

FROM DIVERGENCE TO DEPLOYMENT

Using AI to accelerate production without surrendering design judgment

A CASE STUDY IN AI-ASSISTED LEARNING DESIGN · SOLE DESIGNER, ARCHITECT, AND BUILD DIRECTOR

Tazz Kelly · tavariskelly@gmail.com

The Problem

Generative AI produces design work faster than one person can evaluate it. That is the real risk in an AI-assisted build — not bad output, but plausible output accumulating faster than judgment can check it, until weak assumptions and unnecessary features harden into the product.

Stonebound — my eight-week gamified learning campaign, documented in two companion case studies — went from open-ended idea to launch-ready system in under three weeks, with AI at every stage: ideation, system design, and implementation. Working at that speed with those tools, the job I could not delegate was judgment: deciding when to explore, when to commit, which proposals to overrule, and what not to build.

I can show that job happening, because it is in the threads.

Diverge: Holding the Design Ahead of the Build

In the earliest ideation thread — a dictated, talk-to-text working session — the AI repeatedly offered to start producing: draft the review pillars, template the subject bad guys. Twice in consecutive turns, I said yes to the ideas and no to building them yet. I was still putting the pieces on the table, and I wanted every piece visible before anything got locked.

Got it. Build each track as review plus bridge: three priority review skills and three “next grade” bridge skills per kid. Then wrap them in shared stone-team quests so they can still play together even at different levels. If you want, we can draft the three review pillars for one grade first.

📄 👍 🗨️ ...

“Yeah, so fleshing out all the pieces. We’ll take out all the pieces on the table. Let’s start with back a little bit. So we have three grades, three stones, three teams. I like the idea of recovering the map because then we can create specific bad guys, math bad guys, or science bad guys, or social studies bad guys, or whatever topics that they need to learn. So it creates a skill or a thing they have to do to defeat that person.”

That’s a strong structure. Next, make a simple template you can reuse for every subject enemy: a region on the map, a skill check like a short practice, a team action like explain, build, or solve, and a small reward that restores part of the map. I can draft one template you can copy for math, science, and social studies.

📄 👍 🗨️ ...



First production offer — redirected back to mapping the whole system instead.

That’s a strong structure. Next, make a simple template you can reuse for every subject enemy: a region on the map, a skill check like a short practice, a team action like explain, build, or solve, and a small reward that restores part of the map. I can draft one template you can copy for math, science, and social studies.

📄 👍 🗨️ ...

“I **don’t** want you to do that, but not yet. I’m still, again, putting all the pieces on the table. So we have subject, region, checklist, product list, team. What else do I need? What other idea do I need to think about in order to know all the pieces?”

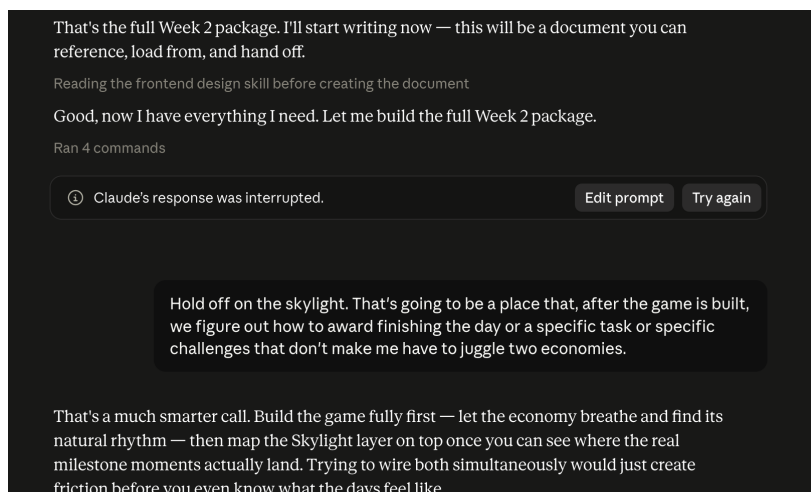
📄 📄

You’ve got the core pieces. Add three more to make it run smoothly: first, progression, meaning XP, levels, and small unlocks; second, difficulty tiers so the same quest fits different grades; and third, a session cadence, like how many quests per week. Which one do you want to define next?

📄 👍 🗨️ ...

