THE

PROMPT GOVERNOR

How to Control AI Output Without Surrendering Judgment

A Practical Playbook for Professionals

By Leonardo Pasqualin

Contents

Who This Playbook Is For — And Who It Is Not

This playbook is written for people who use AI in real work and care about outcomes.

It assumes you are not experimenting for novelty, entertainment, or hype. You are producing things that matter—documents, decisions, communications, analyses, or systems that other people rely on.

If that describes you, this book is meant to raise your standard.

This Playbook Is For You If:

- You use AI regularly in professional or operational contexts
- You are responsible for the accuracy, quality, or consequences of AI-assisted work
- You are tired of confident nonsense and output that sounds right but isn't trustworthy
- You want predictable, reviewable results—not clever surprises
- You are building workflows, processes, or agents and need control, not vibes
- You understand that delegation does not remove accountability

This playbook is especially relevant if you are an operator, manager, consultant, engineer, analyst, or decision-maker who cannot afford silent failure.

This Playbook Is Not For You If:

- You are looking for viral prompt tricks or clever phrasing hacks
- You want the "best prompt" for the current model of the month
- You expect AI to replace judgment instead of supporting it
- You are uncomfortable taking responsibility for AI-assisted output
- You want automation without specification
- You believe "the AI said so" is an acceptable explanation

This is not a guide to sounding smarter, faster, or more creative. It is a guide to staying in control.

A Note on Expectations

This book will not make AI reliable. No prompt can do that.

What it will do is make unreliability visible, bounded, and governable. It will help you specify clearly, diagnose failures quickly, and decide when automation is appropriate—and when it is not.

If you are willing to retain judgment, apply discipline, and own outcomes, this playbook will give you leverage. If you are looking to offload responsibility, it will feel restrictive.

That reaction is the point.

Introduction

THIS BOOK IS ABOUT CONTROL

This is not a book about "prompt tricks." It's a book about control. If you use AI occasionally, this book will make you more consistent. If you use AI daily, this book will keep you out of trouble. If you're building workflows or agents, this book is the difference between leverage and silent failure.

This playbook is for people who already use AI in real work and are tired of inconsistent results, confident nonsense, and output they can't quite trust. It assumes you care about correctness, responsibility, and outcomes—not novelty.

This is not a guide to getting AI to "sound smarter." It won't teach you clever phrasing or viral prompt hacks. It won't tell you which model is best this month. Those things change too fast to matter.

What doesn't change is this: probabilistic systems require structure. If you don't specify clearly, the system will guess. And if you automate guessing, you automate risk.

Everything in this book treats prompts as specifications, not conversations. You'll learn how to define intent, constrain behavior, surface uncertainty, and retain authority—whether you're writing a single email or orchestrating an agentic workflow.

HOW TO USE THIS BOOK

Read it once end to end. Then keep it open while you work. Apply the framework to real tasks, not examples. When output fails, diagnose the spec instead of blaming the model.

If you want AI to think for you, this book isn't for you.

If you want AI to work for you—without surrendering judgment—keep reading.

Chapter 1: Why Prompting Matters

Most people blame the AI when they get bad results. They assume the model isn't smart enough, or they need a better subscription, or the technology just isn't there yet. In reality, the problem is almost always the input.

A $20/month ChatGPT subscription and a free Claude account will both produce garbage if you give them vague instructions. The model matters far less than people think. Prompting is the only variable you actually control.

> **THE CORE TRUTH**
>
> Better prompts equal better results. A mediocre model with a great prompt will beat a frontier model with a lazy prompt almost every time.

Here's the difference in practice:

■ Vague	✓ Specific
Write me a marketing email	`Write a 150-word email to lapsed customers of a B2B SaaS product, tone: direct and slightly urgent, goal: get them to book a 15-minute call, include one specific pain point about manual data entry.`

The difference between those two prompts is about 30 seconds of thinking. The difference in output quality is enormous.

Prompting is a skill, not a personality trait. It can be learned, practiced, and systematized. The people getting real value from AI aren't smarter than you. They've just learned to specify what they actually want.

