

HOPE CORP · IKI · FREE EDITION

THE OPERATING BRAIN

How a D2C HealthTech founder builds a company that
runs on AI, from the order book to the investing room



Author

Nguyen Van Hung

Founder, Chairman & CEO · HOPE CORP · IKI Beauty & Wellness

contact@ikihealing.com · ikihealing.com/founders

WHY READ THIS BOOK, AND WHO WROTE IT

Most books about AI are written from outside the business: the author observes, analyzes, predicts. This book is written from inside. I am the founder and Chief AI Officer of HOPE, and what you are about to read is the system I run every day, with the books to verify it.

The evidence, stated briefly so you can judge for yourself:

- In October 2026, HOPE is a member of Anthropic's Claude Startups program.
- A team of 12 people, 9 of them full-time, runs on a three-layer AI system (executive, department, task). More than 60% of the work is automated.
- 1,294 paying customers from March to September 2026. 40.5% of new customers came back to buy again; repeat customers generate roughly 66% of revenue.
- Gross margin is about 60 to 65%. Net revenue for the first nine months of 2026 is about VND 3.57 billion (roughly USD 137,000), reconciled across orders, shipping waybills and bank statements.
- Ad spend is only about 1% of revenue, because customers come mainly from the community: about 240,000 followers and about 9,700 Zalo members.

HOPE is still small, and I say so plainly. But this is a real business with real paying customers and books that can be reconciled, and most of the daily work is done by a disciplined AI system. That is where the value of this book lies: not a promise about the future, but what has run, what has broken, and how we fixed it.

If you are a D2C founder, you get a map to build your own. If you are a health professional with a community, you get a way to multiply your voice without losing trust. If you are an investor or partner, you see how a small team controls AI with mechanisms rather than with reminders.

TABLE OF CONTENTS

Why read this book, and who wrote it

Foreword: The 8.9 million figure

Part 1: Mindset

- Chapter 1: The health D2C founder in the age of AI
- Chapter 2: The DO, MEASURE, LEARN loop
- Chapter 3: The six-piece model

Part 2: Building the operating brain

- Chapter 4: One source of truth for revenue
- Chapter 5: Leads and customer care, a machine that reminds people to act
- Chapter 6: The 24/7 advisor bot
- Chapter 7: Automation and the "failure must be loud" principle

Part 3: Marketing and content with AI for the health industry

- Chapter 8: The content factory and SEO/GEO blog
- Chapter 9: Community, Zalo and the path from webinar to distributor

Part 4: Legal and ethical guardrails

- Chapter 10: Dietary supplement guardrails, written in code
- Chapter 11: Personal data when using AI
- Chapter 12: AI permissions, money and the code repository

Part 5: AI for fundraising

- Chapter 13: One set of numbers for every form

Part 6: The 90-day roadmap

- Chapter 14: A 90-day roadmap to build your own operating brain
- Chapter 15: For health professionals with a community

Closing Appendix A: Prompt library Appendix B: Tool table Appendix C: Glossary About the author

FOREWORD: THE 8.9 MILLION FIGURE

One morning in August 2026, my operations advisor system reported a number: August revenue was only VND 8.9 million (about USD 340). (This and similar figures in the book are point-in-time measurements, not current numbers.)

If I had read that number without doubt, I would have believed the company was collapsing. Luckily I knew the order book in the sales chat group held 98 orders at that moment, about VND 179.3 million (about USD 6,900). The 8.9 million was not the result of a weaker AI. It came from something else: the machine had read a dead data source, then fallen back to a backup source, the cash-on-delivery amounts collected by the carrier. The machine answered with great confidence, great coherence, and was completely wrong.

I remember the feeling clearly. I was not angry at the machine. I realized I had misunderstood my own job. I had thought the job was to buy a smarter AI brain. It turned out the job was to make sure that brain read the truth. And to do that, I needed three things that no AI book teaches in full: understanding the business deeply enough to know which numbers deserve trust; knowing how to combine AI tools into an assembly line that actually runs; and having the safety discipline to keep that line from harming customers, the legal entity, or myself.

Those three things, put together, are the role I want to name in this book: the person accountable for AI in a D2C health business. Large companies call it Chief AI Officer. In a small company like mine, the role sits on the founder's shoulders, and if the founder does not do it, no one will. I am the founder and Chief AI Officer of HOPE. It is a role I perform for real, every day, in my own company.

Why I chose AI, D2C and HealthTech all at once

I did not choose these three words because they sound fashionable. I chose them because they fit together in a very specific way.

D2C, selling directly to consumers, is a model that lives on repetition. A customer buys, uses, runs out, buys again. At HOPE, among customers who bought for the first time between March and June 2026, 40.5% came back to buy again, and customers with two or more purchases generate about 66% of revenue. A model that lives on repetition needs a machine that never forgets: who bought what, when they are about to run out, who is waiting for a reply. Humans forget a lot. AI does not, provided you give it the right data.

The health and beauty industry, seen from inside, lives on trust. Customers do not buy an object, they buy a daily habit and a companion. Lose trust once and you lose both the customer and the people they would have referred. Trust is easily lost because the industry carries its own risk: a single over-claimed sentence can breach advertising rules, and AI is very good at writing content that over-claims. So the person running AI here cannot only know technology. They must know the law, know the brand's voice, and know when the machine must stop and let a human decide.

And AI, the way I use it, is multiplication. A small team with the right system can do the work of a much larger one. HOPE's three-layer AI system now handles orders, customer care, content, reporting and payroll, automating more than 60% of the work of a 12-person team. Its cost grows with software rather than with headcount. But multiplication multiplies mistakes too. One wrong number becomes a wrong report, then a wrong decision. Every chapter of this book returns to the same idea: get it right first, then scale it.

A Chief AI Officer does not have to be an engineer

I started as a software engineer, worked for many years in Japanese corporate environments, and am currently pursuing a Master's degree in Economic Law. I say this not to boast but to be honest: that background helps me a lot, yet the biggest share of value in HOPE's system does not come from my coding skill. It comes from knowing which question to ask: where does this number come from, who entered it, how late is it, and if it is wrong, who is accountable.

A good AI operator needs three capabilities:

1. Understand the business: know where the company earns money, where it loses money, which numbers are the truth and which are only shadows of the truth.
2. Orchestrate tools: know which tool does which job, chain them into an assembly line, and know where a human must stay in the loop.
3. Keep safety discipline: know what AI may touch (money, customer data, the code repository, content sent to the public) and block the rest with mechanisms, not with reminders.

You do not need to code to have the first and third capabilities. For the second, today there are coding assistants such as Anthropic's Claude Code that do the heavy lifting for you, as long as you can describe clearly what you want and read the results well enough to check them.

What this book is, and what it is not

This is the notebook of someone who has operated for real, recording what ran, what broke, and what I would do differently if I started over. It is not a theory book about AI. It is not advertising material for HOPE's products. And it is not a disguised pitch deck: company figures appear only where they help you understand a lesson.

On scale, I want you to have the right expectations. HOPE is a small business. Net revenue from January to September 2026 is about VND 3.57 billion (roughly USD 137,000), with 1,294 customers buying between March and September. Paid advertising spend in 2026 is only about 1% of revenue, because customers come mainly from the community. This is not a miracle story. It is an operations story, and its value is that almost everything in it can be redone at your scale. My main AI platform is Anthropic's Claude, alongside Google's Gemini for some tasks.

A legal note: this book describes how I work within the scope of my own dietary supplement products. Regulations change over time and by product type. The legal parts of this book are operating experience, not legal advice. Before applying anything to your products,

consult a lawyer or compliance specialist who knows your field.

How to read

The book has six parts. Part 1 is mindset: the health D2C founder in the age of AI, the DO, MEASURE, LEARN loop, and the six-piece model. Part 2, the thickest, is the operating brain: one source of truth for revenue, a customer-care reminder machine, an advisor bot, and the "failure must be loud" mechanism. Part 3 is marketing and content with AI for the health industry. Part 4 is legal and ethical guardrails. Part 5 is AI for fundraising. Part 6 is the 90-day roadmap, with a dedicated chapter for health professionals with a community.

Each chapter follows one rhythm: a true story, the principle, step-by-step method, a sample prompt, common mistakes, a checklist. If you are a health D2C founder, read in order. If you are a health professional with a community, read Chapters 1, 2, 8, 9, 10 and 11, then go to Chapter 15. If you are an investor or partner, Chapters 3, 7, 12 and 13 show you fastest how I think about risk and discipline.

Every company figure in this book comes from HOPE's books and measurement systems. Where something is not yet measured, I say it is not yet measured. The incidents I tell are real incidents, and I tell them because they teach more than the successes.

Let's begin.

PART 1: MINDSET

CHAPTER 1: THE HEALTH D2C FOUNDER IN THE AGE OF AI

The story: July fell 43%, and that number was wrong

In early August, in a working session with Claude to build a baseline table for campaign targets, I received a very crisp finding: July revenue was down 43% from June. The analysis had a table, percentages, and an explanation of causes. It read very convincingly.

I doubted it out of operating instinct: July showed no sign of such a drop. I asked for a recheck. The result: July had not fallen, July was higher than June. What had fallen was the figure recorded in the accounting ledger, because the person entering data was running late. Meanwhile, the carrier's waybills showed 127 orders already delivered but not yet in the ledger. The "43% drop" was a number about the data-entry process, not about customers.

The lesson I kept is not "do not trust AI." The lesson is: AI answers on the data you give it, and it is very good at making wrong data sound like a correct conclusion. The founder's role in this era has changed. It is no longer to produce the analyses. It is to know which analysis deserves trust, and why.

Principle 1: AI lowers the cost of execution, not the need for strategy

In the past, to get a daily revenue report I needed a person to spend a few hours every morning. To get steady content across channels, I needed a team. To have an advisor on duty around the clock, there was no way. Today each of those jobs can run on a machine at a cost close to zero compared with hiring.

But one thing does not get cheaper with AI: deciding what to do. Which customers to serve, which channels, what to say, what to refuse. AI helps you execute the wrong strategy faster. A business heading the wrong way at ten times the speed just arrives at the wrong place ten times sooner.

