ERICFONDREN.COM

The Instant MVP Prompt Pack

Six prompts to take a messy idea to a scoped, validated MVP — by Eric Fondren

How to use this pack

Run these six prompts in order. You can use one long chat and paste each new prompt as you go, or start a fresh chat per step — either works, since each prompt tells you exactly what to paste in. After each step, copy the AI's full output; you'll paste it into the next prompt.

Every prompt works in ChatGPT or Claude (or any capable chat model). Just paste the prompt, then paste your inputs where marked, and send it.

By the end, you'll have: a locked idea snapshot, a scoped MVP definition with a data model and UI list, a customer research brief, a market signal check, a locked success definition, and a full messaging kit — headline, positioning, objection handling, and a pitch deck outline.

Step 1 — Intake & Idea Snapshot

What it does: takes your idea exactly as you describe it and turns it into a clean, faithful snapshot — no polishing, no "improving" it — plus a list of what's still missing. What you'll paste: your idea, in your own words. What you'll get: a short JSON snapshot of your idea and a list of gaps to fill in before you scope the MVP.

```
# Step 1 — Intake & Idea Snapshot
## Idea → MVP Compression

---

## Role
You are helping capture a founder's idea **exactly as stated**, without interpretation,
compression, or improvement.
You do not decide what the product is.
You are preparing clean input for the next step (MVP definition).

---

## Objective
Create a **faithful snapshot** of the founder's intent and identify what information is
missing to move on to defining the MVP.

---

## Inputs
You may receive input from:
- Notes from a conversation about the idea
- A rough written pitch or description
- Any other raw notes about the idea

Treat all input as raw and non-final.

---

## Rules (Non-Negotiable)

1. **No interpretation**
   Do not reframe, improve, or clarify the idea.

2. **No compression**
   Do not reduce or abstract meaning.

3. **Founder language only**
   Use the founder's own words wherever possible.

4. **Surface gaps explicitly**
```

```
   Missing or unclear information must be listed, not inferred.

5. **No decisions**
   Do not define scope, user, market, or solution.

---

## First-Principles Constraint (Limited)

You may use first-principles reasoning **only** to identify what is missing or
underspecified.

Do NOT:
- Reduce scope
- Reinterpret intent
- Improve the idea
- Propose solutions
- Decide what should exist

Your role is to faithfully compress what is stated and surface gaps only.

---

## Disallowed

Do not apply first-principles reasoning to decide necessity, minimality, or correctness.
Do not remove elements based on judgment.
Do not introduce new structure beyond what is required to describe the idea as given.

---

## Required Output Format (JSON ONLY)

Return a single JSON object. Do not include markdown, prose, or explanations.

{
  "step": "1",
  "founder_intent_verbatim": "string",
  "intake_snapshot": "string",
  "missing_info_list": ["string"]
}

---

## Field Definitions

- **founder_intent_verbatim**
  The founder's stated intent in their own words. Quote directly when possible.

- **intake_snapshot**
  A neutral summary of what the founder says they want to build, using their framing and
language.
```

```
- **missing_info_list**
  A list of concrete questions or missing details needed before moving to MVP definition
(e.g., unclear user, unclear problem, unclear context).

---

## Validation Check (Internal)
Before responding, verify:
- No interpretation or improvement was introduced
- No assumptions were added
- Missing information is listed explicitly

If these checks fail, revise before returning the JSON.

---

## Reminder

This step captures. The next step decides.

--- PASTE BELOW ---
[Your idea, in your own words — a paragraph or two is fine. Include anything else you have:
notes from conversations about it, a rough pitch, texts to a friend, whatever exists.]
```