The Real Failure Mode

The most dangerous failure with AI is not bad output. It's misplaced authority.

People don't just want AI to draft faster—they want relief from judgment. Confident-sounding output feels like competence, and that confidence tempts users to stop questioning, stop verifying, and stop owning the result.

LLMs do not understand truth, risk, or consequence. They generate plausible text, not accountable decisions. When users treat AI output as authoritative instead of provisional, they quietly transfer judgment to a probabilistic system that cannot bear it.

Prompt discipline exists to prevent this transfer.

KEY TAKEAWAYS

- The model matters less than the prompt
- Prompting is the only variable you control
- 30 seconds of specification saves 30 minutes of revision
- This skill can be systematized and learned

Chapter 2: What an LLM Actually Is

An LLM predicts the next word—technically, the next token—based on patterns learned from massive amounts of text. That's it. Everything else you see—the apparent knowledge, the conversational tone, the occasional brilliance—emerges from that single mechanism.

CRITICAL UNDERSTANDING

LLMs do not "know things." They have learned statistical associations between words. Confidence comes from probability, not from verification. A plausible-sounding wrong answer and a correct answer feel exactly the same to the model.

This is why LLMs confidently generate bullshit. The model has no access to real-time information unless you explicitly connect it to tools. It has no way to check its own answers. It has no concept of "true" versus "false"—only "probable" versus "improbable."

PRACTICAL IMPLICATIONS

Every LLM output should be treated as a draft, not a fact. Verification is your job. The model performs best when you give it clear patterns to follow; vague inputs leave room for plausible-but-wrong completions.

LLMs are excellent at transformation, summarization, and generation within constraints. They are unreliable for factual recall, especially for obscure or recent information. Once you understand this, you stop being surprised by failures and start designing prompts that play to the model's strengths.

KEY TAKEAWAYS

- LLMs predict probable text, not true text
- Confidence ≠ Correctness
- Treat all output as drafts requiring verification
- Play to strengths: transformation, generation within constraints

Chapter 3: Prompting Is Programming

When you write a prompt, you're not "talking to AI." You're writing a program in natural language.

Core Principle: A Prompt Is a Specification

A prompt defines behavior. It constrains action. It determines outputs. That makes it a specification.

This is not a metaphor. It is operationally true.

A good prompt does the same job a good spec does in software, operations, or law: it removes ambiguity. A bad prompt does what a bad spec always does—it leaves gaps that get filled by assumptions. In probabilistic systems, assumptions are not harmless. They are risk.

A prompt has:

- Inputs (context, constraints, examples)
- Processing logic (the model's pattern matching)
- Outputs (the generated text or action)

That is a program, whether you acknowledge it or not.

■ Vague	✓ Specific
Summarize this article	`Summarize this article in 3 bullet points for a busy executive who cares about revenue impact.`

Good programmers don't write code and hope it works. They define inputs, expected outputs, and edge cases. Good prompters do the same.

THE MENTAL SHIFT

Stop asking, "What do I want to say to the AI?" Start asking, "What specification would produce the output I need?"

This reframing makes prompting:

- **Iterable** — your first prompt is a hypothesis, not a solution
- **Reusable** — effective structures become templates
- **Debuggable** — failures can be traced back to missing constraints

KEY TAKEAWAYS

- Prompts are specifications, not conversations
- Treat prompt writing like programming
- Iterate, template, and debug your prompts
- Vague specs produce unpredictable results

Chapter 4: The Five Pillars of Effective Prompting

Most prompts fail silently. The AI gives you something—it just isn't what you needed. That happens because the model filled in gaps you didn't know you left. The Five Pillars exist to eliminate guessing. Each pillar controls a specific failure mode.

ROLE	CONTEXT	TASK	OUTPUT	GUARDRAILS
Who is the AI?	What background?	What to do?	What format?	What limits?

Pillar 1: ROLE

Controls the model's operational mode—what expertise, perspective, and behavioral constraints it adopts. Without it, the model defaults to "helpful generalist assistant," producing generic, safe, bland output.