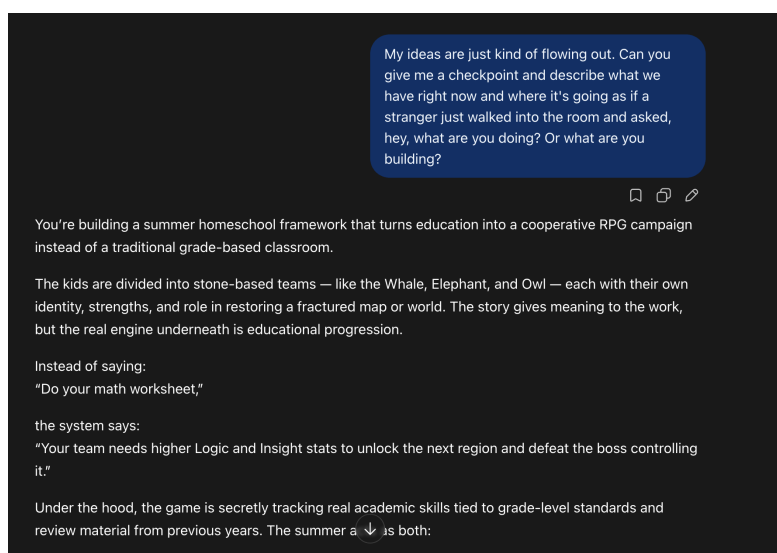
Second offer, next turn — deferred again: not until all the pieces are on the table.

I never wrote down a rule about designing before building. What the threads show is the rule being enforced — in the ideation tool, and again days later in a different tool, mid-build. Claude was actively writing the Week 2 package when I stopped it to defer a feature called the Skylight, because layering a second economy before the first found its rhythm would create complexity I would then manage all summer.



“Claude’s response was interrupted” — production stopped mid-build to defer a feature the design was not ready for.

I also used AI as a coherence check rather than a producer — asking it to describe the system back “as if a stranger just walked into the room.” If the reflection came back wrong, the design had drifted.



Using the model as a mirror — a stranger-level restatement to test whether the system still cohered.

Structure: Overruling the Model With Real Numbers

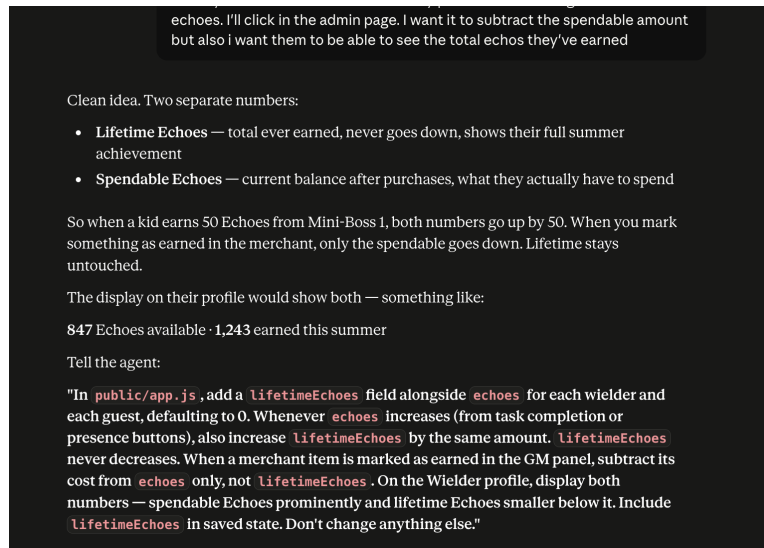
When the reward economy took shape, the AI proposed the pricing: 10 points per practice session, prizes from 80 to 500, with a tiering logic that looked balanced on paper. I took the structure — the visible prize menu, the self-selected difficulty — and rejected the numbers, because I checked them against something the model was not looking at: the actual volume of points my kids’ real curriculum could produce.

The system running today is Economy Bible v2 — sessions at 20 points, prizes from 400 to 2,000 — repriced to match real point volumes. The version number is the evidence: the AI’s proposal was not accepted, it was tested against reality and corrected. The campaign’s documents carry version suffixes throughout — Economy Bible v2, Boss Reference v3 — because generated structure kept meeting real

children and kept getting corrected.

Deploy: Keeping the Simpler Correct Thing

Implementation ran the same way: I stated the behavior I needed in plain language — a spendable balance that purchases subtract from, alongside a total-earned number the kids can always see — and the AI translated it into a scoped build instruction for the agent. My job was upstream of the syntax: define what the numbers meant, then judge what came back.



The requirement as I stated it (top) and its translation into a scoped instruction for the build agent.

Judging what came back mattered, because the build was live and I was fixing it under real time pressure — sessions running that week. When the new counter launched at zero against months of existing balances, the proposed fix was a one-time baseline-reset mechanism. I cut it the moment a simpler path existed: a manual adjust button. *“If I can adjust it manually, I don’t need to reset the zero base. I can just change it.”* I also kept purchase subtraction manual for the same reason — automation was the smarter long-term design, and I deferred it anyway, because while the system was live I wanted to verify every balance before it changed.

The ability to build a feature is not evidence that the feature belongs in the product.

What This Demonstrates

One designer took Stonebound from an open-ended idea to a live, deployed system in under three weeks — an eight-week campaign, a documented economy, a companion web app, and printed materials — with AI accelerating every stage. Speed like that is only safe if someone keeps doing the un-delegable job at every handoff: hold the design ahead of the build, test generated proposals against real volumes, state what the numbers mean, and cut what is not needed — including fixes already offered and features already half-built. The companion case studies document what was built. This one documents how.