Step 2 — MVP Definition & Scope Lock

What it does: turns your idea snapshot into the smallest coherent, build-ready MVP — what's in, what's out, the one user flow, the data model, the screens. What you'll paste: Step 1's output plus a few plain-language notes about the problem and customer. What you'll get: a locked MVP definition you can hand to a developer, an AI builder, or use to scope your own build.

```
# Step 2 — MVP Definition & Scope Lock

---

## Role
You are defining an **MVP Definition & Scope Lock**.

Your role is to translate a clarified problem and promise into the **smallest coherent, build-ready MVP** and to freeze scope.
Do not validate markets, refine customer profiles, design UX polish, or plan execution — that happens in later steps.

---

## Objective
Convert the idea snapshot and the founder's notes into a **locked MVP definition** by eliminating non-essential elements and formalizing what is included and excluded.

---

## Inputs
You are allowed to use **only** the following inputs:

- Output from Step 1 (the idea snapshot)
- The founder's notes, answering:
  - Problem Statement — what problem does this solve?
  - Customer Hypothesis (provisional) — who has this problem, as best you can tell?
  - Core Job-to-Be-Done — what job is the user trying to get done?
  - MVP Promise — what's the one outcome this delivers?
  - Draft Single Happy Path — the single most important thing a user does in this product, start to finish, as a rough draft
  - Assumptions & Risks — anything you're assuming or worried about
- Your Own Research Notes (optional — technical specs, implementation details, or pre-build research you've already done. When present, use these to inform scope decisions, feature definitions, tech stack preferences, and the single happy path. Treat as authoritative context.)

If information is missing or inconsistent, you must surface the gap explicitly. You may not invent.
```

```
---

## Rules (Non-Negotiable)

1. **Scope freezing is mandatory**
   You must explicitly define what is included and what is excluded.

2. **No feature expansion**
   You may not add features beyond what is strictly required to support the happy path.

3. **No market or customer-profile changes**
   Customer-profile refinement and market validation are forbidden in this step.

4. **Minimal sufficiency**
   When choosing between options, select the smallest coherent structure that still
delivers the MVP Promise.

5. **Prefer elimination over addition**
   If an element is not necessary, it must be excluded.

6. **Single-path enforcement**
   The MVP must support one primary user flow only.

7. **Actor & Ownership (Non-Negotiable)**
   If the happy path includes a human actor performing actions (e.g., "user uploads," "user
downloads"), you MUST explicitly choose **exactly one** of the following and reflect it in
the data model and relationships:

   - **PER_USER_OWNED**
     Include a 'User' (or 'Actor') entity and define ownership relationships to all
resources created or used in the happy path.

   - **SINGLE_ANONYMOUS**
     Do not include a 'User' entity. Explicitly state that the MVP is single-user or
anonymous and does not support per-user separation.

   **Do not omit this decision. Do not assume.**
Failure to explicitly declare the Actor & Ownership choice invalidates the output.

---

## First-Principles Constraint (Non-Negotiable)

You must define the smallest coherent product that can deliver the stated outcome.

For every component, feature, screen, or data element:
- Ask: "If this is removed, does the outcome still occur?"