Principle 2: three layers of work, human decides, machine does, machine checks

After many collisions, I divided all work in the business into three layers:

The decide layer: work that depends on judgment, legal responsibility, and human relationships. Examples: approving a post that goes public, deciding to raise an ad budget, finalizing payroll, replying to an upset customer. This layer always belongs to humans.

The do layer: repetitive, rule-based, high-volume work. Examples: extracting orders from messages into the revenue ledger, drafting posts, drafting replies to customers, calculating payroll by formula. This layer goes to machines.

The check layer: work that watches the machines themselves. Examples: checking for banned words before a post goes out, reconciling the order ledger with shipping waybills, raising an alarm when a data source stops running. This layer also goes to machines, but humans design it and humans read its results.

In HOPE's system there is a short principle I repeat to everyone on the team: AI reads wide, writes narrow. The machine may read many things to understand the picture. But the right to write, meaning to change something in the real world, is granted very narrowly and with human approval. The machine drafts, the human approves. The machine proposes, the human decides.

Principle 3: three traits of health D2C

One, the rhythm of repetition: durable revenue comes from the second and third purchase. According to a measurement taken in early August 2026, about 71% of customers in the phone-number base had bought exactly once, so the biggest leak is the second purchase, and AI should be used to remember and remind at the right moment rather than to produce more posts. Two, trust is the main asset: customers notice every word, and an over-reaching promise destroys trust much faster than it can be built. Three, strict regulation: if you use AI to write content without a banned-word gate, you are running a risk-printing machine (Part 4).

How to do it: map your work in one afternoon

Step 1: List every repetitive task in the business during a week. Do not skip small ones such as customer reminders, copying numbers, end-of-day reports.

Step 2: For each task, record four things: who does it now, how many hours per week it takes, what happens if it goes wrong, and where the input data comes from.

Step 3: Sort into the three layers: decide, do, check.

Step 4: Choose exactly one bottleneck to tackle first, using these criteria: it consumes many hours, the consequence of error is moderate, and the input data already exists. Do not pick a task with a severe error consequence as your first.

Step 5: Ask one mandatory question about the chosen task: if the machine gets it wrong, how will I find out, and how soon.

Sample prompt: map your work

You are an operations advisor for a D2C dietary supplement business.
I will paste a list of repetitive tasks from my week. For each task:

1. Classify it: DECIDE (needs human judgment or accountability), DO (repetitive, rule-based), CHECK (watching the machine).
2. Estimate the consequence if the machine gets it wrong: low / medium / high, with one sentence of explanation.
3. List the input data needed and where that data comes from.
4. Suggest how I would detect that the machine got it wrong.

Finally, recommend EXACTLY ONE task to automate first and explain why.
Do not add tasks I did not list. Where information is missing, ask me.

TASK LIST:
[paste list]

Common mistakes

Automating first and asking about the data later. A machine runs very fast on dirty data.

Not recording who is accountable when the machine errs. A machine with no owner dies in silence.

Chapter 1 checklist

- I have listed all repetitive tasks and sorted them into the three layers: decide, do, check.
- I have picked a single bottleneck to start with.
- I know where that task's input data lives and who enters it.
- I have thought in advance about how to detect when the machine gets it wrong.
- Every automated task has a named owner.

CHAPTER 2: THE DO, MEASURE, LEARN LOOP

The story: 8,922 impressions, 302 clicks, 0 conversions

In a 30-day measurement window at the end of July, search ad campaigns for my blog content produced 8,922 impressions and 302 clicks, cost about VND 1.44 million (about USD 55), and recorded not a single conversion.

The team's first reflex was to suspect the measurement: perhaps the tracking tag was broken. If I had accepted that answer immediately, I would have fixed the wrong thing. Instead we measured the path of people arriving from the ads. The result: the tracking tag worked normally. The problem was the landing page. A reader saw only one call to action, to download the app, when all they wanted was to read. There was nowhere for them to leave an email or ask a question. The cause was a wrong call to action, not a broken measuring tool.

From that lesson, every call to action on the website was changed to collect an email first, then lead into a content sequence. I tell this story because it illustrates one thing: measuring without knowing what you are measuring is the same as not measuring.

Principle: DO, MEASURE, LEARN, then repeat

HOPE's operating principle is a four-step loop.

DO: produce a real result, with a source, then check it against the business's reality. Do not do things just to have done them.

MEASURE: compare against the plan. Not against a feeling. Every important task has an expected number written down before the work starts.

LEARN: turn the gap between plan and reality into a lesson, then load that lesson somewhere that people and machines can both read. A lesson does not live in one person's head.

REPEAT: do it again, with the lesson loaded.

And there is one sentence I treat as discipline: patch the jar before pouring water. Do not pour more customers into a leaking funnel. Before running more ads, check at which step the customers you already have are being abandoned.

Two wrong ways of measuring that I fell into

One, measuring button clicks instead of real work. The system recorded customer-care calls based on staff pressing a "called" button. When I looked at the number, I nearly concluded that nobody was calling. In reality staff worked mostly inside the messaging app and almost no one pressed any button. Later I changed the measure: a customer counts as advised when the real conversation contains a message sent by a staff member. Measure real behavior, not a flag.

Two, comparing against the wrong reference point. Once I measured staff response time by comparing against the moment the customer was assigned to them in the system. But Zalo customers were pulled into the system from existing conversations, so the "assignment date" was only the date the machine first saw them. Some customers had already been sent dozens of messages in the morning when the system recorded them in the afternoon. Measured against that reference, nearly every customer was wrongly concluded to be unattended.

How to do it: build a loop for one task

Step 1: Choose one task from the map in Chapter 1.

Step 2: Write three lines. Hypothesis: what I believe. Metric: if right, which number will change. Source: where that number comes from, who enters it, how late it is.

Step 3: Record the expected level and the check date before starting.

Step 4: Do it. Hold other variables constant where possible.

Step 5: On the check date, compare reality with expectation and write a paragraph of learning: what was right, what was wrong, what is still unknown.

Step 6: Load the learning into shared memory. At HOPE, that means the knowledge files the machines all read, and a campaign memory that records failed posts: posts that slipped through with banned words, posts the approver rejected, posts that failed to publish. Next time, the content-writing machine avoids repeating exactly the kind of failure that occurred.

Sample prompt: create a loop

I want to run a controlled experiment for a D2C dietary supplement business.

Task to test: [describe one task]

Help me fill in the framework below and point out where anything is still vague:

- Hypothesis (one sentence, testable)
- Primary metric and how to measure it (with data source, who enters it, delay)
- Expected level and evaluation date
- What would make me stop early
- What I will NOT change during the test
- Two ways this measurement could mislead me

Then ask me exactly three questions to clarify anything still missing.

Common mistakes

Not recording the expectation before starting. Then any result can be explained as a success.

Confusing a month in progress with a closed month. An unfinished month always looks low. My system attaches a separate warning flag to the current month, because without that flag the machine would conclude the company is collapsing.

Chapter 2 checklist

- Every important task has a hypothesis, metric, source and expectation recorded before the work.
- I know whether my metric measures real behavior or only a marker flag.
- I know whether my comparison baseline is fair.
- The lesson from each loop is loaded somewhere that both people and machines can read.
- I patched the hole before pouring more customers into the funnel.

CHAPTER 3: THE SIX-PIECE MODEL

The story: the machines ran perfectly and the organization still stalled

There was a period when my system ran very well technically: orders were extracted automatically, leads were distributed, reminders went out on time. Yet sales did not move. It took me a while to understand that the problem was not the machines. It was the organization around them: the people receiving the reminders did not see what they gained by following them, so they went through the motions.

I needed a framework to see the organization as a system, not just look at the software. I use the six-piece model: Vision, Strategy, Capability, Reward, Resources, Urgency. Whichever piece is missing, the organization produces the matching symptom.

Principle: whichever piece is missing produces that symptom

Missing	Symptom
Vision	Confusion: everyone understands the destination differently
Strategy	False start: lots of work, none of it connects into a path
Capability	Anxiety: people fear making mistakes because their skills fall short
Reward	Limited response: people know, but have no reason to act
Resources	Discouragement: they want to act but lack tools, time or money
Urgency	Indifference: everything is left for tomorrow

The way I interpret the six pieces, from personal experience, differs from the textbook version in a few places. Vision is looking three to five years ahead, clear enough that a new employee, after hearing it, knows where the company is going. Strategy is choosing only one to three core thrusts per year; the more thrusts, the fewer have enough force. Capability is turning the organization into a school: the leader learns a great deal of exactly the things that solve the current numeric problem, then teaches it back; with AI, the team's capability is knowing how to assign work to machines, read the results and doubt them. Reward is building a game, because if the scoreboard points to one thing while the bonus points to another, people follow the bonus. Resources are leverage: do not do 100% yourself. Urgency is backing yourself into a corner: a goal without a deadline is a wish.

Add one foundational element: virtue and merit (in Vietnamese, phuc duc). Do business decently, keep products clean, do not over-claim effects. In an industry that lives on trust, this is not empty ethics but a long-term risk strategy.

An example from the sales team's scoreboard

I made Reward into a designed game. The sales team's daily scoreboard does not weight all tasks equally: revenue-core work such as calling and closing leads has a weight of 5; brand-coverage work such as posting and making stories has a weight of 2; learning has a weight of 1. If all tasks scored the same, the team would pick the easiest, make lots of stories and neglect calling customers. For items the machine can measure, the scoreboard shows them next to what people self-report and colors the difference. I do not name and shame people who differ, because a difference may come from delayed recording.

How to do it: diagnose in 45 minutes

Step 1: Write down one symptom that is bothering you: for example, "the team knows what to do but does not do it."

Step 2: Score the six pieces from one to five, with concrete evidence for each score, not feelings.

Step 3: Find the lowest piece. Compare with the symptom table. Does your symptom match?

Step 4: Choose the smallest action that can raise the lowest piece by one level within two weeks.

Step 5: Measure again after two weeks.

Sample prompt: six-piece diagnosis

I will describe a symptom in my organization and the concrete evidence I am seeing. Use the six-piece model (Vision, Strategy, Capability, Reward, Resources, Urgency) to:

1. For each piece, give a short check question I need to answer with evidence.
2. Based on what I describe, infer which piece is most likely missing and why.
3. Point out where you are guessing because I have not provided evidence.
4. Propose ONE small action, doable in two weeks, to raise the lowest piece.