■ Vague	✓ Specific
Review this contract clause	`You are an in-house counsel at a SaaS company. Review this contract clause for liability exposure.`

Pillar 2: CONTEXT

Defines the relevant background information the model needs to do the task correctly. Without it, the model makes assumptions that may be wrong, outdated, or irrelevant to your situation.

Pillar 3: TASK

Specifies the executable action the model should take—what "done" looks like. Without it, the model produces vague, hedge-filled, or incomplete output.

■ Vague	✓ Specific
Help me with my presentation	`Write 5 slide titles and one bullet point per slide for a 10-minute investor update. Focus on Q3 revenue growth and product roadmap.`

Pillar 4: OUTPUT SHAPE

Controls format, structure, length, and level of detail. Without it, the model defaults to verbose, essay-style prose. You get walls of text when you needed a table.

■ Vague	✓ Specific
Explain the pros and cons of remote work	`List 3 pros and 3 cons of remote work for a 50-person startup. Use bullet points. Max 15 words per bullet.`

Pillar 5: GUARDRAILS

Defines what the model should NOT do—constraints, exclusions, verification requirements, and conditions for refusal. Without them, the model overreaches, hallucinates, or includes things you explicitly don't want.

■ Vague	✓ Specific
Summarize this research paper	`Summarize this research paper. Do not include information not present in the paper. If the methodology is unclear, say so rather than guessing. Do not editorialize.`

DIAGNOSTIC CHECKLIST

When output fails, ask which pillar you underspecified. The pillars work as a system—a prompt with great ROLE but no GUARDRAILS will still produce hallucinations.

Chapter 5: The Universal Prompt Template

Most prompting failures aren't caused by bad wording. They're caused by missing components. The Universal Prompt Template exists to prevent omission. It maps directly to the Five Pillars and works because it forces you to specify the things the model would otherwise guess.

This is not a clever framework. It's a checklist.

THE UNIVERSAL PROMPT TEMPLATE

```
ROLE:
You are [specific role].
You prioritize [key characteristics].
You are NOT [what to avoid].

CONTEXT:
[Only the background information required for correct execution.]

TASK:
[The precise, executable objective. Define what "done" means.]

OUTPUT:
[Exact format, structure, length, and level of detail.]

GUARDRAILS:
[What the model must not do. Constraints, exclusions, verification
rules.]
```

Why Each Section Exists

ROLE

Controls the model's operating mode. Without it, the model defaults to a generic assistant. Define what the model is AND what it is not.

CONTEXT

Supplies only the information required for correct execution. Missing context forces assumptions. Excess context dilutes focus.

TASK

The most critical section. If you can't state the task clearly enough to hand it to a competent human, you're not ready to prompt.

OUTPUT

Prevents the model from choosing structure for you. "Three bullet points, max 20 words each" beats "keep it brief."

GUARDRAILS

Your safety system. This is where you constrain invention, surface uncertainty, and prevent overreach. If accuracy matters, this section is not optional.

How to Use It

This template is adaptable, not rigid. For quick tasks, compress sections. For complex or high-risk tasks, expand them. Do not skip sections unless you understand the tradeoff.

Templates don't make prompts smarter. They make failures visible—and controllable.

Chapter 6: Iterative Prompting

The fantasy is that you craft one perfect prompt and the AI delivers exactly what you need. The reality is that your first prompt is a hypothesis. You test it, see where it breaks, and fix the spec.

The Iteration Loop

1	2	3	4
Write structured prompt	Evaluate against intent	Identify weak pillar	Tighten constraint

→ → → **REPEAT UNTIL OUTPUT MATCHES SPEC** → → →

Example: Newsletter Summary

ITERATION 1

Prompt: "Summarize this article for my newsletter."
Result: 400-word summary with generic insights—too long, too bland.
Diagnosis: Missing OUTPUT SHAPE for length, weak CONTEXT about audience.

ITERATION 2

Prompt: "Summarize in 3 bullet points for startup founders. Focus on actionable takeaways."
Result: Three bullets, but surface-level.
Diagnosis: TASK still too vague—what counts as "actionable"?