- If yes, remove it.
- If no, keep it.

Assume no future phases, no iteration, and no expansion.
Only what is strictly necessary may remain.
```

```
---

## Scope Lock Rules

- Explicitly list what is excluded.
- Do not preserve optionality.
- Do not reference future versions or improvements.

---

## Required Outputs
Return the following outputs **in order**. Each output must be concise, declarative, and
treated as final for the rest of this pack.

### 1. Product One-Liner
State in one sentence:
- What the product is
- What the product is explicitly not

Avoid marketing language.

---

### 2. MVP Scope (Included)
List the **minimum set of capabilities** required to support the happy path.

Each item must map directly to a step in the happy path.

---

### 3. MVP Scope (Explicitly Excluded)
List all major elements, features, or ideas that are **not** part of the MVP.

This list must be explicit and non-trivial.

---

### 4. Single Happy Path (Final)
Rewrite the happy path as a **locked sequence of 5–8 steps**.

This replaces the founder's draft and becomes authoritative.

---

### 5. Core Data Model
Define the **minimum viable data model** required to support the happy path.

Rules:
- 3–7 entities maximum
- Each entity includes only essential fields
- All relationships required by the happy path MUST be explicitly declared
```

```
- Do not speculate beyond what the happy path requires
- If a relationship is unclear or ambiguous, surface it as a blocking ambiguity (do not
invent)

For each entity, define:
- entity name
- fields
- relationships to other entities (if any), including:
  - related entity
  - relationship type (one-to-one, one-to-many, many-to-many)
  - purpose of the relationship (why it exists for the happy path)

---

### 6. UI Surface List
List the **minimum set of screens or surfaces** required to complete the happy path end-to-
end.

No design detail. No layout guidance.

---

### 7. Action Map
For each step in the happy path, state:
- User action
- System response or transformation

Keep this conceptual, not technical.

---

### 8. MVP Success Definition
Define what "working" means for this MVP in **learning or functional terms**, not growth.

Example framing (do not copy verbatim):
- A user can complete the happy path without assistance.

---

### 9. Tech Stack Preferences (OPTIONAL)

Capture non-negotiable constraints or deviations from common defaults.

**Common default stack (assumed if omitted):**
- Frontend: React
- Backend: Express (Node.js)
- Database: Postgres

If you have a preference, specify:
- **backend**: EXPRESS (default) or OTHER
- **frontend**: REACT (default) or OTHER
- **db**: POSTGRES (default) or OTHER
```

```
- **constraints**: Additional technology constraints (optional array)

If omitted, these defaults apply when it's time to build.

---

### 10. Tech & Refactor Contracts (OPTIONAL BUT BINDING)

If you already know you'll need to swap out a piece of the stack later (e.g., moving from a quick database to a production one, or from a stub AI model to a real one), define that here as a non-negotiable requirement for the build.

Each constraint must be written as a requirement that can later be marked SATISFIED, NOT_SATISFIED, or DEFERRED during implementation.

Do not describe how to implement these constraints.
Only state what must be true.

For each contract, specify:
- **id**: A unique identifier (e.g., "TC-1")
- **category**: One of AI_MODEL, DATABASE, AUTH, STORAGE, PAYMENTS, INFRA, or OTHER
- **current_stack**: The technology used in the MVP (name, provider, optional notes)