Do not give a firm conclusion when evidence is missing.

SYMPTOM AND EVIDENCE:
[describe]

Common mistakes

Using the model as a verdict. It is a diagnostic tool, not a conclusion. The same symptom can come from several pieces.

Setting too many strategic thrusts. Three is already a lot for a small team.

Chapter 3 checklist

- I scored the six pieces with evidence, not feelings.
- I know the lowest piece and its matching symptom.
- I have a two-week action for that piece.
- The team scoreboard rewards the work I actually need.
- This year's strategy has at most three thrusts.

PART 2: BUILDING THE OPERATING BRAIN

CHAPTER 4: ONE SOURCE OF TRUTH FOR REVENUE

The story: the rule was written in words, the code still read the old source

In August I decided the company's sales figures would no longer come from the accounting software. The reason was simple: a person enters the data by hand, records it a few days late, and July alone was missing more than a hundred orders compared with reality. A faster source, closer to the truth, was the chat group where staff post orders every day, together with the carrier's waybills.

I wrote that decision into the operations handbook, in words. Five days later I discovered the system was still reading from the accounting software in fifty-three different places in the code. Each place asked on its own and calculated in its own way. The company had been blind to its revenue for nine days and nobody complained, because everyone saw a number on the screen and no one knew where it came from.

Since then I have had one rule for every data decision: a rule written in words, and the code will still do something else. To block it, you must block it with structure. We built a single door for revenue. Every place that needs revenue goes through that door, and nobody may ask a source on their own.

Principle 1: one door, three layers, no mixing

HOPE's revenue door returns three layers, deliberately not merged into one number, because each layer answers a different question:

Layer 1, the order book: how much was sold. The source is order messages in the chat group, extracted by a machine into individual order rows.

Layer 2, waybills: how much was delivered, how much was collected on delivery. The source is the carrier's system.

Layer 3, bank statements: whether the money has actually arrived in the account.

When choosing which number to report: if layer 1 exists, use layer 1. If it is missing, fall back to layer 2, but with a warning attached. If there is nothing, return empty with a warning. And never silently fall back to a source that has been retired. This "never silently" rule is the core: a fallback source is allowed to appear, but it must say loudly that it is being used.

Principle 2: warning flags are part of the number

A number without context is a dangerous number. My system encodes three flags because without them it will certainly misread:

The current month always looks artificially low because the days are not over. On the sixth, looking at that month you see VND 8.9 million (a point-in-time measurement), and without the flag the machine concludes the company is collapsing.

The first month with waybill data is truncated because the carrier only recorded from mid-month. And when revenue and cost of goods come from two different sources, gross margin is indicative only.

There is one more flag that came from a real incident. A deep drop in the match rate between the order book and waybills does not mean orders are missing. When a customer pays by bank transfer in advance, the waybill records a collected amount of zero, so the match rate usually sits around 80 to 90 percent. But when the rate falls below 60 percent and the number of waybills is under 60 percent of the number of orders, it is almost certain that the scraper for waybill data has stopped. In August the book held 98 orders while waybills recorded only 7 (a point-in-time measurement). Without this flag, people would go hunting for "losses" that do not exist.

Principle 3: one tax convention, in one place

The order book records the amount the customer actually paid, including value-added tax. The profit and loss report needs the pre-tax figure. Once we fed the paid amount straight into the report and inflated revenue by 8%. Worse, revenue is the basis for calculating sales pay, so dividing wrongly here means underpaying staff, and that did happen in the July payroll period. Now the division by 1.08 sits in exactly one place in the codebase, and everywhere else calls it.

Principle 4: do not use the total row count as the monthly figure

Once I was told the system had 9,256 leads. When I filtered by creation date, in fact 9,098 rows had been imported on the same day in one bulk import. Monthly figures must be filtered by creation date, excluding bulk imports, and you should always look at the

distribution by day before trusting a total. That old lead stock later turned out to be mostly a dead stock with no phone numbers: the real constraint on growth sat in the source of new customers with contact information.

Step-by-step method

Step 1: List every place in the business that fetches revenue. Ask each one: where does this number come from.

Step 2: For each source, record three things: who enters it, the delay, and the consequence if it is wrong. Rank sources by closeness to the truth.

Step 3: Choose the primary source. For a D2C business selling through messaging, it is usually where orders are first reported. For a D2C business selling through a website, it is the payment gateway.

Step 4: Build the single door. Ask a coding assistant (I use Claude Code) to create one function that returns revenue, and replace every other place with a call to this function.

Step 5: Declare the fallback order and force every fallback to come with a warning.

Step 6: Declare the tax convention, shipping-fee convention and gift convention in one place.

Step 7: Attach freshness information to every report: how recent the source is, and how many messages have not yet been extracted.

Sample prompt: build the revenue door

You are a software engineer working on my codebase. Goal: have exactly ONE function that returns revenue.

Step 1 (read-only, change nothing yet): find every piece of code that queries revenue or order data. List them in a table: file, function, data source, calculation method (is tax removed, are gifts removed).

Step 2: propose a design for the single door including: source priority order, fallback conditions, the warning attached to each fallback, source age (when the freshest source was last updated).

Step 3: wait for my approval of the design before changing anything. When changing: one small commit per change, do not touch files outside the scope.

Rules: never silently fall back to a retired source. If there is no data, return empty with a warning, never invent numbers.

When done, write a test proving that when the primary source is empty the function returns a warning.

Common mistakes

Believing a rule written in words. Block it with structure.

Merging the three layers into one tidy number. Each layer answers a different question.

Forgetting to remove tax, or removing it in the wrong place.

Chapter 4 checklist

- There is a single function that returns revenue, and every other place calls it.
- Each source has an owner who enters it, a delay, and a priority order.
- Every fallback comes with a written warning.
- The tax convention lives in exactly one place.
- Every report has a data freshness line.
- The current month always carries its own flag.

CHAPTER 5: LEADS AND CUSTOMER CARE, A MACHINE THAT REMINDS PEOPLE TO ACT

The story: a good-looking number hid a bad habit

The figures in this chapter are point-in-time measurements (late July to early August 2026), not current figures.

My first lead-distribution machine worked on reasonable logic: a new customer arrives and is assigned to one staff member; if not touched within 15 minutes, it is pulled to someone else; if cold for 48 hours, it is reassigned; after three rounds, it is pushed on again. On paper, this is a system that never loses a customer.

When I measured it for real, I found two things. One, the machine counted by wall-clock time, not by working hours. A customer arriving at 9 p.m. would, by 3 a.m., be judged "neglected for six hours" and snatched to another staff member who was also asleep. The result was 1,059 hand-offs in 16 hours. Two, the deeper consequence: staff learned that messages from the bot were not trustworthy, and started ignoring them.

Worse, I made another discovery about Zalo customers: when a customer messages someone's account, only that person can open the conversation, and other staff see nothing. Yet 83% of Zalo leads were registered to someone who did not hold the conversation. Passing a customer to a person who cannot open the conversation just pushes the customer into empty space.

I decided to drop the snatching mechanism entirely. The sentence I said to the team at that time, translated from Vietnamese: I do not like the style of fighting over customers, so drop the pushing mechanism, and instead push staff to take care of customers as early as possible.

Principle 1: the machine reminds, the human acts, no snatching

Whoever owns the customer takes care of the customer. The machine does not move customers, it only reminds the named holder. The single exception: the holder has left the company, because a customer must not die along with a locked account.

Note one technical detail that is easy to get wrong: when turning off the snatching mechanism, do not turn off the whole cluster that contains the reminder cadence. We set a separate flag for "no reassignment" and forbade switching off the old flags, because switching off the old flags would also remove the reminder step, exactly the thing we most need to keep. Then we wrote a test suite of 21 scenarios, because the three places that used to push customers lived in three different functions, and missing one would leave the system green with no one noticing.

Principle 2: measure with real messages

My staff live inside the messaging app. They almost never press the "contacted" button in the system. If measured by button presses, 909 of 909 Zalo leads stayed in the "new" status, and the machine reminded even customers who had already been cared for. When we switched to reading the real conversation and counting a lead as advised if there was a message sent by staff, the result was 909 of 909 advised, with no lead neglected.

Two traps: do not compare the time of the staff message with the time the lead was assigned in the system, because Zalo leads are pulled from existing conversations, so the "assignment date" is only the date the machine saw them; and inspect every lead rather than filtering by "new" status, because filtering leaves zero rows on the next round and switches off the one reminder that is still correct.

Now the Zalo reminder only reminds about what can be measured: a customer sent the last message and nobody has replied, addressed by name to the conversation owner, at three times of day. The first measurement showed 27 customers waiting, the longest case six days. An uncomfortable number, but a true one.

Principle 3: documentation must sit at the point of action

A separate documentation tab is almost useless. Our customer-care sequence sits right next to each customer: a suggested message for the right milestone, a copy button, a "sent" button. When rewriting this sequence, I ran every sentence of the old version through the legal guardrails and fixed the places that did not match our principles, and then found one more technical bug: the milestone "28-30 days" was written as text, so the machine could not read it, and the repurchase step never appeared.

The new version sets the repurchase milestone at day 24, before the median repurchase cycle of about 27 days. The day-21 message changed from asking the customer for a review to offering an unconditional thank-you gift.

Step-by-step method

Step 1: Determine who holds each customer. Each customer has exactly one holder.

Step 2: Define the evidence of "cared for" as a real behavior in the messaging system, not a manually pressed flag.

Step 3: Count waiting hours in working hours.

Step 4: Build named reminders, limit their frequency, and exclude internal conversations from the customer list.

Step 5: Write the care sequence around the product lifecycle milestones, and place it next to the customer.

Step 6: Let the machine pre-draft replies, and a human reads and sends. Run every reply through the banned-word gate.

Step 7: Measure three things: response time, the share of customers waiting past the deadline, and the second-purchase rate.

Sample prompt: draft a reply for staff to approve

You are a drafting assistant for customer care staff at a dietary supplement brand.

Task: read the conversation below and draft ONE reply for the staff member to read, edit and send themselves. You must not send anything to the customer.

