How to Vibe-Code a Profitable App

A guide to both the business and the app implementation

11 min read · Just now

Gunnar Östberg

Context

With the rise of vibe coding, where AI and natural-language prompts are used to generate working applications, a long-standing bottleneck has been removed. Programming skill is no longer the primary limiting factor for turning an idea into something that actually runs.

This changes where the real work lies.

When implementation becomes easier, the difficulty does not disappear. It moves. It moves upstream, toward problem selection, needs analysis, and deliberate design decisions. It also moves downstream, toward marketing, positioning, sales, and operations.

In other words, coding is no longer where most projects fail. Judgment is.

Quality, therefore, matters more, not less. Almost anyone can now generate something that kind of works. Very few can build something that works reliably, explains itself clearly, and feels intentional enough that people are willing to pay for it.

The door is open. You should absolutely walk through it. Use AI wherever it is relevant and helpful. Be curious and inquisitive. But throughout all of this, focus on quality. And above all, start in the right order.

Method

What many people do, especially when excited by new AI tools, is skip needs analysis, requirement definition, architectural thinking, and business validation, and jump straight into building the app.

That order is wrong.

To use vibe coding to build a revenue-generating, ultimately profitable app, you need to follow a sequence of deliberate steps. Each step transforms the idea into a more defined and more valuable state. Each step is guarded by a gate that asks a concrete question. If the answer is no, you stop, pivot, or discard the idea before investing more time and energy.

The early gates are intentionally analytical. Their purpose is not to slow you down, but to prevent you from wasting months building and marketing ideas that should have been killed much earlier.

The gate sequence

1. **Problem selection (founder–problem fit)**
   Gate: Have you selected a real, valuable problem, and are you personally motivated and well-positioned to pursue it?

2. **Quality of problem understanding**
   Gate: For your defined ideal customer profile, is your understanding of the customer's practical, internal, and external needs deep enough that your solution choices are deliberate rather than accidental?

3. **Desk validation**
   Gate: Is the app business idea feasible, differentiated, and economically plausible?

4. **Market validation (demand validation)**
   Gate: Do real customers demonstrate a willingness to pay, with evidence appropriate to the product and market?

5. **MVP build**
   Gate: Does the product deliver real value in real use?

6. **Go-to-market scaling**
   Gate: Can you acquire customers in a repeatable way?

7. **Product scaling**
   Gate: Do users stay, and does the product improve over time?

8. **Re-architecture**
   Gate: Can the system sustain growth without collapsing under its own weight?

9. **Process management**
   Gate: Can sales, support, and delivery run without heroics?

10. **Product management**
    Gate: Is continuous improvement built into the business structure?

Step 1. Problem Selection

Finding and defining a good business idea is not mysterious, but it does require effort and honesty. A viable app business is always based on a real problem

experienced by real people.

The first step, therefore, is not about features or technology. It is about identifying a genuine need, understanding who experiences it, and deciding whether it is a problem you are willing and able to work on over the long term.

A useful signal is existing competition. If no one is attempting to solve a problem, it is often because the problem is not valuable enough. Competition does not invalidate an idea. On the contrary, it often confirms that demand exists. Differentiation comes later.

The strongest starting point is when you experience the problem yourself and understand the domain firsthand. That gives you insight, empathy, and motivation, all of which matter far more than clever ideas.

The extraction mindset

Do not begin by inventing ideas and then trying to convince the world they are important. Instead, extract existing demand.

Look for explicit signals in forums, communities, reviews, and social media. People frequently say things like “I wish there was an app for…” or “Why isn’t there a tool that…”. These are not idle complaints. They are expressions of unmet demand.

You can also study what investors and accelerators publicly seek. These lists are the result of prior market research and are another form of demand extraction.

AI is particularly useful here. It can scan hundreds of discussions, reviews, or complaint threads and surface recurring patterns far faster than manual reading.

Founder–problem fit

Even if a problem is real, you should not pursue it unless you are genuinely motivated to do so.

Ask yourself, honestly, whether you can imagine working on this problem for years rather than weeks. Ask whether you have the time, energy, and persistence required. Ask whether you can clearly envision an app-based solution, and whether you have some insight, experience, or access that others lack.

The distance between “I have an idea” and “I have a profitable business” is measured in sustained effort, not inspiration.

B2C and B2B problem discovery

For business-to-consumer ideas, scale matters. The economics usually require large numbers of paying users. Use AI-assisted search to identify many potential problems, then evaluate them based on their severity and the number of people affected.