- **target_stack**: The intended future technology (name, provider, optional notes)
- **timing**: MVP_ARCH_READY (must be architected for swap in the MVP) or POST_MVP_IMPLEMENT (swap happens after the MVP)
- **swap_driver**: Why the refactor will happen
- **hard_requirements**: What must be true for the swap to work (interface boundaries, data retention, etc.)
- **migration_acceptance**: What must remain unchanged after the swap
- **enforcement**: BLOCK (default, blocks the build if not satisfied) or WARN (warning only)

---

## Output Constraints

- No technical stack, tooling, or architecture references EXCEPT for tech_stack_preferences and tech_and_refactor_contracts fields
- No future versions or roadmap
- No optimization language
- No optionality or branching
- Return JSON only. No prose. No markdown.
- Output must match the schema keys exactly. Use empty arrays or nulls where allowed.
- All strings must be plain text (no bullet symbols).

OUTPUT FORMAT (JSON ONLY):

Return a single JSON object that matches this schema exactly.
Do not include markdown, code blocks, or explanatory text.

{
  "step": "2",
```

```
  "product_one_liner": "string",
  "mvp_scope_included": ["string"],
  "mvp_scope_excluded": ["string"],
  "final_single_happy_path": ["string"],
  "actor_model": {
    "type": "PER_USER_OWNED | SINGLE_ANONYMOUS",
    "ownership_rules": ["string"]
  },
  "core_data_model": [
    {
      "entity": "string",
      "fields": ["string"],
      "relationships": [
        {
          "related_entity": "string",
          "cardinality": "one-to-one | one-to-many | many-to-many",
          "purpose": "string"
        }
      ]
    }
  ],
  "ui_surface_list": ["string"],
  "action_map": [
    {
      "step": "string",
      "user_action": "string",
      "system_response": "string"
    }
  ],
  "mvp_success_definition": ["string"],
  "tech_stack_preferences": {
    "backend": "string (optional, defaults to 'Express')",
    "frontend": "string (optional, defaults to 'React')",
    "db": "string (optional, defaults to 'Postgres')",
    "constraints": ["string (optional)"]
  },
  "tech_and_refactor_contracts": [
    {
      "id": "TC-1",
      "category": "AI_MODEL | DATABASE | AUTH | STORAGE | PAYMENTS | INFRA | OTHER",
      "current_stack": { "name": "string", "provider": "string (optional)", "notes":
"string (optional)" },
      "target_stack": { "name": "string", "provider": "string (optional)", "notes": "string
(optional)" },
      "timing": "MVP_ARCH_READY | POST_MVP_IMPLEMENT",
      "swap_driver": "string",
      "hard_requirements": ["string"],
      "migration_acceptance": ["string"],
      "enforcement": "BLOCK | WARN (optional, defaults to BLOCK)"
    }
  ]
}
```

```
Outputs must read as **final decisions**, not suggestions.

---

## Validation Check (Internal)
Before returning your answer, verify:
- All outputs map directly to the MVP Promise
- Every included element supports the happy path
- All non-essential elements are explicitly excluded
- Scope is frozen and unambiguous

If any check fails, revise before responding.

---

## Reminder

Your job is not to design a good product.

Your job is to define the **smallest product that must exist** for the promise to be true.

--- PASTE BELOW ---
[Output from Step 1]
[Your notes: What problem does this solve? Who has this problem, as best you can tell? What's the one job the user is trying to get done? What's the single most important thing a user does in your product, from start to finish? Any assumptions or risks you're already aware of? Any tech preferences or constraints, if you have them.]
```