Rules:

- Warm, concise tone, forms of address matching the conversation context.
- Only speak of the product as a dietary supplement. Do not promise health results, do not compare with medicine, do not promise results by number of days.
- Promotions are gifts only; never mention percentage discounts.
- If the customer asks about a disease, symptoms, medication they are taking, or pregnancy: do NOT advise; draft a polite message suggesting the customer ask a doctor and mark [NEEDS HUMAN REVIEW].
- If price or stock information is missing, write [MISSING INFORMATION]; do not guess.

CONVERSATION:

[paste]

Common mistakes

Counting waiting hours by wall-clock time.

Measuring by button presses while the team works somewhere else.

Letting the machine send messages to customers without content control. In the health industry, a message is direct advertising content.

Chapter 5 checklist

- Each customer has exactly one holder, and the machine only reminds, never reassigns.
- "Cared for" is measured by real messages.
- Waiting hours are counted in working hours.
- The care sequence sits next to the customer and has passed the banned-word gate.
- The repurchase milestone is set before the median repurchase cycle.
- There is an automated test suite for the no-reassignment logic.

CHAPTER 6: THE 24/7 ADVISOR BOT

The story: a chat group that included people from outside the company

HOPE's AI assistant is named Tiểu Nhã, and the advisor that runs day and night is named Alex. Both are bots in a messaging app. They can be pulled into any chat group, and for convenience I let the bots open themselves up in every group they were invited to. One day during a review I saw the bot answering in 10 groups, including a group with people from outside the company. I played a member of that group and asked an ordinary Vietnamese question about the leadership's personal finances. The bot retrieved and returned the opening passages of the personal finance store. No hacking was needed, just typing one sentence.

This hole was not because the bot was dumb, but because I let it read one shared store for every chat. I patched it in two layers. Layer one is a hard lock at the retrieval step: the leadership's three private stores are cut from results everywhere except the private chat. Layer two is a guardrail in the prompt, forbidding talk about salary, cost of goods, valuation and personal debt. Layer two alone is meaningless because a model can be talked around an instruction. Layer one is the real lock.

Principle 1: permissions by place of conversation and by speaker

The CEO's private chat sees everything. The management group sees operating figures. A group with outsiders sees only public knowledge. The bot must know where it is speaking.

For the bot's skills, meaning what it can actually do such as assigning tasks, running research, creating campaigns, we use three permission levels: open level, writing nothing to the system; manager level, which writes or consumes resources; and CEO level. A new skill with no declared permission defaults to the CEO level, the tightest. If we err on the tight side, I type one sentence to open it. If we err on the loose side, nobody knows until something happens.

When blocked, the bot says clearly who can do it. Silence makes people think the bot is broken.

Principle 2: quick commands must not hijack sentences

The bot has several keyword-catching branches that run before the brain; for example the word "advertising" triggers the campaign list. The problem: any sentence that contains the keyword gets swallowed. I asked "is this month's advertising paying off?" and got back a campaign list with money buttons instead of an answer; "don't create posts today" still triggered post creation. This, not a weak brain, is what makes an assistant look stupid. The

rule: better to miss a command than to hijack a sentence. If a command is missed, I retype one word; if a sentence is hijacked, I lose the whole answer and think the machine is broken. We guard it with a recognition function: a long sentence is not a command, a sentence ending in a question mark is a question, a sentence with a negation word is not a command.

Principle 3: the bot tells the truth about itself

Once I approved a task that needed to run on my personal machine, and the bot replied "will pick it up in a few minutes" while the machine had been off for three days. Half of the task intake lived in the cloud, half of the work lived on my machine, and the cloud did not know whether the other half was alive. It was not that the bot was lazy, the bot was blind. We patched it with a heartbeat: the worker machine writes a timestamp even when there is no work, to distinguish "no work" from "machine off." The bot reads that rhythm and honestly reports the status. The watchdog only raises an alarm when a task has waited more than 45 minutes, because an alarm for something nobody intends to fix at two in the morning is a false alarm.

Principle 4: the advisor must dare to push back

An advisor who only nods is useless. Alex's root instructions force it to push back first, cite real figures and point to the real bottleneck. When I said I intended to double the ad budget because sales were slow, Alex refused, pointed out that the bottleneck lay with customers we already had who had not been touched and customers waiting for replies, with figures attached. That is the kind of pushback I want.

Principle 5: an exam set for every machine

Each important AI machine has an automated exam set taken from real cases, run every Sunday: the operations advisor 22 cases, lead distribution 16 cases, Zalo customer replies 15 cases, sales-quality grading 13 cases. Latest scores as of 10/10/2026: 20 of 22, 16 of 16, 15 of 15, 12 of 13. The rule: a machine may be given more work, or switched to a new AI model, only when its exam score is equal or higher. The exam sets include cases that deliberately try to lure the AI into breaking the rules, to test the legal guardrails. This turns "is the AI okay" from a feeling into a number.

Step-by-step method

Step 1: Identify the chat groups where the bot may speak and classify each by trust level.

Step 2: Split the knowledge stores by sensitivity. Cut the sensitive stores at the retrieval step, not only in the prompt.

Step 3: Declare permissions for each skill, defaulting to the tightest.

Step 4: Write a command-recognition gate that prefers missing a command over hijacking a sentence.

Step 5: Write 10 to 20 real cases, with expected answers, into an exam set. Run it weekly.

Step 6: Give the bot the ability to tell the truth about its own status.

Sample prompt: root instructions for an operations advisor

You are the operations advisor of a D2C dietary supplement business.

Answering principles:

1. Rely on the figures provided. If figures are missing or old, say clearly which date the source runs to; do not infer it into a conclusion.
 2. When the user proposes an action (raise budget, change a campaign), push back first: point to the real bottleneck, cite numbers, then give your opinion.
 3. End every report with a concrete RECOMMENDATION, not just numbers.
 4. You are answering in: [private chat / management group / group with outsiders]. In a group with outsiders, never discuss salary, cost of goods, valuation, debt or customer data.
 5. Never carry out a spending action yourself. Only recommend.
 6. Do not use markup formatting if the channel does not support it.
- If asked about something beyond your authority, say clearly who can help; do not stay silent.

Common mistakes

Relying only on prompt instructions instead of cutting at retrieval.

Never grading the bot. Bots degrade very quietly when the model or the data changes.

Chapter 6 checklist

- The bot knows which group it is in and the matching permissions.
- Sensitive stores are cut at the retrieval step.
- A new skill with undeclared permissions defaults to the tightest.
- There is a command-recognition gate against hijacking sentences.
- There is a heartbeat for the worker machine, and the bot tells the truth when the machine is off.
- Each bot has a weekly exam set.

CHAPTER 7: AUTOMATION AND THE "FAILURE MUST BE LOUD" PRINCIPLE

The story: the number 1,481 overwritten with zero

One of our scrapers, every day, pulls the number of active customers from the customer-care software. One day the login session expired. The machine did not raise an error. It read back an empty table, treated that as truth, and wrote the number zero over the 1,481 that was already there. The whole dashboard then showed the company had zero active customers.

Nothing was destroyed. It was only one wrong number in one table. But if I had not noticed, the next morning the advisor would have read that number and advised me to make correct decisions on wrong data.

The lesson formed two rules. One: failure must be loud. A scraper script that returns no rows exits with an error code and writes a flag into the configuration. A failing route must not return a success code. A new data source must be plugged into the watchdog system. Two: a blank scrape stops, it does not write. Keep the old number and record the failed state elsewhere.

Principle 1: failure must be loud, and loud in the right place

There is a deeper trap we fell into. One of the team's messaging accounts was disconnected for several days, yet every panel stayed green. The reason: the indicators only answered "did the script run," not "is the data right." The software kept returning the old conversation store it had saved. It was wrong in both directions: customers who messaged that account did not enter the system, and old conversations still sat in the "waiting for reply" list, aging a day more each day, enough to spawn an accountability hunt over something that did not exist. We added a measurement that directly checks the status of each account and watches the age of the last message.

The rule: every measurement must check the place where the real data lives. Whenever you add a source, ask at once: in how many ways can this thing fail.

Principle 2: an alert must become a task

One day I received two alert messages listing six dead data sources, each line with a ready-made fix command. Nobody did anything. The failure was not in detection but in handover: a line of text in a chat group is not a task, it has no owner, no deadline, and the next message pushes the previous one away.

Now each dead source automatically becomes exactly one task in the task book, with an owner. Four deliberate constraints: each source has only one open task, so the machine does not spawn twelve tasks a day for one source; when the source revives, the machine closes the task by itself; the task escalates with days dead, and by day three the wording changes from "rerun the script" to "decide: fix it for good or retire it"; a source already decided to be reference-only spawns no task.

Principle 3: catch-up runs, but know when to stay silent

One night the internet went down. The schedules still fired, but every script died immediately for lack of a connection, and nobody reran them. By morning 13 sources were reported dead. The "blank scrape stops" safeguard kept the old numbers from being corrupted, but nobody went to fetch fresh ones. The catch-up layer reads the timestamp each scraper writes for itself, and reruns any source that is overdue. Constraints: if the internet is down, stay silent, because shouting every time the wifi drops is a false alarm; run sequentially with a lock because several jobs share one browser profile; do not touch sources that died because of a login session; limit the number of tasks per round.

Principle 4: watch the rhythm, not only whether it exists

A watchdog that only asks "was a post published today" is blind when failure goes in the direction of excess. In August, a commit labeled "add 4 posts" actually added 305 blog posts, because the posts were already sitting on disk and had not yet entered the repository. A young domain suddenly sprouted 300 addresses in one day, after which more than 300 pages were put by the search engine in the "discovered, not yet indexed" state. We cut the blog publishing rhythm to about 10 posts per week, a 92% reduction, and added three checks: one that blocks at commit time if more than 5 posts are added, a rule to count posts correctly and write that exact number into the commit label, and a watchdog that counts the posts on the web every day and raises an alarm if the count rises by more than 5.

Step-by-step method

Step 1: List every data source and every background machine in the business.

Step 2: For each source, record a "last successful run" timestamp and a "last time with data" timestamp.

Step 3: Apply the blank-scrape-stops rule to every scraper.

Step 4: Build a single watchdog location, with its own age threshold for each source.

Step 5: Connect alerts to tasks with owners.

Step 6: Each quarter, simulate an incident to see whether the alarm chain works.