ITERATION 3 ✓

Prompt: "Summarize in 3 bullets. Each bullet: one thing a founder could implement this week. No generic advice. If no implementable insights exist, say so."
Result: Three specific, actionable bullets. One flags an abstract section. Usable.

The skill isn't prompt cleverness. It's diagnosis. Three iterations taking two minutes total will outperform twenty minutes of trying to write the "perfect" prompt upfront.

End-to-End Example: From Vague to Governed

The following example shows what iterative prompting looks like when the output is intended for reuse or automation—not just one-off drafting.

Initial Prompt (Unspecified)

Prompt: "Review this contract clause and tell me if there are any risks."

Result: A confident, high-level response with general legal commentary. It sounds plausible, but mixes factual interpretation with assumptions.

Why this is dangerous: The output reads authoritative. A non-lawyer could easily treat it as advice. This is not just low quality—it's ungoverned.

Iteration 1: Structured But Still Weak

Prompt: "You are a legal expert. Review this contract clause and identify potential risks for our company."

Result: More focused, but still overconfident. The model invents intent and speculates about enforceability.

Diagnosis: ROLE improved but vague. TASK still ambiguous. GUARDRAILS still missing.

Iteration 2: Governed Specification ✓

```
"You are an in-house counsel at a mid-size SaaS company.

Context: This clause is from a customer agreement under U.S. law.

Task: Identify potential liability or ambiguity explicitly present in the clause text.

Output: Bullet list with one sentence per item.

Guardrails: Do not speculate beyond the clause text. If a risk depends on missing
information, state the dependency explicitly. If no clear risks are present, say so."
```

Result: A short list identifying two concrete issues, each tied directly to the clause language. One item explicitly states that enforceability depends on jurisdictional details not provided.

This output is now safe to reuse in a workflow with human review. It is no longer pretending to be authority—it is producing structured input for judgment.

KEY TAKEAWAYS

- Your first prompt is a hypothesis, not a solution
- Diagnose which pillar failed before changing wording
- Iteration improves specifications, not models
- 2–3 iterations is normal; 4–5 means the task is too complex
- Governed prompts make uncertainty visible and reusable

Chapter 7: High-Leverage Prompt Patterns

Prompt patterns are reusable structural fixes for recurring failure modes. They are not magic formulas. They do not make bad specifications good. They work only after you've diagnosed why the output failed.

Role Anchoring → Fixes Generic Output

Failure symptom: Output is safe, bland, non-committal, or refuses to take a position.

Pattern: "You are [specific expert]. You prioritize [specific value]. You are NOT a general assistant."

Why it works: It forces the model out of default assistant mode and constrains perspective.

Constraint Stacking → Fixes Rambling

Failure symptom: Output is directionally correct but too long, unfocused, or poorly structured.

Pattern: Explicitly stack constraints on length, format, quantity, and exclusions.

Why it works: LLMs optimize for plausibility, not brevity. Multiple constraints reduce degrees of freedom.

Negative Specification → Fixes Unwanted Content

Failure symptom: The model keeps including things you explicitly don't want.

Pattern: "Do NOT [behavior]. Exclude [category]. Never [action]."

Why it works: Positive instructions describe what can happen. Negative specifications define hard boundaries.

Example Injection → Fixes Misaligned Tone/Format

Failure symptom: The model consistently misinterprets what "good" looks like.

Pattern: Provide 1–2 examples of desired output—or explicit anti-examples.

Why it works: Examples collapse ambiguity faster than abstract description.

Forced Structure → Fixes Buried Answers

Failure symptom: The correct answer exists but is hidden inside long prose.

Pattern: Specify exact sections, headers, or a response skeleton.

Why it works: Structure forces prioritization and makes outputs predictable and reviewable.

Uncertainty Surfacing → Fixes Overconfidence

Failure symptom: Output sounds confident even when information is incomplete or inferred.

Pattern: "If unsure, say so explicitly. Flag assumptions. Distinguish facts from inferences."