Step 3 — Customer Research Brief

What it does: pre-fills a structured research brief from your idea snapshot and MVP definition, so you know exactly what to go find out about your customer — and what you don't know yet. What you'll paste: Step 1 and Step 2's outputs. What you'll get: a brief you can use to guide your own digging through reviews, forums, and conversations with real prospective customers.

```
# Step 3 — Customer Research Brief

You are generating a **Research Brief** for customer profiling by extracting and
synthesizing information from the idea snapshot and MVP definition produced in Steps 1 and
2.

## Inputs
You will receive:
- Output from Step 1 (idea snapshot)
- Output from Step 2 (MVP definition & scope lock)

## Task
Generate a research brief that pre-populates as much as possible from those two outputs.
For fields where information is not available, provide a reasonable placeholder or leave
empty.

## Required Output (JSON ONLY)

{
  "step": "3",
  "offering_description": "What exactly is being delivered - extracted from the Step 2
product_one_liner and MVP scope",
  "delivery_model": "How it's delivered - inferred from Step 2 scope and happy path",
  "timeframe": "How long value delivery takes - inferred from the MVP promise if
mentioned",
  "pricing_range": "Unknown - fill this in yourself if you already have a sense of it,
otherwise leave as unknown",
  "hypothesis_who_might_buy": "Demographics from the Step 2 customer hypothesis",
  "hypothesis_current_situation": "Context from the problem statement",
  "hypothesis_problem_solving": "From the core job-to-be-done and MVP promise",
  "hypothesis_differentiation": "Inferred from Step 2's scope exclusions and unique
approach",
  "exclusions_who_not_for": ["From Step 2's scope_excluded and any stated assumptions"],
  "exclusions_what_problem_not_solved": "From Step 2's mvp_scope_excluded",
  "evidence_interest": "Unknown - fill this in yourself once you've gathered evidence",
  "evidence_competitive": "Unknown - fill this in yourself once you've gathered evidence",
  "evidence_observations": "From the Step 1 founder_intent_verbatim if it mentions
observations",
  "research_geographic_focus": "Global (change if your product is local or regional)",
  "research_depth": "Standard depth to aim for on your own: roughly 40-60 data points
```

```
across 6–8 platforms (forums, review sites, social media, communities where your customer
hangs out)"
}

## Rules
- Extract verbatim where possible
- Mark unknown fields with "Unknown — fill in yourself"
- Do not invent information not present in the inputs
- Return JSON only, no markdown

## What to do with this brief
This brief is a map for research you do yourself: read reviews of competing products,
search relevant subreddits and forums, and if you can, talk to a handful of people who
might be your customer. Use the brief's questions to know what to look and listen for.
Bring whatever quotes, reviews, forum posts, or notes you collect into Step 4.

--- PASTE BELOW ---
[Output from Step 1]
[Output from Step 2]
```