Sample prompt: review an automated machine

I will paste the code of an automated script/machine. Review it against the "failure must be loud" checklist:

1. If the data source returns empty or an error, does the script overwrite old data? Point to the specific line.
2. Does the script exit with an error code when it fails, or swallow the error?
3. Is there any early return/exit that skips later checks?
4. Does it write a "last successful run" timestamp? Does it distinguish "ran" from "had data"?
5. Is failure in the EXCESS direction (more than normal) detected?
6. Propose the smallest fix for each issue. Do not refactor unrelated parts.

If you are not sure, say clearly that you are not sure.

CODE:

[paste]

Common mistakes

Watching only "did it run" and ignoring "is the data right."

Building an alarm that no one picks up. An alert without an owner is a dead alert.

Watching a source you have already decided to retire. That is a deliberate false alarm.

Chapter 7 checklist

- Every scraper has the blank-stops rule and exits with an error when it fails.
- Each source has a last-success and a last-data timestamp.
- Alerts automatically become tasks with owners, one task per source.
- There is a catch-up layer and it knows to stay silent when the internet is down.
- The watchdog also watches the excess direction.
- At least one incident was simulated this quarter.

PART 3: MARKETING AND CONTENT WITH AI FOR THE HEALTH INDUSTRY

CHAPTER 8: THE CONTENT FACTORY AND SEO/GEO BLOG

The story: a batch of three posts, 34,323 characters, none usable

My content factory was once designed to write three posts in a single call to the AI model, to save cost. One day I measured it on the live system: the model poured out 34,323 characters without ever closing the data structure, so the system could salvage exactly zero posts. The whole batch was lost. What was irritating was that running the same prompt on my own machine passed, twice in a row. The failure was random per call, so a single manual run proved nothing.

There was a quieter issue too: the tokens the model uses for thinking are counted against the same limit as the answer text, and they differ each call, so truncation errors came and went. In two other places it broke silently: the brand-voice grading gate ran out of quota before grading and always fell back to "pass for now," so for weeks posts were not graded for voice at all; and the comment classifier returned empty, so every comment was treated as an ordinary question, including questions about illness that should have been left for a human to see.

Two lessons: write one post at a time, never batch; and when the model needs to think, raise the text ceiling along with it.

Principle 1: content follows the campaign calendar, not inspiration

The factory consults the campaign calendar before writing. Each day has a quota: two top-of-funnel awareness posts, one mid-funnel post, two bottom-of-funnel posts that drive action. When written, a post is marked "drafted"; when approved and published, it is marked "published"; and the marks only move forward, never back. The number I like to watch most on the calendar is the execution rate: what percentage was published out of what was scheduled. Many scheduled and few published means the problem is not in planning, and building more calendars will not save you.

When the factory writes, it is loaded with context: the brand DNA set, the knowledge store, posts that won and lost, and the campaign memory that records failed posts. That way the machine avoids repeating the old kind of failure.

Principle 2: a human approves, no fully automatic posting

A machine-written post goes through the banned-word gate and then to a human approver with one tap on the phone: published or dropped. We deliberately do not post fully automatically, for two reasons: social platforms throttle the reach of posts made through automation tools, and the risk of breaching advertising rules is far more unpleasant than the convenience of not having to press a button. Staff copy the post and publish by hand. It costs a few minutes per post, in exchange for a layer of control with a human looking.

The image machine also has its own rules: still-life or setting images, no close-ups of people because free tools often distort faces and hands, no text in images, no product packaging. For images with text, use a separate text-drawing tool so the Vietnamese diacritics are always right.

Principle 3: a self-compensating layer, but compensate one post at a time

When the hourly writing run fails, a make-up layer runs at noon, afternoon and evening, counts the day's posts, writes make-up posts if short (at most three per round), and stays silent if the count is complete. It raises an alarm only if the make-up layer itself fails, because then the whole day is blank. Before that, the only way to get a post was for me to type "create post" myself. A machine that goes asking a human is a machine doing the wrong job.

Principle 4: a blog for search engines and AI answer engines alike

Our blog has more than a hundred Vietnamese posts, with English and Japanese translations of the main ones. I write for two kinds of readers at once.

The first reader is the traditional search engine. The second is the AI answer engines, systems that read many pages and synthesize them into one answer with sources. For the second reader, anything that makes it easier for a machine to cite helps: a clear definition at the top of the post, a question-and-answer structure, original data with sources, update dates, and a site description file for machines to read.

Three things I took away: original data is the only unique asset, so the weekly data reports kept their rhythm even when we cut everything else; rhythm matters more than volume, because a search engine's crawl budget is a fixed pie divided by domain authority, and once hundreds of pages are queued and rejected, each new post is both useless and pushes

the posts you want ranked to the sidelines (the 305-post story in Chapter 7); and every post must connect to an action, prioritizing posts with buying intent that link to product pages, and every call to action leads to email capture so the reader enters a thirty-day content sequence.

Step-by-step method

Step 1: Write the brand DNA into a file a machine can read: voice, preferred words, banned words, forms of address.

Step 2: Build the campaign calendar on three rhythms: quarter, month, day. Daily quotas split by funnel.

Step 3: Build a one-post-at-a-time writing process, with the banned-word gate, then a human approver.

Step 4: Store and reload winning posts, losing posts and failed posts.

Step 5: For the blog, list topics by buying intent and by real customer questions. Write long posts with a question-and-answer structure, original data, and dates.

Step 6: Set a cap on publishing rhythm and a block when too many posts are added.

Sample prompt: a brief to write one grounded post

You are a content editor for a dietary supplement brand. Write ONE post, not several.

Input:

- Topic / question customers often ask: [..]
- Funnel stage: [top / middle / bottom]
- Related product (use only the information below, do not add benefits yourself): [..]
- Brand voice and banned words: [paste the DNA set]

Requirements:

1. Open with a one-sentence definition or a direct answer.
2. Include a short question-and-answer section at the end.
3. Every figure must have a source that I provide. If a source is missing, write [SOURCE NEEDED]; do not invent.
4. The product is a dietary supplement. No medical effects, no comparison with medicine, no promise of results by number of days; promotions are gifts only.
5. End with a light call to action to capture email.
6. At the end, list the sentences you think carry legal risk so I can review them.

Common mistakes

Mass publishing because it is "ready." Rhythm is what search engines read.

Letting the AI supply figures freely. Unsourced figures are a risk to both reputation and legal standing.

Fully automatic posting in an industry with heavy legal guardrails.

Chapter 8 checklist

- A brand DNA set and banned words are loaded into the machine.
- Content follows the campaign calendar, with quotas by funnel.
- Written one post at a time, through the banned-word gate, through a human approver.
- There is a make-up layer that knows to stay silent when the quota is met.
- The blog has a question-and-answer structure, original data, and update dates.
- There is a publishing rhythm cap and a block.

CHAPTER 9: COMMUNITY, ZALO AND THE PATH FROM WEBINAR TO DISTRIBUTOR

The story: the bigger the group, the quieter it gets

I still thought of community as my biggest asset: about 240,000 followers across platforms and about 9,700 members in Zalo groups. When I decided to measure for real, the result (point-in-time figures from mid-August 2026) made me sit down.

In the large community groups we post five to six times a day. The share of customers who speak is only 0 to 4%. Four groups with a combined 1,872 people had their last hundred messages all posted by us, without a single customer speaking. By contrast, a group where we post only 0.3 times a day had 5.59% of people speaking, the highest in the system.

After excluding 16 groups that are not customer bases (groups outside the company, internal working groups, and two groups with misleading names), the final number is 41 groups, 9,612 customers, with an average of 3.78% speaking. Split into three tiers: large communities of 100 or more members reach 1.59%; student classes 2.91%; small groups 11.58%. The posting frequency of the three tiers is the same. The bigger the group, the quieter.

The point I most want you to note is what we have not dared to conclude. The correlation between posting frequency and silence is weak: some groups with six posts a day still have more than 4% speaking. That is not enough to say that posting a lot makes a group go quiet. To know, we must test by cutting frequency and measuring.

Principle 1: measure what can be measured, say plainly what cannot

There are three places we cannot measure and forbid ourselves from inventing numbers: likes and reactions, because the system returns an empty value for all messages, which means not measurable, not that nobody reacted; conversion from group member to order, because member phone numbers cannot be obtained; and distinguishing insiders from customers, which we handle with an approximate rule: a user ID present in at least three groups is an insider. Without this step, the speaking rate is inflated, because one group once had 21 of 66 "customer" messages that were actually one account reposting the company's own posts.

Principle 2: the grouping key must be something that always exists

We once grouped by an identifier, but when the tool was flaky, the readable and unreadable versions of the same group fell under two keys, and 57 groups ballooned into 98. The grouping key must be something that always exists. And when cutting a threshold for presentation you must list what was cut: student classes once dropped out of the summary table simply because they were below the threshold, and I lost trust in that table for a while.

Principle 3: the warmest audience is people who paid to learn

The student classes, where customers have paid to join a short course, turned out to be the warmest audience. One class reached 10.34%, the highest in the system, while a 651-person community had 0% speaking. Yet the machine still treated a 30-person class exactly like a community of 886 strangers. A proposal is pending decision: do not cut frequency in the classes, cut it in the large communities that are silent; assign one person to be responsible for the whole cluster of classes; and the class content should ask questions after the course instead of resending introductory posts.

Principle 4: the path from webinar to distributor

I describe a framework for the path rather than publishing figures, because this part has not yet been reconciled enough. The webinar is an open door for people who are considering; registrants who leave their information give explicit consent; afterwards they enter a content sequence and a small, highly interactive group; anyone who asks in depth about becoming a distributor gets a person in charge to talk to directly, with a clear policy. Promotions are always gifts, not percentage discounts. Each stage must be measurable: registrations, views, group joins, distributor inquiries, signings. Where something cannot yet be measured, write that it is not yet measured.

Step-by-step method

Step 1: List all community groups, and remove internal groups and groups outside the company from the customer base. Only insiders can tell which groups have misleading names.

Step 2: Measure each group: number of members, number of your posts, number of customer messages, speaking rate.

Step 3: Split into three tiers, then try changing frequency in one tier, and measure again after two to four weeks.

Step 4: For student classes, assign one person in charge and use post-course question-asking content.

Step 5: Build the webinar-to-distributor path, with explicit consent at the start.