Why it works: LLMs do not self-correct. This pattern makes uncertainty visible.

Task Decomposition → Fixes Complex Task Confusion

Failure symptom: The prompt attempts to do too much and the output loses coherence.

Pattern: Break the task into explicit, numbered sub-steps within the prompt.

Why it works: Large, implicit tasks overload the model's reasoning. Decomposition restores alignment.

Using Patterns Correctly

Patterns are composable, but diagnosis comes first. One prompt may use several patterns. No prompt should use patterns blindly. When output fails, ask: What is the symptom? Which pillar is weak? Which pattern addresses that failure mode?

Patterns don't fix thinking. They encode it.

Chapter 8: Common Failure Modes

When AI output fails, it's almost always because a pillar is missing or weak. Learn to trace symptoms back to causes. Don't blame the model—diagnose the spec.

Failure Mode Diagnostic Table

SYMPTOM	LIKELY CAUSE	FIX
Vagueness	Weak TASK / missing CONTEXT	Add specifics and situational context
Overconfidence	Missing GUARDRAILS	Add uncertainty flagging instructions
Hallucination	No source constraints	Restrict to provided documents only
Rambling	Missing OUTPUT SHAPE	Add explicit length/format constraints
Wrong detail level	Weak CONTEXT	Specify audience and depth
Off-topic drift	Weak TASK	Tighten scope, exclude adjacent topics
Generic output	Missing CONTEXT/ROLE	Add concrete context, strengthen role

When output fails, run through this table. Ask: Which symptom am I seeing? Which pillar is likely weak? Fixing the wrong pillar wastes iterations. Accurate diagnosis is the skill that makes prompting fast.

Chapter 9: Reducing Errors and Hallucinations

CORE PRINCIPLE

Errors cannot be eliminated—only reduced. Your job is to design prompts that surface uncertainty, constrain invention, and make review efficient.

Strategies for Error Reduction

SURFACE ASSUMPTIONS

Guardrail: "Before answering, list the assumptions you're making. If any assumption is wrong, your answer may not apply."

REQUEST CONFIDENCE SIGNALS

Guardrail: "For each claim, indicate whether it's based on (a) provided text, (b) general knowledge, or (c) inference that may be wrong."

SEPARATE PROPOSAL FROM EXECUTION

Workflow: Ask for a plan or draft first. Review it, correct errors, then prompt for execution.

CONSTRAIN THE SOURCE

Guardrail: "Only use information from the attached document. If the answer isn't there, say 'Not found in document.'"

REQUEST CITATION

Guardrail: "For each factual claim, include a direct quote from the source. If no supporting quote exists, omit the claim."

DESIGN FOR EFFICIENT REVIEW

Output shape: "Present findings in a table with columns: Claim, Source, Confidence Level."

None of these strategies guarantee correctness. They shift the burden from "trust the AI" to "make verification tractable." The goal is not to make the AI reliable. The goal is to make unreliability visible.

BOTTOM LINE

If errors are unacceptable, you need human verification. Full stop. No prompt engineering changes that.

Chapter 10: Why Agents Fail Without Prompt Discipline

> **CRITICAL WARNING**
>
> Agentic systems don't eliminate prompt problems—they multiply them. Every flaw in your prompt specification compounds with each autonomous step.

An "agent" is not a smarter model. It is a loop: prompt → output → action → new prompt → repeat. The model does not become more intelligent or more careful because it is running autonomously. Nothing about that loop improves truthfulness, judgment, or accountability.

In a single prompt, a weak guardrail might produce one hallucination you catch. In an agent running twenty steps, that same weakness creates twenty opportunities for error—most of which you will never see until damage is done.

Example Failures

> **SILENT DATA CORRUPTION**
>
> An agent researches competitors and updates a spreadsheet. The prompt lacks a constraint against inventing data. On step fourteen, it fabricates a revenue figure. Your sales team uses it in a pitch.

> **HALLUCINATION PROPAGATION**
>
> An email-drafting agent extracts "key points" and then drafts a message. If the extraction step hallucinates a claim, the final email presents that hallucination confidently to your client.

