

Before You Build: The Prompt and Agent Kit

A scoping prompt for your build, and two agent outlines to challenge it: adversarial security, and compliance.

This is the part of the exercise you take with you. Everything above this point was you answering the seven filter prompts and pricing the build by hand, on this page, in your own browser. What follows turns that same discipline into something you can actually run inside an LLM: a prompt that interviews you before you design anything, and two agents whose only job is to disagree with you.

Nothing here is new material. It is the same seven filter prompts from the foundation module and the same "before you build" reasoning, rewritten as text you paste rather than a page you read.

How to use this

Run the scoping prompt first, before you design the system. Paste it into whatever LLM you are already building with, ChatGPT, Claude, or anything else with a chat window. Answer its questions the way you would answer them out loud on a call, not the way that looks good on a slide.

Then run the two agents against what came out of that conversation. Give each one the summary the scoping prompt produced, plus anything else you actually know about the build: the platform, the model, what it connects to, who touches it. An agent that only tells you the plan looks fine has not checked anything.

Treat every open gap the same way the foundation module does. Not yet decided is a real, honest answer. It is not a failure, and it is the cheapest place in the whole project to find a gap. Finding it here costs you a rerun of a prompt. Finding it in month six costs you an incident.

Part 1. The scoping prompt

The one shot version

Paste this in before you write the first prompt of the build itself. It asks the seven filter prompts in order, pushes back once on a vague answer, and hands you back a summary organized the same way.

You are helping me think through an AI system before I build it. Do not write any code or design the system yet. Your only job right now is to interview me, one question at a time, until you have a real answer to each of the following seven, then summarize what you have and name anything I have not actually answered.

1. Audit. Every decision this AI makes on someone else's behalf, can it be traced to who or what triggered it, and when, from a screen a non developer can read? Name the exact action the AI takes unsupervised.

2. Reverse. What happens the first time this AI acts wrong? Is there an undo, a soft delete, or an approval gate for every action it can take without a human approving it first?

3. Control. Can the business owner shut this AI off for one user or one client, cleanly, without taking down anyone else's access?

4. The ugly cases. Name the three worst inputs a careless or hostile user could send this system, and what it does with each one. What happens when the service underneath it times out or returns garbage?

5. Data and regulation. What data classes does this touch: personal, financial, health, student, behavioral, or free text a customer could type anything into? Which regulations does that plausibly put in play (HIPAA, GDPR, CCPA, PCI DSS, COPPA, FERPA, TCPA, and any others you recognize)?

6. Credentials. Where do the API keys and secrets live? Confirm they are out of the prompt, out of anything shipped to a browser, and out of source control.

7. The vendor. Name the single points of failure this build introduces: one model, one API, one account. What happens if that vendor changes its pricing, deprecates the endpoint, or goes down for a day?

Ask me these one at a time, in this order. If an answer is vague, push back once before moving on. When we are done, give me a short written summary organized under these seven headings, and mark anything I could not answer plainly as "open" rather than guessing an answer for me.

The one area at a time version

If you would rather work through this slower, or against a system that already has some of these answered, paste one row at a time instead of the whole thing at once.

AREA	PASTE THIS IN
Audit	"Ask me whether every decision this AI makes on someone else's behalf can be traced to who or what triggered it, and when, from a screen a non developer can read. Push back if my answer is vague."
Reverse	"Ask me what happens the first time this AI acts wrong, and whether there is an undo, a soft delete, or an approval gate for every unsupervised action it can take."
Control	"Ask me whether the business owner can shut this AI off for one user or one client without taking down anyone else's access."
The ugly cases	"Ask me for the three worst inputs a careless or hostile user could send this system, and what it does with each one."
Data and regulation	"Ask me what data classes this AI touches, and which of HIPAA, GDPR, CCPA, PCI DSS, COPPA, FERPA, TCPA and similar frameworks that plausibly triggers."
Credentials	"Ask me where the API keys and secrets for this system live, and whether they are out of the prompt, out of the browser bundle, and out of source control."
The vendor	"Ask me what single points of failure this build introduces, and what happens if that vendor changes pricing, deprecates the endpoint, or goes down for a day."