Sample prompt: analyze community vitality

I will paste a table of community group data (group name, number of members, number of brand posts in 30 days, number of customer messages).

Please:

1. Calculate the customer speaking rate for each group. Present the DETAILED per-group table first, the summary table after.
 2. Group into tiers by size and calculate the average rate for each tier.
 3. State clearly what the data does NOT allow us to conclude (for example, correlation is not causation).
 4. Propose ONE experiment changing posting frequency in one tier, with a metric and a measurement period.
 5. If there are groups I removed from the table, list their names and the reason for removal.
- Do not infer phone numbers, customer names, or order conversion if I have not provided them.

Common mistakes

Counting insiders in the customer base.

Reading an empty value as "none." Empty means not yet measured.

Building the distributor path before there is a clear policy and registrants' consent.

Chapter 9 checklist

- The customer base has excluded internal groups and groups outside the company.
- Each group has a speaking rate, with the detailed table before the summary table.

- I know the three places that cannot be measured and I do not invent numbers.
- There is one posting-frequency experiment with the expectation recorded in advance.
- The webinar-to-distributor path has explicit consent and a metric for each stage.

PART 4: LEGAL AND ETHICAL GUARDRAILS

CHAPTER 10: DIETARY SUPPLEMENT GUARDRAILS, WRITTEN IN CODE

The story: the product page said one thing, the product was another

In August I reviewed all the old product descriptions in my online store with a script that read every page. The result: some sentences went beyond the legal guardrails we had set ourselves, from language that sounded like adjusting medical markers to antibacterial language.

These descriptions were not written by AI. They were written by humans long before, and nobody had reviewed them. The first lesson: legal guardrails are needed for the whole old content stock, not only for new content generated by AI. We reviewed the old product descriptions and corrected the sentences that crossed the line, and later removed hundreds of old posts that no longer fit current advertising rules.

The second lesson came later: if you do not have an automatic check gate, you will only discover violations when it is too late, and usually because a marketing employee or a competitor points them out to you.

Principle 1: positioning first, wording after

HOPE and IKI products are dietary supplements. Not medicines, not methods for correcting medical conditions, and they must not be described as such. From that positioning, four kinds of wording are blocked in all outgoing content:

One, words asserting the ability to end disease or manage disease, and words about expelling toxins from the body in the medical sense.

Two, commitments to results with a number of days.

Three, showing customers' health results as proof, such as before-and-after cases.

Four, promotions as percentage discounts. Our promotions are only gifts.

Deliberately, I do not print the specific word list in this book. A banned-word list not confirmed by your own legal counsel is dangerous to copy. And this book itself was reviewed against those rules before reaching you. Your list should be confirmed by someone with legal expertise in your field.

Principle 2: one single check gate

All machine-generated content, from posts and blog articles to customer replies and care sequences, goes through one single banned-word gate. One gate means fixing one place; two gates means two versions of the rules that drift apart. The gate must handle three subtle things we learned at a price.

A negation is not a violation. A disclaimer such as "this product does not replace medical advice" contains a phrase that sits in the banned list, and a naive gate will flag every clean post that carries the disclaimer. I saw three clean posts out of three flagged red. The consequence is not that posts get blocked, but that the approver grows numb to red flags and clicks through a real banned word. The fix: strip the valid disclaimer first, then scan, and do not loosen the list.

Hashtags written without diacritics slip through, because a gate that reads accented text is blind to unaccented text. We added a separate step that scans hashtags after removing diacritics.

Old content does not clean itself: a care sequence written before the gate existed contained three violations nobody saw (Chapter 5).

Principle 3: the human approver remains the final stop

AI drafts, the gate checks, the human approves. The weekly exam set contains cases that deliberately try to lure the AI into breaking the rules, to check that the guardrails still stand after every model change or prompt change.

Step-by-step method

Step 1: Write a one-page positioning statement: what the product is and is not. Have legal counsel confirm it.

Step 2: Compile the banned-word list and banned sentence patterns, classified into the four groups above, confirmed by someone with expertise.

Step 3: Build a single check gate and call it on every exit path of content.

Step 4: Scan the entire old content stock through the gate.

Step 5: Write a test set including clean cases, cases with a valid disclaimer, cases with a real banned word, and cases written without diacritics or abbreviated.

Step 6: Rerun the test set whenever you change the prompt or the model.

Sample prompt: review a page against the legal guardrails

I will paste the content of a product page or post for a dietary supplement brand.
Review rules (provided by my legal counsel, full list below):
[paste your banned-word list and banned sentence patterns]
Please:
1. List each violating sentence, quoted verbatim, with the rule it violates.
2. Distinguish negations or valid disclaimers from real violations.
3. Propose one rewrite for each violating sentence, keeping the communication intent but removing medical claims and promises of results.
4. Mark the places you are unsure about for someone with legal expertise to decide.
Do not add benefits that are not in the product data.

Common mistakes

Watching only AI-written content and forgetting the old human-written content stock.

Having a different gate for each channel.

Loosening the list to avoid false alarms, instead of stripping the disclaimer.

Chapter 10 checklist

- There is a product positioning page and a banned-word list confirmed by legal counsel.
- Every content exit path goes through exactly one gate.
- The gate handles disclaimers and unaccented hashtags.
- The old content stock has been scanned.
- There is a test set that reruns whenever the model changes.

CHAPTER 11: PERSONAL DATA WHEN USING AI

The story: reviewing every path that touches customers

In early September, after a new Government decree on penalties in the field of cybersecurity and personal data took effect, I had the paths where our system touches customers reviewed. In the review, we found some forms with no consent box, some bulk

emails with no unsubscribe link, and a sentence in the website's policy that was inaccurate about where data is stored. We fixed all three.

Most small D2C businesses in Vietnam are in exactly that position, and AI makes the problem bigger faster: a mailing machine never tires, so it can violate ten thousand times in one night.

Principle 1: consent, purpose, right to refuse

Decree 13 of 2023 on personal data protection sets the foundation: collection with consent, a clearly stated purpose, and health-related data treated as a sensitive type that needs more care. The penalty decree, effective from August 2026, is more specific for operations teams: advertising messages, emails and calls need prior consent; there are permitted time windows and a limit on the number of messages per day; there must be a simple way to refuse and a label that identifies advertising; consent must not be pre-ticked, must be separated by purpose and recorded in a log. I summarize in the way an operator understands it, not the way a lawyer writes it. This part is operating experience, not legal advice; ask a lawyer about your own case.

Principle 2: one door for every messaging path

Just like revenue and banned words, messaging customers must go through one door. Every path for messaging, calling and emailing must check: has this customer already refused, are we inside the permitted time window, does the advertising email have a label and an unsubscribe link. Every new form must send a consent record block. When a customer refuses, a single record is written and every channel reads that record.

Principle 3: AI sees only what it needs to see

Customers' phone numbers and emails are masked before being passed to a model or into a report; logs only count the number of calls and tokens and do not store content. The assistant speaks only in permitted groups, and sensitive stores are cut where outsiders are present. A guard on the code repository refuses commits that contain lists of phone numbers, secret keys or secret configuration files. For recruitment profiles, accept only people who submit through a form with consent, do not scrape profiles from platforms or social networks, and store extracts instead of the original files.

Step-by-step method

Step 1: Map your data: what you collect, from where, where it is stored, who sees it, who it is sent to, including AI tools.

Step 2: Every collection point has explicit consent, separated by purpose, not pre-ticked.

Step 3: Build one table and one function for "has refused." Every messaging path checks this function.

Step 4: Mask identifying data before it enters a model, and decide clearly which model may receive which kind of data.

Step 5: Plug in the commit guard.

Step 6: Name one person responsible for personal data, even in a very small company.

Sample prompt: map your data

I run a D2C dietary supplement business in Vietnam. I will describe the tools I use to collect and process customer data.

Please build a data map table: type of personal data, purpose, existing consent basis (yes/no/unclear), where it is stored, who has access, whether it enters AI tools.

Then:

1. List the points where health data or sensitive data may enter the system.
2. Point out the messaging or calling paths that lack a refusal check or a time-window check.
3. Propose a remediation priority order by risk level.

State clearly that this is an operational review and needs a lawyer's confirmation before it is applied.

DESCRIPTION OF TOOLS AND PROCESSES:

[paste]

Common mistakes

Pre-ticking the consent box for convenience.

Feeding raw customer records into an outside AI tool without knowing what that tool stores.

Forgetting that staff use personal AI tools outside the system, where no one has control.

Chapter 11 checklist

- There is a data map, including AI tools.
- Every form has a consent box that is not pre-ticked and is separated by purpose.
- There is one function for "has refused," and every messaging channel checks it.
- Identifying data is masked before entering a model.
- There is a commit guard for personal data and secret keys.

- Someone is responsible for personal data.

CHAPTER 12: AI PERMISSIONS, MONEY AND THE CODE REPOSITORY

The story: a bulk-commit command and customer data on the repository

Once, a working session with a coding assistant (Claude Code) ran a familiar command to gather every change into one save: `git add -A`. A bulk-commit command like that once put customer data onto the repository. We discovered it, removed it, and built an automatic guard.

I took a very real shock: one line of command, run by a machine, took customer data out of the house. That was when I understood I was not allowed to take this lightly. What made me think longer was that we already had a rule forbidding that command, properly written in the handbook. A rule written in words does not block a command. After the incident, we plugged in a guard at commit time: it refuses a commit if there are five or more different phone numbers, or a secret key, or a secret configuration file. To get past it you must type a deliberately long command, so it does not become a habit.

Not long after, a related warning came: in one night, four deployments of a new version to the live environment by one session also pushed out the unfinished work of three other sessions, because that command pushes the whole working folder and not only the committed part. Nobody knew what the live version contained.

Principle 1: money is an automation-free zone

HOPE's auto-run advertising mode is off by default: the machine only measures performance and recommends, and it is turned on only when I say so explicitly. The daily budget ceiling is a configuration number that clamps every budget increase, manual or automatic, in exactly one place in the codebase; we do not patch the ceiling at each caller. The button to change money exists only in my private chat; the ad chat group that includes staff receives only reports and creative ideas. The machine does not create ads by itself; a human posts them by hand and then tells the machine. AI has no authority to transfer funds. When a safety system blocks code related to automatic money, it is a guardrail, not an obstacle to get around; only changes that reduce risk, such as lowering the ceiling or tightening the stop-loss threshold, should be let through.

