market, technical, or business evidence.

Your turn: use AI without outsourcing the decision

- Choose one unresolved product decision and provide the current evidence and hypotheses.
- Ask AI to identify contradictions, alternatives, missing evidence, and broken traceability.
- Verify factual claims and reject unsupported suggestions.
- Record the final human decision, rationale, risks, and open questions.

OUTPUT AND GATE Output: AI-assisted Product Decision and Human Decision Record. Gate: did AI improve the quality of reasoning without taking ownership of the decision?

The MVPCompanion Working Review

Use this review before handing the specification to a development team or committing to a validation activity. It is not a substitute for the stage gates; it is a compact final check that the outputs fit together.

Traceability review

Link	Review question
Problem	Is the problem explicit and separated from the proposed solution?
User	Is a primary user identified, with relevant non-target boundaries?
Outcome	Is the intended change for that user stated clearly?
Job	Does the Job describe what the user is trying to accomplish, rather than a feature?
Capability	Does each essential capability support a Job and outcome?
Requirement	Is the required behavior observable and sufficiently unambiguous?
Acceptance Criterion	Can the behavior be checked under stated conditions?
Metric	Will the evidence inform a specific hypothesis or decision?

Evidence and reasoning review

- Evidence is distinguishable from assumptions and hypotheses.
- Derived insights identify the reasoning or inputs they depend on.
- Open questions are visible, prioritized, and assigned to a next action where appropriate.
- Decisions have a rationale and their downstream effects have been checked.
- Master Case claims do not exceed the evidence available; missing historical results are stated as missing.

Specification coherence review

- The product model, lifecycle, Jobs, experience, information architecture, data model, and requirements do not contradict one another.
- Every essential feature has a clear reason to exist.
- The MVP includes a coherent first-value journey and important failure states.
- Technical boundaries are sufficient for responsible estimation and implementation without over-designing the future.
- The scope includes explicit exclusions and a documented readiness decision.
- The learning plan identifies uncertainties, evidence, interpretation, and decisions that may follow.

The final question

Could another person understand what to build, why it matters, what is deliberately out of scope, and how the team will know whether the MVP is working as intended? If not, identify the missing decision. Do not solve the problem by adding more pages to the specification. Solve it by clarifying the right question.

THE METHOD IN ONE SENTENCE MVPCompanion progressively clarifies an uncertain product idea into a coherent, traceable, buildable and learnable MVP specification—then uses build, test, and learning to improve the next decision.

Compact Glossary

Acceptance criterion — A condition that makes a requirement's expected behavior observable and checkable.

Business Model hypothesis — A provisional explanation of how the product may create and capture value.

Capability — Something the product must be able to do to support a Job.

Derived insight / derived reasoning (D) — A conclusion drawn from stated inputs through explicit reasoning. Use the exact label required by the source context: “Derived Insight” in the 15-Stage Specification; “Derived Reasoning” in the Master Case Evidence Matrix.

Evidence (E) — Information that supports or weakens a claim; its strength depends on its source and relevance.

Hypothesis (H) — A proposition that may be true but has not been established sufficiently for the decision at hand.

Integrated MVP Specification — The aggregated, traceable output of the 15 stages; it is not an additional stage.

Job — What a user is trying to accomplish or make progress on in a particular situation.

Learning — The process of interpreting evidence to revise a hypothesis or inform a product decision.

MVP — The smallest coherent solution that can deliver intended value and support a relevant learning objective.

Open question (O) — A material issue that remains unresolved.

Progressive clarification — The method principle of answering focused product questions in sequence, with each output informing later decisions.

Requirement — A description of product behavior or constraints sufficiently clear to implement and assess.

Stage gate — A check that the current stage's question has been answered clearly enough to proceed.

Traceability — The ability to follow a product decision from problem and user outcome through Job, capability, requirement, acceptance criterion, and metric.

The Master Case remains a teaching case with documented concepts, hypotheses, planned scope and open evidence gaps. No market validation, beta completion, recommendation utility, retention improvement, final ranking algorithm, final schema, or final technical architecture is asserted without supporting source evidence.