> **UNDEFINED OPERATIONS**
>
> An agent told to "clean up the CRM by removing duplicates" deletes records sharing an email address but belonging to different people. You lose customer data.

The failure mode isn't "the agent went rogue." It's mundane: the agent did exactly what poorly specified prompts told it to do, at scale, without human checkpoints.

The Automation Gate

Before you turn a prompt into an agent, workflow, or autonomous loop, every condition below must be true:

- ■ I can run this task manually and get reliable, repeatable results
- ■ All five pillars are explicit—not implied

- ■ The output format is predictable and easy to review
- ■ Uncertainty and assumptions are surfaced, not hidden
- ■ There are human review or approval checkpoints
- ■ Actions can be stopped, corrected, or reversed
- ■ I know exactly what the agent is not allowed to do

If any box is unchecked, automation is premature.

KEY TAKEAWAYS

- Agents multiply prompt flaws; they do not fix them
- Missing guardrails become catastrophic at scale
- Checkpoints are mandatory, not optional
- If single prompts aren't reliable, don't build agents yet

Chapter 11: You Are the Governor, Not the AI

ACCOUNTABILITY

The human is always accountable for outcomes. "The AI said so" is not a defense—legally, professionally, or ethically.

When you use AI to produce work, you are responsible for that work. Your client doesn't have a contract with OpenAI. Your employer didn't delegate authority to Claude. You made the decision to use the tool. You own the result.

A consultant uses AI to draft a market analysis. The analysis includes a fabricated statistic. The client makes a decision based on that statistic. When it turns out to be wrong, the client doesn't sue the AI vendor—they sue the consultant.

Real Risks

LEGAL EXPOSURE

Contracts you sign, advice you give, statements you publish—these carry liability regardless of how they were produced.

REPUTATIONAL RISK

If your AI-assisted output is wrong, embarrassing, or harmful, the damage attaches to you, not to a model.

OPERATIONAL RISK

Errors can cascade into corrupted data, missed deadlines, or broken customer relationships.

Governance is not passive. It's not enough to "use AI responsibly" in some vague sense. Governance means review before action, verification before publication, and override when output is wrong.

Delegation is not abdication. You can delegate drafting, research, and synthesis to AI. You cannot delegate judgment, accountability, or final authority.

THE PROFESSIONAL STANDARD

If you wouldn't sign your name to it without reading it, don't publish it, send it, or act on it.

Chapter 12: How to Apply This Playbook

Start small. Add structure incrementally. Resist the urge to automate before you can specify. This playbook is a foundation, not a finish line.

Implementation Path

STEP 1: SINGLE PROMPTS FIRST

Get reliable output from one real task before chaining tasks. Use the Universal Prompt Template on work that actually matters. See where it breaks. Iterate.

STEP 2: ADD STRUCTURE WHEN OUTPUT FAILS

If a simple prompt works, don't complicate it. If it fails, diagnose with the Five Pillars and add the missing constraint.

STEP 3: BUILD A PERSONAL PROMPT LIBRARY

When you find a prompt that works reliably for a recurring task, save it. Templates compound. Don't solve the same problem twice.

STEP 4: RESIST PREMATURE AUTOMATION

The urge to "build an agent" is strong. But automation multiplies whatever you've built—including flaws. Only automate what you can already do reliably with manual prompts.

What This Playbook Doesn't Cover

It doesn't make AI factually reliable—you still verify. It doesn't replace domain expertise—you still need to know when output is wrong. It doesn't address model selection, fine-tuning, or API architecture. Those are different skills.

What comes next is evolution, not revolution. Models will improve, but the fundamentals won't change. Better models still require clear specs. Probabilistic systems still require governance.

Last Words

AI does not remove responsibility.

It concentrates it.

When something goes wrong, there is no model to blame, no vendor to point at, no abstraction to hide behind. There is only the person who decided what to ask, what to accept, and what to act on.

That person is you.