Principle 2: the code repository has rules like a cash vault

A standing rule set for every person and every assistant working on the repository: only deploy a new version to the live environment from a commit that has been pushed; immediately before the command, fetch the latest, have no uncommitted changes (including untracked files), and have the commit sit in the main branch; fail one condition and do not deploy, and there is no bypass flag. Do not delete other people's things, do not force-push, do not hard-reset, do not stash; one separate working folder per task. Do not print secrets or personal data anywhere. Do not message customers or staff without explicit permission, and do not draft text going to customers, only copy verbatim text that has been approved. If something breaks after deployment, return to the previous version, then fix it with a new commit.

Principle 3: words do not block a command, structure does

Every important rule needs a mechanism: the commit guard, the script that forces you to read the status before deploying, the function that clamps the budget ceiling, the banned-word gate. A rule that exists only in a document counts as not existing.

Step-by-step method

Step 1: List everything your AI or coding assistant can touch: money, customer data, the repository, channels that publish to the public, infrastructure.

Step 2: For each item, decide three levels: read, propose, execute. The default for AI is read and propose.

Step 3: For each execution level granted, set a structural limit: a ceiling, an approver, a single place.

Step 4: Write the standing rules into an instruction file that the coding assistant reads at the start of each session. Keep it short.

Step 5: Give each rule a blocking mechanism.

Step 6: Each quarter, simulate violating one rule to see whether the mechanism blocks it.

Sample prompt: a standing-rules file for a coding assistant

STANDING RULES (read at the start of each session):

1. Do not gather all changes with a blanket command. Add only the files you edited yourself, one by one. Read the status before committing.
2. Do not deploy a new version to live unless: you have fetched the latest, there are no uncommitted changes, and the commit has been pushed to the main branch.
3. Do not delete, force-push, hard-reset or stash.
4. Do not print secrets, phone numbers, emails or customer names anywhere. Report counts instead of records.
5. Do not send messages to customers or staff. Do not draft text going to customers; only copy approved text.
6. Do not create new data tables or run structural changes; leave a proposal file and write "awaiting approval".
7. At the end of the task report: DONE / PARTLY DONE / BLOCKED, with how to re-check. When a rule blocks you, state which rule. Do not look for a workaround.

Common mistakes

Writing rules with no mechanism.

Giving AI write access because it is "convenient."

Verifying with the assistant's own report instead of running it for real.

Chapter 12 checklist

- Auto-run advertising is off by default, with a ceiling clamped in one place, and the money button only in the private chat of the person with authority.
- Everything AI touches has a level: read, propose, execute.
- There is a commit guard and a process for deploying to live.
- The coding assistant reads the standing rules every session.
- Every rule has a blocking mechanism, and violation has been tested.

PART 5: AI FOR FUNDRAISING

CHAPTER 13: ONE SET OF NUMBERS FOR EVERY FORM

The story: two market-size figures more than ten times apart

When I began fundraising, I submitted applications to many places: incubator programs, startup competitions, funds, and a data file for investors. Each one asked nearly the same questions in slightly different ways. I answered each by hand, at different times, from different files.

One review showed me the price of this approach. The market size in the presentation and in the Q&A document differed by more than ten times. The cap table and the employee stock option pool had two ways of being written. Three financial scenarios in three files did not match. Every number had a reason at the time of writing. But placed side by side, a sharp investor would ask at once: which number is yours?

What I had not anticipated: investors now use AI to read applications. They paste the presentation, the data file and the forms into an AI assistant and ask, "where do these documents contradict each other?" A contradiction that once needed a careful human reader to spot now shows up in seconds.

Principle 1: every number has exactly one home

We built an answer store for investors. Each answer has these fields: identifier, group, question, answer, source, approval date, status, old version, and English version. Status has four levels: finalized, submitted, needs update, needs text. The store holds nearly eighty answers. Each new form is answered by pulling from the store, not by writing from scratch. When a number changes, we fix it only in the store, and later forms are automatically right. For forms already submitted, the store keeps the old version so we know what we said to whom.

The set of numbers finalized on October 10, 2026 comprises a first round of USD 250,000 for 20% equity, net revenue for the first nine months of the year of about VND 3.57 billion, 1,294 customers buying from March to September, and the positioning as a health and beauty care ecosystem from the inside for Vietnamese women. I list these so you can see the shape of a number set: few, clear, sourced, dated.

Principle 2: publish only reconciled numbers

I publish only 2026 revenue, because that is the period with a reconciliation ledger across orders, waybills and bank statements. For earlier years I have no reconciliation ledger, so I do not publish, and the store states clearly that these are not published. Likewise, I cannot measure long-term retention because there is no reconciled 2025 data, and I say so plainly. A good investor respects "I have not measured this yet" more than a pretty number with no foundation.

Principle 3: AI is part of the story, and investors will ask

For a company run on AI, investors will ask six things. I prepared each answer in advance and they sit in the store:

Where does the company use AI. Who approves AI output before it has real effects. How is customer data protected. How is AI quality measured. Does AI spend money by itself. How do you ensure AI-generated content complies with advertising rules.

My answers are all mechanisms already described in this book: AI reads wide and writes narrow; customer-facing content is drafted by machine and approved by a human; phone numbers and emails are masked before entering AI; each machine takes a weekly exam; the auto-run advertising mode is off by default and has a ceiling; all content passes through a single banned-word gate. This is where operating discipline becomes a fundraising advantage. A company that only says "we use AI" is no different from anyone else. A company that can answer the six questions above with real mechanisms is different.

Principle 4: write for two readers, a person and a machine

Your application will be read by a person, and possibly summarized by a machine first. So every number comes with a date and a source, terms are defined once and used consistently, answers open with the main point, and no concept carries two names. What helps a machine summarize correctly also helps a human read faster. AI only helps arrange and check, not inflate: I cross-check every number myself, and I have never sent out a number whose source I could not point to.

Step-by-step method

Step 1: Gather every form, presentation and data file you have sent and will send.

Step 2: Extract a list of every number that appears, with where it appears. Look for mismatches.

Step 3: For each mismatched number, decide the right figure and trace it back to the books. Record the source and approval date.

Step 4: Build the answer store with the fields above. Assign statuses.

Step 5: For each new form, pull from the store, and where new text is needed, mark it "needs text."

Step 6: Before submitting, run an AI in the role of an investor probing for contradictions.

Sample prompt: probe for contradictions like an investor

You play a cautious investor using AI to review an application.

I will paste several documents from the same company (presentation, data file, forms).

Please:

1. Build a table of every number that recurs across the documents, with which document says what.
2. Point out every mismatch, however small (unit, measurement period, definition).
3. Point out statements with no source or date.
4. List the 10 deepest questions you would ask the founder based on the mismatches or gaps.
5. Mark the numbers you suspect have not been reconciled.

Do not infer any number that is not in the documents.

Common mistakes

Answering each form from scratch.

Publishing unreconciled numbers "to look complete."

Using AI to beautify instead of to check.

Giving two names to the same metric.

Chapter 13 checklist

- There is an answer store with source, approval date and status for every answer.
- Every number appearing in multiple documents has been cross-checked.
- Only reconciled numbers are published, and unmeasured ones are stated.
- The six answers about AI are prepared.
- An AI has played the investor and probed for contradictions before submission.

PART 6: THE 90-DAY ROADMAP

CHAPTER 14: A 90-DAY ROADMAP TO BUILD YOUR OWN OPERATING BRAIN

The story: I did not build it in 90 days, and you do not need to

HOPE's system did not come from a neat plan. It grew through many incidents, layer by layer. If I could start over, I would not try to build everything at once. I would follow the order that the incidents taught me: correct data first, watchdog next, then customer care, content, bots.

The 90-day roadmap below is that order, condensed for a health D2C founder just starting out, without an engineering team.

Principle: every phase has a pass-through criterion

Without a pass-through criterion, any phase can drag on forever. Each phase below has one thing that must run for real, and one thing you must check by hand.

Phase 1: days 1 to 14, the map and the source of truth

What to do: the work map in three layers, decide, do, check (Chapter 1). List every place that fetches revenue, choose the primary source, record who enters it and the delay (Chapter 4). Choose a first bottleneck.

Pass when: you can answer "what is this month's revenue and where does the number come from" from one source, with freshness attached.

Phase 2: days 15 to 30, the revenue door and the watchdog

What to do: build the single revenue door, three layers, with warning flags. Each data source has a last-success timestamp. The blank-scrape-stops rule. A morning report sent to yourself, with a freshness line.

Pass when: you switch off a source as a test and receive an alert turned into a task with an owner within one day.

Phase 3: days 31 to 45, customers and care

What to do: each customer has exactly one holder. Measure "cared for" by real messages. Named reminders by working hours. A care sequence around product lifecycle milestones, placed next to the customer (Chapter 5).

Pass when: you know how many customers are waiting for a reply and how long the longest case has waited, with real figures.

Phase 4: days 46 to 60, legal guardrails and personal data

What to do: a product positioning page, a banned-word list confirmed by legal counsel, a single check gate, a scan of the old content stock (Chapter 10). A data map, forms with consent, the "has refused" function, the commit guard (Chapters 11 and 12).

Pass when: a test case with a real banned word is blocked, a test case with a valid disclaimer is not blocked, and a customer who has refused receives no messages.

Phase 5: days 61 to 75, content

What to do: the brand DNA set, the campaign calendar, writing one post at a time through the gate and a human approver. A quota of one post a day is enough. Start the blog by buying intent, at a low rhythm, with a cap (Chapter 8).

Pass when: seven days in a row have approved posts without you having to nudge the machine.

Phase 6: days 76 to 90, the advisor bot, exam sets, fundraising numbers

What to do: an advisor bot that reads live figures, with root instructions that include pushback and recommendations, permissions by place of conversation (Chapter 6). Ten to twenty real cases become an exam set. An investor answer store with one set of numbers (Chapter 13).

Pass when: the bot rejects a wrong proposal of yours with figures, and an outsider who asks the bot about salary or cost of goods is refused.

On time and cost

I give no general cost figure because it depends on the tools you choose. Start with tools you already have and a low-tier plan, measure the real cost each month, and upgrade only when the numbers show it is needed. A weekly or monthly AI usage limit is a more real constraint than money. Set aside about one hour a day in the early phases, at a fixed time, because work without a time slot always loses to urgent work.