Appendix A — The 15 Stages: Outputs and Readiness Checks

Use this register to check whether each stage has produced an output clear enough to inform the next decision. The checks do not require certainty; they require explicit reasoning, visible assumptions, and a coherent decision. The controlled 15-Stage Specification remains the source for the stage architecture. Reader-facing Job terminology follows MD-001, incorporated into the formally frozen Method Baseline v0.2-R3; historical source wording is retained only when explicitly describing the earlier Stage Specification.

Stage	Core question	Canonical output	Readiness check	Failure to avoid
01 — Idea	What could this be?	Idea Brief + Idea Hypothesis	Idea Ready: idea, context, provisional user/outcome, and major uncertainties are explicit.	Do not treat assumptions as facts or reduce to an MVP prematurely.
02 — Problem	What real problem exists?	Problem Statement + Evidence / Assumptions	Problem Ready: solution-independent problem, context, behavior, alternatives, consequences, and uncertainties are explicit.	Do not disguise the proposed solution as the problem.
03 — User	Who has the problem?	Primary User Profile + Target / Non-Target Segment	User Ready: primary user, context, constraints, current behavior, and non-targets are explicit; uncertainty remains visible.	Do not choose everyone as the target or invent a validated beachhead.
04 — Value	Why does solving it matter?	Value Proposition + Benefits + USP + Positioning; use JTBD as supporting analysis	Value Ready: outcome and value proposition are explicit without a feature list; alternatives and differentiation are visible.	Do not substitute a feature list or unsupported uniqueness claim for value.
05 — Product	What is the product fundamentally?	Product Mental Model + Entity Map	Product Ready: central object, supporting entities, and relationships are coherent.	Do not start from screens or let the model become an unbounded entity catalogue.
06 — Context & Lifecycle	In what context does value occur, and how do relevant objects/processes evolve?	Context & Lifecycle Model	Context/Lifecycle Ready: relevant contexts, states, transitions, and recurring loops are explicit enough to guide scope and behavior.	Do not model states that have no operational value; do not omit meaningful transitions.
07 — Job	What must users accomplish?	Job definition / prioritized Job set (aligned to MD-001; legacy Stage Specification uses “Core User Goals”)	Stage gate must be checked against the controlled stage source; keep the canonical traceability link as Job per MD-001, pending baseline incorporation.	JTBD is supporting analysis, not a separate stage or mandatory artifact; do not silently rewrite the frozen baseline.
08 — Scope	What is the smallest useful solution candidate?	MVP Feature Map + Out-of-Scope List	Scope Ready: every proposed MVP capability supports a Job; exclusions and dependencies are explicit.	This is the first coherent scope proposal, not the final integrated MVP boundary.
09 — Experience	How does the user reach value?	Core User Flows + Screen Inventory	Experience Ready: core flows reach first value and include essential states; screen inventory is derived from the flows.	Do not begin with a screen inventory; include empty, loading, error, and recovery states where relevant.

Stage	Core question	Canonical output	Readiness check	Failure to avoid
10 — Information Architecture	How is the product organized?	Information Architecture + Navigation Model	IA Ready: hierarchy and navigation are explicit; every MVP capability has a natural home.	Organize around the product mental model and Jobs; the case hierarchy is conceptual, not validated screen-level IA.
11 — Data Model	What information must exist?	MVP Entity/Data Model	Data Ready: essential entities, fields, relationships, ownership, and lifecycle data are explicit.	Distinguish manual/imported/computed data; remove purposeless fields and defer speculative schema.
12 — Requirements	What exactly must the system do?	Functional Requirements + Acceptance Criteria	Requirements Ready: critical behaviors and acceptance criteria are explicit enough to implement and test without repeated guessing.	Specify observable behavior and relevant inputs, outputs, validation, permissions, integration behavior, and states.
13 — Learning	What must the MVP teach us?	Success Metrics + Learning Questions	Learning Ready: core value event, key signals, learning questions, and decision-changing evidence are explicit.	Do not confuse 'the app works' with evidence of value; avoid vanity metrics and metrics disconnected from hypotheses.
14 — Technical Boundaries	What technical conditions constrain the MVP?	High-Level Architecture + Technical Constraints	Technical Ready: platform, backend, data, authentication, integrations, AI dependencies, risks, and open questions are visible.	Define enough to estimate/build; do not over-architect or invent Matchshelf-specific technical decisions.
15 — MVP Boundary	What exactly is the smallest coherent product?	Final MVP Scope + Exclusions + Boundary Statement	MVP Boundary Ready: promise, user, Jobs, capabilities, experience, IA, data, AI/integrations, learning, technical boundaries, exclusions, and open questions are coherent.	Integrate and challenge Stages 1–14. The Integrated MVP Specification aggregates stage outputs; it is not Stage 16.