Step 4 — Market Signal Check

What it does: grounds your locked MVP in real-world evidence — competitors, substitutes, demand signals, pricing anchors, and risks — without changing your product or scope. What you'll paste: Step 1 and Step 2's outputs, plus any customer research you've gathered. What you'll get: a plain, evidence-based read on whether this problem is real and costly enough to matter.

```
# Step 4 — Market Signal Check

---

## Role
You are doing a **Market Signal Check**.

Your role is to ground a locked MVP in **observable reality signals** without changing the product, scope, or customer definition.
Do not validate markets, forecast success, or plan go-to-market strategy — that's not this step.

---

## Objective
Identify and compress the **minimum set of real-world signals** that indicate whether the MVP addresses a costly, urgent, or persistent problem — without slowing momentum.

---

## Inputs
You are allowed to use **only** the following inputs:

- Output from Step 1 (idea snapshot)
- Output from Step 2:
  - Product One-Liner
  - MVP Scope (Included / Excluded)
  - Final Single Happy Path
- Any customer quotes, reviews, forum posts, or research notes you've collected — from Step 3's brief, or your own digging. If you haven't done that research yet, say so plainly instead of guessing.

Do not infer customer characteristics beyond what's in the evidence you provide.
If you have no real customer evidence yet, flag market signal reliability as HIGH RISK.

If evidence is weak or missing, you must surface that explicitly. You may not infer or invent.

---
```

```
## Rules (Non-Negotiable)

1. **Signals over speculation**
   Only include signals that imply real cost, effort, urgency, or persistence.

2. **No product changes**
   You may not suggest feature additions, scope changes, or redesigns.

3. **No market sizing or forecasts**
   TAM/SAM/SOM, growth projections, and probability estimates are forbidden.

4. **No go-to-market planning**
   Channels, campaigns, or acquisition tactics are out of scope.

5. **Compression over exhaustiveness**
   Prefer the smallest set of signals that materially ground the idea.

6. **Evidence-quality awareness**
   If you have little or no direct customer evidence, all market signals in the output must
be flagged as potentially unreliable.

---

## Required Outputs
Return the following outputs **in order**. Each output must be concrete, concise, and
directly tied to the customer and MVP promise.

### 1. Category & Existing Alternatives
Describe what users currently choose instead of this MVP.

Include:
- Direct alternatives
- Adjacent tools or substitutes
- Manual or DIY approaches

---

### 2. Direct Competitors (If Any)
List existing products that directly attempt to solve the same problem for the same
customer.

If none exist, state "None identified."

---

### 3. Adjacent Substitutes
List tools or behaviors that partially solve the problem or reduce its impact.

---

### 4. Demand Signals
List concrete signals that indicate the problem is real and recurring.
```

```
Examples of acceptable signals:
- Repeated complaints
- Time-consuming manual processes
- Ongoing spend on workarounds

Avoid vague interest indicators.

---

### 5. Differentiation Wedge
State in one sentence **why this MVP is meaningfully different** from existing
alternatives.

Do not claim superiority. State contrast.

---

### 6. Pricing Anchors
Identify reference points that suggest what the customer currently pays or sacrifices.

This may include:
- Subscription costs
- Time cost
- Opportunity cost

---

### 7. Key Risks
List the most material risks that could stall or invalidate the MVP.

Do not mitigate or solve them.

---

## Output Constraints

- No feature suggestions
- No optimism or persuasion
- No future roadmap references
- Return JSON only. No prose. No markdown.
- Output must match the schema keys exactly. Use empty arrays or nulls where allowed.
- All strings must be plain text (no bullet symbols).

OUTPUT FORMAT (JSON ONLY):

Return a single JSON object that matches this schema exactly.
Do not include markdown, code blocks, or explanatory text.

{
  "step": "4",
  "category_and_alternatives": ["string"],
```

```
  "direct_competitors": ["string"],
  "adjacent_substitutes": ["string"],
  "demand_signals": ["string"],
  "differentiation_wedge": "string",
  "pricing_anchors": ["string"],
  "key_risks": ["string"]
}

Outputs must read as **observations**, not arguments.

---

## Validation Check (Internal)
Before returning your answer, verify:
- All signals are tied to real behavior or cost
- No product or scope changes appear
- The signal set is minimal but sufficient

If any check fails, revise before responding.

---

## Reminder

Your job is not to prove this will work.

Your job is to **anchor the idea in reality without slowing it down**.

--- PASTE BELOW ---
[Output from Step 1]
[Output from Step 2]
[Any customer quotes, reviews, forum posts, or research notes you've collected — from Step
3's brief or your own digging. If you haven't done that research yet, say so.]
```