For business-to-business ideas, problems are rarely discussed openly. They must be uncovered through conversations. That means interviewing decision-makers, running structured discovery meetings, or facilitating workshops where subject-matter experts can surface and prioritize real issues. AI can help you design interview guides and synthesize notes, but it cannot replace access to people who actually make decisions.

Choosing one idea

Once you have several candidates, choose one. Only one.

This feels risky because it seems safer to keep options open. In practice, spreading your attention across multiple ideas prevents depth. You cannot validate several ideas with the same rigor you can apply to one.

Choose the idea that combines real pain, a sufficiently large market, your motivation, and your unique position to solve it.

Gate 1

At this point, ask yourself plainly whether you have selected a real and valuable problem, and whether you are personally motivated and well-positioned to pursue it. If the answer is yes, continue. If not, stop and start again.

Step 2. Problem Understanding

This step is about the quality of your understanding. If you do not understand the problem deeply, you will build the wrong solution, even if you build it quickly.

The output of this step is clarity. Clarity about who the product is for, what problem it solves, and what conditions must be true for the solution to work.

Defining the ICP

Begin by defining your ideal customer profile. This is not “everyone with the problem”. It is the specific group that feels the pain most strongly, has the ability

and willingness to pay, and is reachable through channels you can realistically access.

A narrow ICP simplifies everything. It sharpens marketing, clarifies feature decisions, and accelerates learning. You can always expand later.

Describe your ICP concretely. For consumers, this includes behaviors, habits, and where they spend time. For businesses, it includes company size, industry, roles involved, and buying cycles.

Modeling the user's process

If the solution is obvious, state it. If it is not, you need to model the user's current process.

A proper needs analysis focuses on the user's reality, not on the app. Map the current process step by step. Identify where information is used, where decisions are made, and where friction occurs. This is the "as-is" model.

Then design the improved process, the "to-be" model, where your app removes friction and adds value. The gap between these two models defines what you actually need to build.

Also consider stakeholders, goals, constraints, and the domain vocabulary. At this stage, no technology is involved. AI can be extremely helpful for structuring and documenting these models.

Understanding the full problem

Solving the problem means more than addressing a surface need. It also means understanding how the problem feels to the user and why it matters to them.

In practical terms, this often involves three levels. There is the external problem, which is the practical task the user struggles with. There is an internal problem, which concerns frustration, anxiety, or uncertainty. And there is a deeper philosophical dimension that relates to values and identity.

You do not need to use this language explicitly in the product, but you should understand it. When you do, your later marketing becomes precise rather than generic.

Data and feasibility

No solution is viable if its data requirements are impossible.

Ask where the data comes from, whether it is legally accessible, whether it can be obtained reliably and at acceptable cost, and what happens if the source changes or disappears. An idea that depends on fragile or inaccessible data is not feasible, regardless of how attractive it sounds.

PESTLE check

A brief feasibility scan across political, economic, social, technological, legal, and environmental dimensions helps surface risks early. This is not an academic exercise. It is a way to identify deal-breakers before they become expensive mistakes.

The PRD

Document what you have learned in a product requirements document. This is not a detailed technical specification. It is a thinking tool that captures the problem, the ICP, the solution concept, user journeys, success criteria, constraints, assumptions, and open questions.

A well-written PRD becomes valuable context for AI coding later. The clearer your thinking, the better the output you get from AI tools.

Gate 2

Now ask whether your understanding of the problem is deep enough that your solution choices are deliberate rather than accidental. If the answer is yes, proceed. If not, deepen your analysis.

Step 3. Desk Validation

Desk validation is about plausibility. Before investing in development, you want to know whether the idea could work as a business.

Translate the PRD into a concrete product concept. Define what is included in the first version and, just as importantly, what is excluded. Discipline here means subtraction.

Study competitors and alternatives, including manual workarounds and the option of doing nothing. Understand pricing, positioning, and customer complaints.

Estimate market size realistically. Large theoretical markets mean little if you cannot reach them.

Set pricing based on customer value rather than your costs. Then estimate unit economics, including customer acquisition cost and lifetime value. These numbers will be imperfect, but writing them down forces clarity.

Finally, formulate a simple positioning statement that immediately clarifies who the product is for, what it does, and why it matters.

A simple kill sheet

At this stage, you should stop if you cannot identify a clear differentiator, if the data dependency is fragile, if the economics do not work even under optimistic assumptions, or if the scope cannot be reduced without destroying value.

Gate 3

If the idea is feasible, differentiated, and economically plausible, continue. Otherwise, stop.