Terminology note: the earlier frozen Stage Specification contains "Core User Goals" in its traceability chain and Stage 7 title. MD-001 establishes "Job" as the canonical reader-facing term and is incorporated into Method Baseline v0.2-R3. The earlier source wording is preserved for historical traceability, not as the current canonical term.

Appendix B — Evidence and Traceability Controls for the Master Case

This appendix records how to read the case responsibly. It does not add new product facts. The Master Case Evidence Matrix v0.3 is the source of record for evidence status; where the source marks an item as a hypothesis or open question, this guide must not present it as a validated result.

Case element	Evidence status	Reader-facing rule
Personal Book Intelligence Platform	Documented positioning / product hypothesis	Can be described as the documented concept or positioning. Do not imply market validation.
Reader's Paradox / Decision Fatigue	Problem framing in the product package; remains a hypothesis about user pain	Use as the case's stated problem hypothesis, not proof of prevalence or severity.
Primary user: individual reader with a developed taste and recurring next-read decision	Defined primary-user hypothesis; no validated persona/segmentation evidence	Describe as the proposed primary user; do not call the beachhead validated.
Library → Taste → Discovery → Decision → Learning	Documented conceptual product hierarchy	Label as a conceptual model, not a validated information architecture or final screen structure.
Work, Edition, ISBN, UserBook, Interaction, UserTaste, Recommendation	Objects named in the product model; final schema/relationships remain open	Treat as conceptual product entities, not an implementation-ready database schema.
Book DNA target above 70%	Target, not achieved result	Call it a target or hypothesis. Never state that coverage was achieved.
€5.99–€7.99/month or €49–€69/year	Pricing hypothesis to test	Do not present as validated pricing or willingness to pay.
Illustrative ranking formula	Illustrative formula, not final implementation	Label as an example only; do not claim production use.
50–100 reader beta	Planned validation activity in the case materials	State that the available materials do not document actual completion or observed results.
BookScout → MyBookScout → Matchshelf	Complete historical chronology not established by available sources	Use as a narrative framing only if explicitly marked as reconstruction, not verified history.
Recommendation utility / retention	Learning targets/questions, not observed outcomes	Do not imply measured improvement, retention, or validated recommendation quality.

Traceability rule

Use the canonical reader-facing chain established by MD-001 and incorporated into Method Baseline v0.2-R3: Problem → User → Outcome → Job → Capability → Requirement → Acceptance Criterion → Metric. The earlier frozen Stage Specification records "Core User Goal" in its historical chain; retain that wording only when describing that source version.

Evidence labels

For stage-level reasoning, use E = Evidence, H = Hypothesis, D = Derived Insight, O = Open Question. For the Master Case Evidence Matrix, preserve its additional distinctions C = Decision and R = Result/Status. A decision is not automatically a result, and a planned test is not evidence that the test occurred.

Worked-example verification status (Stages 8–15): the available traceability example is coherent as an illustrative single-capability chain, but does not constitute a full worked example for each stage. The manuscript therefore labels this limitation explicitly; stage-specific worked examples remain an editorial gap rather than being invented.