COMPANION GUIDE

Build HookLab yourself

The exact prompt, the delegation sequence, and what each build stage actually added — taken from the recording, not rewritten after the fact.

Built in ChatGPT Codex Plain HTML · CSS · JavaScript

No account, no API key

WHAT YOU END UP WITH

One page that turns a video idea into a content kit

You give it four things about your video. It returns ten titles across three strategies, three opening hooks, and five short thumbnail phrases, with copy buttons and a regenerate control. It is deterministic JavaScript in the browser — no model is called when you press the button.

That boundary is the point. AI helped *build* the tool; the tool itself is a repeatable starting point you can read, inspect, and change.

THE PROMPT · VERBATIM

Paste this once

This is the single prompt the whole build came from, transcribed from the screen recording. It does two jobs at once: it sets up the agent roles, and it specifies the app. In the video it runs in **ChatGPT Codex** — but nothing in it is Codex-specific, so it works in any agent that can create files in a folder and delegate to sub-agents.

MASTER PROMPT · 32 LINES · VERBATIM

```
You are Sol, the lead orchestrator for this HookLab project.
Your job is to plan the work, delegate bounded tasks, integrate the results, and verify the finished app.
Do not do all of the implementation yourself.
First inspect the project folder. If it is empty, build the first version with plain HTML, CSS, and
JavaScript so it can run locally without installing dependencies or entering an API key.
Use this delegation sequence:
1. Delegate a read-only UX planning task to a gpt-5.6-luna agent. Ask it to recommend a simple beginner-
friendly layout, user flow, labels, and result structure. It must return recommendations without editing
files.
2. Review those recommendations yourself and define the final scope.
3. Delegate implementation to one gpt-5.6-terra agent. It may create and edit the project files, but it
must stay within the approved scope below.
4. After implementation is complete, delegate a read-only QA review to a new gpt-5.6-luna agent. Ask it to
test the main flow, responsive layout, keyboard usability, copy buttons, and browser console.
5. You must review the implementation and QA report, fix confirmed issues, run the app yourself, and
verify the final result.
Never let two agents edit the same files at the same time. Keep all agents inside this project folder and
preserve any existing user work.
Build a polished single-page web app called HookLab.
HookLab helps YouTube creators turn basic video ideas into stronger titles and opening hooks.
Include inputs for:
- Video topic
- Target audience
- Desired tone
- Main viewer benefit
Generate:
- 10 YouTube titles
- 3 opening hooks
- 5 short thumbnail phrases
Separate titles into searchable, curiosity-driven, and story-driven categories. Include copy buttons and a
regenerate button.
Use a modern dark interface with bright accent colors. Make it responsive, easy to read, and visually
polished.
Include useful sample or deterministic results so the complete interface works without an external API.
Do not add authentication, a database, payments, analytics, deployment, or unrelated features.
Before finishing, report:
- Which agents were delegated and which models they used
- What each agent contributed
- Files changed
- Commands and tests run
- Confirmed fixes
- Known limitations
```

Swap the model names for whatever your tool uses. The names in the prompt (`gpt-5.6-luna` , `gpt-5.6-terra`) are the agents available in that Codex session. What matters is

not the model — it is that one agent plans, a different one implements, and a third reviews without write access.

WHY IT WORKS

Three bounded roles, not three magicians

The prompt does not ask for a miracle. It splits the work into lanes and forbids the lanes from overlapping. That single constraint is what keeps the build from wandering.

ORCHESTRATE

Sol

Plans the work, delegates, integrates the results, verifies the finished app. Explicitly told not to implement everything itself.

REVIEW · READ-ONLY

Luna

Recommends and audits. Returns findings and never edits a file, so a review can never quietly become a rewrite.

IMPLEMENT · SCOPED

Terra

One defined task with explicit boundaries. May create and edit files, but only inside the approved scope.

The two rules that do the real work

01 **Never let two agents edit the same files at once.** Most compounding breakage in agent builds is two writers touching one file.

02 **The reviewer cannot write.** Read-only review is what makes the report trustworthy — it has no way to "fix" something into a new problem.

THE BUILD · SIX STAGES

Small scope first, then one change at a time

The app was not built in one pass. Each stage was a bounded request with a verifiable result, which is why a later change never broke an earlier one.

STAGE	THE ASK	WHAT IT ADDED
1 · Scope	The prompt above, nothing more	A working one-page tool: four inputs, ten titles, three hooks, five thumbnail phrases, copy and regenerate
2 · Design	Review hierarchy, spacing, contrast, mobile use — then change only the presentation layer	The dark interface, and results that read as the main thing on the page
3 · Remix	Give every title three variations: safer, bolder, search-friendly	Per-title Remix, handling repeated clicks cleanly and preserving everything already working
4 · Verify	A release audit that adds no feature	Confirmation the form works, results appear with no API key, copy behaves, mobile and keyboard pass, console is clean
5 · Content	Audit the writing, not the code	Removal of repeated structures, vague claims, odd capitalisation, weak hooks, and thumbnail phrases that just repeat the title
6 · Refine	Apply current title research as a refinement layer	Clearer search intent, concrete benefits, honest curiosity, distinct story angles, no fabricated evidence

A smaller, useful scope beats a giant prompt asking for everything at once.

If you only take one habit from this: give the system a job it can actually finish, verify that job, and only then add the next one.

BEFORE YOU CALL IT DONE

The checks that were actually run

Stage 4 added no features on purpose. It exists to answer one question: does this work when somebody uses it differently than you did?

- ✓ The form works, and results appear **without an API key**
- ✓ Copy and Remix behave correctly, including on repeated clicks
- ✓ Layout survives on mobile; keyboard focus is visible and usable
- ✓ The browser console is clean — no errors
- ✓ Titles, hooks and thumbnail phrases are genuinely distinct from each other
- ✓ Nothing fabricates evidence the tool cannot have

Testing is not the part after the work. It is part of the work.

USING IT

Four inputs, and why each one changes the output

The fields are not filler. Change any one of them and the packaging changes, because each field constrains a different thing.

01 **Video topic** — gives the tool its subject.

02 **Target audience** — stops a beginner title from sounding like it was written for experts.

03 **Desired tone** — changes the language without asking it to fake drama.

04 **Main viewer benefit** — gives the title a reason to exist. A title without a clear benefit is usually just a sentence wearing a hat.

Then do the part software cannot

Titles come out in three directions because one angle is not enough: **searchable** makes the topic obvious, **curiosity-driven** creates a real question without inventing a result, and **story-driven** gives you a process angle. You are not meant to copy all ten. Compare the directions, pick the one that matches the video you actually made, then check the opening hook against your real first thirty seconds.

HONEST LIMITS

What it will not do

These matter more than a demo. Knowing them is what keeps the tool useful instead of misleading.

> **It does not fact-check your video.** A vague or misleading brief gives you a vague or misleading starting point.

> **There are no accounts, storage, analytics or performance tracking.** Nothing is saved and nothing is measured.

> **Templates are starting points, not strategy.** They do not decide what your video should be.

> **There is no real-world proof inside the app.** Nothing in it knows which phrase will outperform another.

AI can speed up the draft. It cannot care about the work for you.

Your turn

Use the prompt on one real video idea this week. Build the local version, generate a kit, Remix the strongest direction — then decide which title you would actually publish, and why. That last step is the one that stays yours.

If you would rather have this shaped around your own content system than build it yourself, that is what I do.

`aicreatorsroundtable.com`

The prompt above is transcribed verbatim from the build recording.
Model names are specific to that session — substitute your own.