The difference between people who get leverage from AI and people who get burned by it isn't intelligence, access, or tooling. It's discipline. Professionals specify clearly, verify relentlessly, and never confuse plausibility with correctness.

Nothing in this playbook is about being clever. It's about being precise. Clear specs. Explicit constraints. Review before action. Governance before automation.

The models will improve. Interfaces will change. Hype cycles will repeat. None of that changes the fundamentals. Probabilistic systems will always require structure, and automation will always magnify whatever you put into it—good or bad.

YOUR PATH FORWARD

Start small. Pick one real task. Write a structured prompt. Evaluate the result. Fix what failed. Repeat. That's the work. And now you know how to do it.

Appendix A: The Governor's Checklist

A One-Page Control System

This checklist is the operational core of this playbook. It applies regardless of model, tool, workflow, or interface. Use it before you trust, publish, send, automate, or act on AI output.

1. Specification Check (Before You Prompt)

- ■ Do I know exactly what "done" looks like?
- ■ Could I hand this task to a competent human and get the same result?
- ■ Have I specified who the model should act as (ROLE)?
- ■ Have I provided only the relevant background (CONTEXT)?
- ■ Is the task executable, not vague (TASK)?
- ■ Did I control format, length, and structure (OUTPUT SHAPE)?
- ■ Did I state what the model must not do (GUARDRAILS)?

2. Output Review (Before You Trust It)

- ■ Does the output actually meet the spec I wrote?
- ■ Did the model make assumptions I didn't intend?
- ■ Are any claims stated as facts that might be guesses?
- ■ If this is wrong, do I know where it's likely wrong?
- ■ Could I explain this output—and its limits—to another human?

3. Hallucination Control

- ■ Did I restrict the source of truth where needed?
- ■ Did I ask the model to surface uncertainty or assumptions?
- ■ Are citations, quotes, or references verifiable?
- ■ Does any number, name, or claim require external confirmation?

4. Iteration Discipline

- ■ When output was wrong, did I diagnose the pillar that failed?
- ■ Did I tighten the spec instead of rephrasing vaguely?
- ■ Did I stop iterating once the output met the spec?
- ■ Am I keeping a reusable version of this prompt if it works?

5. Automation Readiness (Before Agents)

- ■ Can I get reliable results from this task manually?
- ■ Are all five pillars explicit—not implied?
- ■ Are there human review checkpoints?
- ■ Can I stop, correct, or reverse actions?
- ■ Do I know what the agent is not allowed to do?

6. Accountability Reality Check

- ■ Would I sign my name to this output?
- ■ Would I send this to a client or stakeholder as-is?
- ■ If this causes harm, delay, or embarrassment, am I prepared to own it?
- ■ Is judgment still human, or did I quietly outsource it?

> **THE RULE THAT OVERRIDES ALL OTHERS**
>
> If you wouldn't approve it without AI, don't approve it with AI. You are not here to trust the model. You are here to govern the system.

Appendix B: Prompt Failure Diagnostic Sheet

A One-Page Failure Diagnosis Tool

Use this immediately when AI output is wrong, weak, or untrustworthy. Do not rephrase the prompt. Do not change models. Diagnose first.

Step 1: Identify the Failure Symptom

- ■ Sounds reasonable but says nothing specific
- ■ Confident but likely wrong
- ■ Invents facts, sources, or details
- ■ Too long / rambling
- ■ Wrong level of detail
- ■ Off-topic or adjacent to the task
- ■ Refuses or hedges excessively

If none apply, your problem is not prompting.

Step 2: Map to the Failed Pillar

SYMPTOM	LIKELY CAUSE	FIX
Vagueness	Weak TASK / missing CONTEXT	Add specifics and situational context
Overconfidence	Missing GUARDRAILS	Add uncertainty flagging instructions
Hallucination	No source constraints	Restrict to provided documents only
Rambling	Missing OUTPUT SHAPE	Add explicit length/format constraints
Wrong detail level	Weak CONTEXT	Specify audience and depth
Off-topic drift	Weak TASK	Tighten scope, exclude adjacent topics
Generic output	Missing CONTEXT/ROLE	Add concrete context, strengthen role

If you cannot name the failed pillar, you are guessing.