Step 5 — Success Criteria Lock

What it does: checks that the scoped MVP still matches your original intent, and locks a short, concrete definition of what "success" looks like in the first 6-12 weeks. What you'll paste: Step 1, Step 2, and Step 4's outputs, plus your notes on what you learned about the customer, if any. What you'll get: a locked success definition you can hold yourself to once you build.

```
# Step 5 — Success Criteria Lock

---

## Role
You are locking **Intent & Success** for this MVP.

Your role is to confirm that the scoped MVP still faithfully serves the founder's original
intent, and to lock a short, concrete definition of success for the MVP window.
Do not change product scope, customer definition, market framing, or execution plans —
that's already decided.

---

## Objective
Verify alignment between the founder's original intent and the final MVP, and define a
**bounded, learning-oriented success definition** for the MVP period.

---

## Inputs
You are allowed to use **only** the following inputs:

- Output from Step 1:
  - Founder's original intent (verbatim)
- Output from Step 2:
  - Product One-Liner
  - MVP Scope (Included / Excluded)
  - Final Single Happy Path
- Output from Step 4:
  - Key Risks
- Your own notes on what you learned about the customer, if you did the research from Step
3 (or a plain statement that you haven't yet)

If you have real customer evidence indicating the customer isn't who you originally
thought, you must acknowledge that risk in the alignment assessment below rather than
claiming full alignment.

If the original intent is missing or unclear, surface the gap explicitly. Do not infer or
reinterpret.
```

```
---

## Rules (Non-Negotiable)

1. **Intent preservation is primary**
   The founder's original intent is the source of truth. The MVP must be evaluated against it.

2. **No scope or direction changes**
   You may not add, remove, or adjust MVP scope, features, or positioning.

3. **No coaching or judgment**
   Do not advise the founder on what they should want or value.

4. **Bounded success only**
   Success must be defined for a short MVP window (6–12 weeks) and framed around learning or functionality, not business outcomes.

5. **Explicit non-requirements**
   Clearly state what the MVP will not require from the founder.

6. **Evidence-quality respect**
   If your customer evidence is thin or contradicts your original assumption, this must be acknowledged in the alignment assessment. You may not claim full alignment if the customer picture is shaky.

---

## Required Outputs
Return the following outputs **in order**. Each output must be declarative and treated as final.

### 1. Original Intent (Verbatim)
Quote the founder's original intent exactly as provided.

---

### 2. Final MVP Intent
Write one sentence describing what the MVP is intended to accomplish.

This must be clearly traceable to the original intent.

---

### 3. Alignment Assessment
State plainly whether the final MVP intent aligns with the original intent.

If alignment is imperfect, describe the nature of the drift without proposing changes.

---
```

```
### 4. Explicit Non-Requirements
List what the MVP explicitly does **not** require from the founder.

Examples include:
- Specific growth tactics
- Ongoing manual effort
- Certain behaviors or commitments

Only include items implied by the scope.

---

### 5. MVP Success Definition
Define 2–5 concrete conditions that indicate the MVP is "working" within a 6–12 week
window.

These must be:
- Learning-oriented or functional
- Directly tied to the happy path
- Independent of growth, revenue, or scale

---

## Output Constraints

- No motivational language
- No future roadmap or iteration discussion
- No optimization or strategy advice
- Return JSON only. No prose. No markdown.
- Output must match the schema keys exactly. Use empty arrays or nulls where allowed.
- All strings must be plain text (no bullet symbols).

OUTPUT FORMAT (JSON ONLY):

Return a single JSON object that matches this schema exactly.
Do not include markdown, code blocks, or explanatory text.

{
  "step": "5",
  "original_intent_verbatim": "string",
  "final_mvp_intent": "string",
  "alignment_assessment": "string",
  "explicit_non_requirements": ["string"],
  "mvp_success_definition_windowed": ["string"]
}

Outputs must read as **final confirmations**, not suggestions.

---

## Validation Check (Internal)
Before returning your answer, verify:
```

```
- Original intent is preserved verbatim
- Final MVP intent is traceable and constrained
- Success definition is bounded and measurable without growth metrics

If any check fails, revise before responding.

---

## Reminder

Your job is not to inspire.

Your job is to **lock alignment and stop drift**.

--- PASTE BELOW ---
[Output from Step 1]
[Output from Step 2]
[Output from Step 4]
[Your notes on what you learned about the customer, if you did the research from Step 3 —
or a plain statement that you haven't done it yet]
```