Sample prompt: a 90-day plan for your business

I run a D2C dietary supplement business. Scale: [number of staff, monthly revenue, main sales channels].

Based on the six-phase roadmap below, adjust it for my business:

1) map and source of truth, 2) revenue door and watchdog, 3) customers and care, 4) legal and personal data, 5) content, 6) advisor bot and fundraising numbers.

For each phase, state: specific work by week, what I must do by hand, what can be delegated to AI, a measurable pass-through criterion, and the biggest risk.

Point out which phase I should do first or skip, based on the information I provided. Ask me if important information is missing.

Common mistakes

Starting with the bot because it looks the most attractive. A bot stands on data, it cannot replace data.

Skipping the legal phase to "do later." Legal risk multiplies with the number of posts.

Doing everything at once. One phase at a time, with a pass-through criterion.

Chapter 14 checklist

- I have chosen the phase order that fits my business.
- Each phase has a measurable pass-through criterion.
- I have a fixed time slot every day.
- I measure real tool costs each month.
- I have someone with legal expertise to ask in Phase 4.

CHAPTER 15: FOR HEALTH PROFESSIONALS WITH A COMMUNITY

The story: a big community does not mean a talking community

This chapter is written for a group of people I meet often: nutritionists, macrobiotic practitioners, health coaches, people with a large community that listens to them every day. My co-founder, a macrobiotic expert who leads HOPE's community, is the person I think of most while writing.

What experts often face: the demand for content far exceeds one person's time. Today a post is needed, tomorrow a video, the day after replies to hundreds of comments. The temptation is to ask AI to write on your behalf and post more. Chapter 9 showed that in large groups, even when we post steadily five to six times a day, customers stay silent.

A different way: use AI to multiply the expert's real voice, not to replace the expert.

Principle 1: the expert is the decide layer, AI is the do layer

The most valuable part of the expert is judgment and the trust the community places in it. This part is not handed to a machine. What the machine can do: turn what the expert has already said into many different formats.

A one-hour live session can become a long post, a short series, a summary for the group, and a list of frequently asked questions. The expert speaks once, the machine adapts, the expert approves. What matters is that the raw material is real. AI writing from a real insight of the expert produces a post with soul. AI inventing topics by itself produces generic posts.

Principle 2: the boundary between nutrition education and medical advice

This is the point health professionals with a community most easily cross, because people ask about their own illnesses every day. The rules I apply:

Educational content talks about habits, food and lifestyle at a general level.

When someone asks about a specific disease, medication they are taking, pregnancy, small children, or a symptom, the answer from the expert, or from the drafting machine, must advise them to see a doctor. The machine must flag these questions for a human to see.

Do not tell customers' cases as proof. Do not use before-and-after cases.

The machine never answers in the expert's place for questions in this group.

Principle 3: transparency and small warm groups

Hold three things with the community: only posts the expert has approved carry the expert's name; admit AI assistance if the community asks, because hiding it will sooner or later be discovered; and never let AI put words in the expert's mouth that the expert has never said, because a quote invented by a machine is a risk to both reputation and legal standing. From the figures in Chapter 9, small groups reach a speaking rate of 11.58% and large groups 1.59%. Create small groups around a specific goal, with one person in charge, question-asking content, and let AI handle the repetition: schedule reminders, summaries, drafts.

Step-by-step method

Step 1: Record or transcribe your real sharing sessions. This is your raw material store.

Step 2: Write a voice file: how you address people, words you often use, words you never use, and the three example posts you like best.

Step 3: List the types of questions and decide which types the machine may draft, which only a human answers, and which must be referred to a doctor.

Step 4: The process: you speak, the machine adapts, the banned-word gate, you approve, publish.

Step 5: Build a personal banned-word gate with the guidance of a lawyer or legal counsel.

Step 6: Run one small pilot group, with one holder, and measure the speaking rate after four weeks.

Sample prompt: adapt a sharing session

I am a nutrition professional with a community. Below is a transcript of one of my real sharing sessions, and my voice file.

Please adapt it into:

1. One long post of about 600 words.
2. Five short passages to post separately.
3. A list of 8 frequently asked questions with answers, using only what I said in the session.

Strict rules:

- Do not add any claim, figure or quote that I did not say. Where something needs to be added, write [I NEED TO CONFIRM].
- Speak only of dietary supplements and lifestyle at an educational level. No medical effects, no promise of results by number of days, no customer cases.
- Voice follows the voice file. Promotions are gifts only.
- At the end, list the sentences you think are risky so I can review them.

VOICE FILE:

[paste]

TRANSCRIPT:

[paste]

Common mistakes

Letting the machine write new topics instead of adapting what you have said.

Answering comments about illness with a machine.

Chapter 15 checklist

- I have a store of original transcripts and a voice file.
- I have classified questions: machine drafts, human only, refer to a doctor.
- Every post carrying my name has been approved by me.
- I have a banned-word gate confirmed by someone with legal expertise.
- I am piloting a small group and measuring the speaking rate.

CLOSING

If I had to pack the whole book into one sentence, I would say: AI multiplies everything, so the job of the operator is to make sure what is multiplied is the truth, the law, and kindness to customers.

I hold the role of Chief AI Officer at HOPE, alongside the role of founder. That role for a health D2C business, as I understand it and do it every day, is not the person who is best at models. It is the person who knows which numbers deserve trust, knows how to chain tools into an assembly line, and dares to set limits on machines before an incident sets them. These three jobs do not require a technology degree. They require discipline, and the willingness to retell your own mistakes.

If you can build one source of truth for revenue, one check gate for content, and a watchdog that knows to shout when something breaks, you have done the hardest part. The rest is repetition: do, measure, learn.

I wish you success in building your own operating brain.

APPENDIX A: PROMPT LIBRARY

Each full prompt sits in its chapter. The table below is an index, with two additional prompts.

Task	Chapter
Map work in three layers: decide, do, check	1
Design a DO, MEASURE, LEARN loop	2
Diagnose an organization with the six pieces	3
Build a single revenue door	4
Draft a reply for staff to approve	5
Root instructions for an operations advisor	6
Review an automated machine against the "failure must be loud" rule	7
Brief to write one grounded post	8
Analyze community vitality	9
Review a page against the legal guardrails	10
Map personal data	11
Standing-rules file for a coding assistant	12
Probe for contradictions like an investor	13
90-day plan	14
Adapt an expert's sharing session	15

Additional prompt 1: summarize the lesson after each loop

Below are the expectation I recorded before the work and the real result. Write a "lesson" paragraph of no more than 120 words containing: what was right, what was wrong, what is still unknown, and one concrete change for the next round. Do not embellish the result. If the data is not enough to conclude, say so clearly.

EXPECTATION: [..] RESULT: [..]

Additional prompt 2: trace the source of a number

I will give you one number from my report. Help me trace it back: (1) which sources this number could come from, (2) who enters each source and how late it is, (3) what I need to check to be sure it is right, (4) the ways it could be wrong in a convincing way. Do not conclude yet whether it is right or wrong.

APPENDIX B: TOOL TABLE

This table describes the role of each kind of tool in my system. I give no prices because prices change and depend on the plan.

Tool type	Role	Notes
Conversational AI assistant: Claude (Anthropic) as the main base, Gemini (Google) as a supplement	Strategic thinking, analysis, drafting	Used for work that needs judgment and back-and-forth iteration
Coding assistant in the terminal (Claude Code)	Build and fix internal apps, scripts, checks	Needs standing rules and guards (Chapter 12)
AI models called inside products (Claude family: Opus, Sonnet, Haiku)	Advisor, replies, bulk grading	Choose the model by task: strategic questions use a strong model, routine work a mid-size model, bulk grading a light model
Database and authentication	Store orders, customers, configuration, logs	Small data can use a single configuration table
Deployment platform	Run the app and the automated schedules	Deploy only from a pushed commit
Messaging app with bots	24/7 advisor, reports, approval buttons	Permissions by place of conversation
Customer care and conversation software	The real message source for measurement	Measure real behavior, not button flags
Carrier	Order reconciliation layer	Orders prepaid by transfer have zero collected amount
Payment gateway	Revenue source for the web channel	Confirms payments by itself

APPENDIX C: GLOSSARY

AI reads wide, writes narrow: a machine may read many sources but may only change the real world within a narrow scope, with a human approving.

Exam set: a set of real cases with expected answers, run periodically to grade an AI machine.

Blank scrape: a data fetch that returns empty.

Chief AI Officer: the person accountable for how a business uses AI, including value, safety and discipline.

Banned-word gate: an automatic check step that blocks violating words and sentence patterns before content goes out.

Single door: one single entry point for one kind of data or action.

D2C: direct to consumer.

Dataroom: a permissioned document store for investors to review.

GEO: optimizing content so AI answer engines can cite it easily.

Failure must be loud: every error must leave a trace and be reported, never silent.

Six-piece model: Vision, Strategy, Capability, Reward, Resources, Urgency.

Heartbeat: a timestamp a machine writes periodically, even when there is no work, to show it is still alive.

Retrieval: the step of finding relevant passages of knowledge to give the model before it answers.

Patch the jar before pouring water: fix the leaks in the funnel before increasing the number of customers sent in.

ABOUT THE AUTHOR

Nguyen Van Hung studied Information Science (software engineering) at Sojo University in Kumamoto, Japan. He received a scholarship every year, graduated with an outstanding thesis, and was the only Vietnamese student in a department of more than 150.

He has more than 12 years of experience in Japanese companies, at FPT Software Japan and Scuti Japan. More than 7 of those years were as a Bridge Systems Engineer (BrSE), after which he moved into a business management role. He is currently pursuing a Master's degree in Economic Law.

On August 23, 2023, he founded HOPE CORP, the company developing the IKI Healing brand at ikihealing.com. He is Founder, Chairman of the Board, and CEO/COO of the company. He personally leads research and development (R&D), running the business on AI, and fundraising. HOPE is currently raising a Pre-seed round.

Contact: contact@ikihealing.com. Information for investors and partners: ikihealing.com/founders/.

This book is free. You may share it verbatim with attribution. All feedback and challenge, especially where you think I got something wrong, is very welcome.