Step 3: Apply the Correct Fix

ROLE failed →

- Specify expertise and perspective
- Explicitly state what the model is not

CONTEXT failed →

- Add only information required for correctness
- Remove irrelevant background

TASK failed →

- Define what "done" means
- Replace vague verbs with executable actions

OUTPUT SHAPE failed →

- Constrain format, length, and structure
- Force scannability (bullets, tables, limits)

GUARDRAILS failed →

- Prohibit speculation and invention
- Require uncertainty to be stated
- Restrict allowed sources

Bad output is not an AI failure. It is a specification failure.

Appendix C: Automation Readiness Gate

A Hard Stop Before Agents, Workflows, or Autonomy

Before turning any prompt into an agent, a workflow, a loop, a background process, or an autonomous system, every condition below must be true.

1. Manual Reliability Check

- ■ I can run this task manually using a single prompt
- ■ I get consistent, repeatable results
- ■ The output no longer surprises me
- ■ I know what "good" looks like before I see it

2. Specification Completeness

- ■ ROLE is explicit and constrained
- ■ CONTEXT is sufficient and not bloated
- ■ TASK is executable and unambiguous
- ■ OUTPUT SHAPE is predictable and reviewable
- ■ GUARDRAILS prevent speculation, invention, or overreach

3. Output Reviewability

- ■ The output can be reviewed quickly by a human
- ■ Errors are easy to spot
- ■ Assumptions are surfaced, not hidden
- ■ The format supports comparison and auditing

4. Uncertainty Visibility

- ■ The system explicitly flags missing information
- ■ The system distinguishes facts from inferences
- ■ The system can say "unknown" or "insufficient data"
- ■ The system does not force completion when confidence is low

5. Control & Reversibility

- ■ Actions can be paused or stopped
- ■ Outputs can be corrected before execution
- ■ Changes can be reversed
- ■ There is a defined failure state

6. Human Authority Checkpoints

- ■ A human approves before irreversible actions
- ■ A human can override the system
- ■ Responsibility is clearly assigned
- ■ No step relies on "the AI decided"

7. Boundary Definition

- ■ I know exactly what the system is allowed to do
- ■ I know exactly what the system is not allowed to do
- ■ Edge cases are explicitly excluded
- ■ Scope creep is technically blocked, not discouraged

PASS / FAIL RULE

All boxes checked → Automation may proceed
Any box unchecked → Automation is prohibited

There are no partial passes.

If a single prompt still surprises you, an agent will surprise you faster and more expensively.

Govern first. Automate last.

Appendix D: Universal Prompt Template

A Canonical Prompt Specification Template

This template exists to prevent omission. Most prompt failures are not caused by bad wording. They are caused by missing components. This template maps directly to the Five Pillars and treats prompts as specifications, not conversations.

Use it for:

- Single prompts
- Reusable templates
- High-risk tasks
- Workflows and pre-automation checks

ROLE

Who the model is and how it should behave

```
You are [specific role].
You prioritize [key characteristics or values].
You are NOT [roles, behaviors, or perspectives to avoid].
```

CONTEXT

Only the information required for correct execution

```
Relevant background information:
[Include what matters.]
[Exclude what doesn't.]
```

TASK

The executable objective

```
Your task is to [precise action].
"Done" means:
[Clear success criteria]
```

OUTPUT

Exact format and structure

```
Produce the output as follows:
• Format: [specify]
• Length: [specify]
• Structure: [specify]
```

GUARDRAILS

Constraints, exclusions, and safety rules

```
The model must:
[Required behaviors]
The model must NOT:
[Prohibited behaviors]
If information is missing or unclear:
• State that explicitly rather than guessing.
```

How to Use This Template

- For quick tasks: compress sections
- For complex or high-risk tasks: expand them
- For reusable work: save the final version
- Do not skip sections unless you understand the tradeoff

Templates do not make prompts smarter. They make failures visible—and controllable.

You are not here to trust the model. You are here to govern the system.