Step 4. Market Validation

Market validation asks a single question: Are real customers willing to pay?

The form of evidence depends on context. For B2B, this often means paid pilots, deposits, or letters of intent. For B2C, it may involve conversions, pre-orders, or early subscriptions, sometimes supported by a thin but usable prototype.

The important thing is evidence, not enthusiasm.

Create a simple sales or landing page that communicates the problem, the solution, and a clear promise. Be honest about what exists and what does not. Drive relevant traffic and observe behavior.

If people sign up, talk to them. Listen more than you speak. Ask about their situation, what they have tried before, and what would happen if they did nothing. Then ask directly whether they are willing to move forward.

Low response is also data. Pivot quickly.

Gate 4

If real customers demonstrate a willingness to pay, you have earned the right to build.

Step 5. MVP Build

Now you build a real product. Not a proof of concept or a mockup, but something that delivers value in real-world use.

AI will accelerate development, but it does not replace responsibility. You must act as product owner and decision-maker. You decide what to build, how it should work, and what quality is acceptable.

Specify requirements clearly. Everything you leave unspecified will be decided implicitly by the AI. Design a simple architecture using well-documented, mainstream technologies. Build incrementally. Test continuously. Use version control from the start.

Deployment and basic operational setup should be simple but real. Monitoring, backups, and security are not optional.

Gate 5

If the product delivers real value in real use, you have a product.

Step 6. Go-to-Market Scaling

With a working product and paying customers, the next question is whether customer acquisition can be repeated.

Analyze where your first customers came from and which messages resonated with them. Focus on channels that work. Document your sales process. Track the metrics that matter, including conversion rates, acquisition costs, retention, and lifetime value.

Scale cautiously. Prove things manually before automating them.

Gate 6

If customer acquisition is repeatable, you have a business.

Step 7. Product Scaling

Product scaling is about improving and extending the product based on real usage.

This requires systematic learning. That means deliberately testing hypotheses, analyzing how users actually behave, complementing quantitative data with qualitative feedback, and paying attention to changes in the competitive landscape. Even if you work alone, these learnings should be made explicit and documented so that decisions are based on insight rather than memory.

Prioritize improvements based on impact rather than novelty. Work in short cycles of building, measuring, and learning. Retention matters more than acquisition. A small improvement in retention often has a larger impact on long-term value than a comparable improvement in acquisition.

Gate 7

If users stay and the product improves over time, you have sustained product–market fit.

Step 8. Re-Architecture

As usage grows, earlier technical choices may become limiting. Performance issues, increasing complexity, or rising costs may signal the need for re-architecture.

Approach this carefully. Avoid large rewrites. Identify specific constraints, design a target state, and migrate incrementally. AI is helpful here, but experienced engineers may be necessary when systems become complex.

Gate 8

If the system can sustain growth, move on to operations.

Step 9. Process Management

A business that relies on heroic effort does not scale. Process management is about making sales, support, delivery, and administration systematic.

Document how things work. Identify repetitive tasks and automate them where possible. Build systems that do not depend on any single person's memory or energy.

Gate 9

If the business can operate without heroics, continuous improvement becomes possible.

Step 10. Product Management

Product management ensures that improvement is deliberate rather than accidental. Products have lifecycles, and strategies must adapt accordingly.

Maintain a roadmap to provide direction, not promises. Establish regular review rhythms to surface issues early. Build learning into how you work, through experiments, data analysis, and user research.

Gate 10

If continuous improvement is built into the structure, you have a sustainable product business.

Conclusion

Building a profitable app with Vibe Coding is not primarily about coding. AI has removed much of the implementation bottleneck. What remains is judgment.

Most app ideas fail not because they cannot be built, but because they should not have been built. No one wanted them enough to pay for them. The economics did not work. The founder lost motivation. The data dependency was fragile.

The gates exist to enforce discipline. By working through them in order, you kill bad ideas early, validate demand before building, focus effort where it matters, and create systems that scale.

AI can handle syntax. Strategy, judgment, and persistence remain human responsibilities. Build something people want enough to pay for. And do it in the

right order.

If you want the full version with more detailed explanations and practical guidance, read the long-form guide here.

Vibe Coding | Product Management | Product Development | Software Development | Software

Edit profile

Written by Gunnar Östberg

196 followers · 143 following

CEO Business Clarity. I write enjoyable, rewarding articles that retain their value. Please check my profile to see the various story topics!

No responses yet

Gunnar Östberg

What are your thoughts?