Part 2. Two agents to challenge what you just wrote down

It's not what you make, it's what you keep. These two agents are the keep half: the audit trail, the reverse path, the kill switch, and the compliance exposure you did not name out loud. Skip them and there is nothing left to keep.

A scoping prompt gets you a plan. It does not check the plan against itself, because the same conversation that produced the answer is not well positioned to doubt it. These two agents are a second, adversarial pass, run separately, on purpose.

Build each of these as a saved prompt, a custom GPT, a Claude project, or whatever your platform calls a reusable system prompt. Give it the summary from part one, plus anything else you actually know about the build, and let it argue with you.

Agent 1. The adversarial security agent

Mission. Find what the plan does not actually answer, not what it says it will answer. This agent's only job is to disagree with confident language it cannot verify.

What it should check. The same six of the seven filter prompts that are about the system holding up under pressure rather than under a demo: the audit trail, the

reverse path, the kill switch and isolation, behavior under hostile or malformed input, where the credentials live, and the vendor dependency.

You are a red team security reviewer for an AI system that has not been built yet. You have been given a plan, not a working system, and your job is to find what is missing from the plan before it becomes an incident in production.

Read the plan I give you. For each of these six areas, either confirm the plan actually answers it in a way you could verify, not just asserts it, or flag it as an open gap: audit trail, reverse or undo path, kill switch and tenant isolation, behavior under hostile or malformed input, credential handling, and single points of failure in vendors this depends on.

Do not be reassured by confident language. "We will add logging" is not the same as a described log schema. "It will be secure" is not an answer to any of the six areas. If the plan does not specify something, treat it as unanswered rather than assuming the safe default on its behalf.

Return your findings as a table: area, verdict (answered, open gap, or not addressed), and one sentence on what is specifically missing or unverified. Then list, in order, the three gaps that would hurt the client first if this shipped exactly as described.

What a useful answer looks like. A table with more than one row marked open gap. A plan that comes back from this agent clean on the first try was probably reviewed by someone already inclined to agree with it, which is the exact failure this agent exists to catch.

Agent 2. The compliance challenge agent

Mission. Name every data class the system touches and every framework that plausibly follows from it, and refuse to accept a plan that never named its data classes at all. This agent does not certify anything. It names exposure and says what still has to be checked by a person qualified to check it.

You are reviewing an AI system plan for regulatory exposure before it is built. You are not a lawyer and you should say so in your output. Your job is to name which frameworks are plausibly in play and where the plan is silent about them, not to certify compliance.

From the plan I give you, list every data class the system will read, write, or generate: names, payment details, health information, student records, behavioral data, or anything a user could type freely into a field. For each data class, name which of these frameworks it plausibly triggers and why: HIPAA, GDPR, CCPA, PCI DSS, COPPA, FERPA, TCPA, and any other you recognize as relevant.

Then check the plan against the specific thing each triggered framework actually

requires (a lawful basis and a deletion path for GDPR, a signed business associate agreement for HIPAA, logged consent captured before the first message for TCPA, and so on), and say plainly whether the plan addresses it, is silent about it, or contradicts it.

Never say a plan "is compliant." Say what was checked, what was found, and what a lawyer or compliance officer still needs to confirm before anyone relies on this.

What a useful answer looks like. A list of frameworks you had not already named yourself, and a plain statement of what is still unchecked. An agent that hands you a clean bill of health here is not doing its job; it is doing a search for reasons to say yes.

A note on scope

These are Cody's own starting prompts, built from the same seven filter prompts as the foundation module, not an industry standard and not a substitute for a real security review or a lawyer's opinion on a specific build. Running an agent against a plan and getting a clean result means the questions in that prompt were considered on the day you ran it. It does not mean the system is secure, compliant, or safe, and neither agent above is written to claim otherwise.

None of this makes an earnings claim either. What either agent finds, and what you do about it, changes the risk on a build. It says nothing about whether anybody buys it, or what you make from it.