Step 6 — Messaging Kit

What it does: turns your locked MVP, customer picture, and market context into plain, minimal language — a value promise, objection handling, positioning angles, a homepage spine, and a pitch deck outline. What you'll paste: Step 2 and Step 4's outputs, plus any customer quotes or language notes you have. What you'll get: a messaging kit you can use to explain your product to anyone without being in the room.

```
# Step 6 — Messaging Kit

---

## Role
You are compressing a locked MVP, customer profile, and market context into **clear,
minimal language**.

Your role is to translate them into language that explains what the product is, who it is
for, and when it matters.
Do not market, persuade, hype, or optimize for conversion.

---

## Objective
Produce a **compressed messaging kit** that allows the MVP to be understood, explained, and
onboarded without you in the room to explain it.

---

## Inputs
You are allowed to use **only** the following inputs:

- Output from Step 1:
  - Problem Statement (from the idea snapshot)
- Output from Step 2:
  - Product One-Liner
  - Final Single Happy Path
- Any customer quotes, exact phrases, or language notes you have — from Step 3's brief or
your own research (what people actually say about the problem, in their own words)
- Output from Step 4:
  - Differentiation Wedge

Do not invent language.
All copy must be traceable to real customer quotes or language you've provided, not
invented phrasing.

If any required input is missing, surface the gap explicitly. Do not infer or invent.

---
```

```
## Rules (Non-Negotiable)

1. **Explanation over persuasion**
   Messaging must explain what the product does and when it is relevant. It must not persuade, hype, or convince.

2. **Customer language only**
   Use the language from the customer quotes and notes you provided. Avoid founder jargon and technical phrasing.

3. **No scope expansion**
   Messaging must reflect the locked MVP only. Do not imply future features or versions.

4. **Clarity over cleverness**
   Prefer plain, direct language over slogans or metaphors.

5. **Explicit exclusion**
   Messaging must clearly state who the product is not for.

6. **Traceable language**
   All messaging copy must reference exact phrases from the customer quotes and notes you provided. Do not invent new language. If you didn't provide any customer language, say so and use plain description instead of invented "voice of customer" phrasing.

---

## Required Outputs
Return the following outputs **in order**. Each output must be concise and directly usable.

### 1. One-Sentence Value Promise
State clearly what outcome the MVP enables and for whom.

---

### 2. Top 5 Pains (Customer Language)
Restate the five most relevant pains the customer experiences that the MVP addresses.

---

### 3. Objections & Counters
List the top objections a customer would raise and a concise counter grounded in the current MVP.

Do not reference future capabilities.

---

### 4. Positioning Angles
Provide three distinct ways to explain the MVP:
- Functional (what it helps do)
- Emotional (what frustration it removes)
```

```
- Identity-based (what kind of person it supports)

Each angle should be one sentence.

---

### 5. Homepage Spine
Provide:
- Headline
- Sub-headline
- Three supporting bullets

This must explain the MVP without marketing language.

---

### 6. Plain-English Onboarding Explanation
In no more than 120 words, explain:
- What the product is
- What a user does first
- What happens next
- What result they get

---

### 7. Anti-Positioning
List at least three clear statements describing who this product is not for or when it is
not appropriate.

---

## Output Constraints

- No calls to action
- No urgency or scarcity language
- No promises beyond the MVP scope
- No references to growth, scale, or optimization
- Return JSON only. No prose. No markdown.
- Output must match the schema keys exactly. Use empty arrays or nulls where allowed.
- All strings must be plain text (no bullet symbols).

OUTPUT FORMAT (JSON ONLY):

Return a single JSON object that matches this schema exactly.
Do not include markdown, code blocks, or explanatory text.

{
  "step": "6",
  "one_sentence_value_promise": "string",
  "top_5_pains": ["string"],
  "objections_and_counters": [
    { "objection": "string", "counter": "string" }
```

```
  ],
  "positioning_angles": {
    "functional": "string",
    "emotional": "string",
    "identity": "string"
  },
  "homepage_spine": {
    "headline": "string",
    "subheadline": "string",
    "bullets": ["string"]
  },
  "onboarding_explanation_plain": "string",
  "anti_positioning": ["string"],
  "pitch_deck_spine": [
    {
      "slide_title": "string",
      "key_points": ["string", "string"],
      "supporting_data": "string (cite evidence from Steps 3–5)"
    }
  ]
}

### Pitch Deck Spine
Generate 6–8 slides covering: Problem, Solution, Customer / Market, Differentiation,
Business Model, Traction Plan, Ask / Next Steps.
Each slide must include:
- `slide_title`: concise slide heading
- `key_points`: 2–3 bullet points for the slide body
- `supporting_data`: one sentence citing specific evidence from the customer research (Step
3), market signals (Step 4), or success lock (Step 5)

The pitch deck spine is a structural outline, not a designed deck. You'll use it to produce
an actual slide deck separately.

Outputs must read as **explanatory facts**, not marketing copy.

---

## Validation Check (Internal)
Before returning your answer, verify:
- Messaging reflects only the locked MVP
- Language matches the customer's actual vocabulary, where you provided it
- No persuasive or hype language appears

If any check fails, revise before responding.

---

## Reminder

Your job is not to sell.
```

```
Your job is to **make the MVP understandable without explanation**.

--- PASTE BELOW ---
[Output from Step 2]
[Output from Step 4]
[Any customer quotes or exact language you have — from Step 3's brief or your own research.
If you don't have any yet, say so.]
```

That's the pack. Six prompts, one direction: a messy idea in, a scoped, validated MVP and a way to explain it out.