AI TOOLING GUIDE

The $120/Month AI Stack That Ships a 5M-User App

Exactly how I split my budget across two providers, which model does which job, and the handoff that keeps quality high without ever running out of usage.

$20	$100	$120	3
Anthropic Pro plan	ChatGPT Pro plan	total monthly spend	models, 1 workflow

> This is not about which model is objectively best. It is about giving every stage of the build — planning, implementation, review — to the model that is actually strongest at that specific job, without paying for four figures a month or ever hitting a usage wall.

STAGE 01

Anthropic Pro — $20/mo

$20/mo

ROLE

UI generation and the final code review pass, using Opus 5.5.

Opus 5.5 is the model I reach for whenever a screen needs to look and feel right — component layout, spacing, visual hierarchy, the kind of work where taste matters as much as correctness.

The second job it does is the last one in the pipeline: reviewing everything the implementation model has already written. This is the final check that catches problems or bugs before they ship, after both planning and implementation are already done.

WHAT TO USE IT FOR

-> **UI generation**
Give it a rough layout or a screenshot of a competitor and let it produce the component structure and styling. It is consistently strong at getting the visual details right.

-> **Second code review pass**
After the implementation model finishes a feature, hand the full diff to Opus 5.5 with nothing but the goal of finding what was missed — bugs, edge cases, security issues, anything that slipped through.

TIP: Never let the model that wrote the code also be the only one that reviews it. A second model with fresh eyes catches what the first one is blind to.

[ ]	Opus 5.5 used for any screen where visual polish is the priority
[ ]	Every implementation diff gets a second review pass before merging
[ ]	Review pass runs with fresh context — not the same thread that wrote the code

STAGE 02

ChatGPT Pro — Astra

$100/mo

ROLE

Ingests the full codebase and writes the architectural plan before any code is written.

Before anything gets implemented, Astra reads the entire codebase and writes the plan for the architectural change — what files need to move, what the new structure looks like, what it will cost to run, and what could break.

This step exists because skipping straight to implementation on a large codebase is how you end up with changes that technically work but quietly break something three files away, or that ignore a cost implication you would have caught by planning first.

WHAT TO USE IT FOR

-> **Full codebase ingestion**
Point it at the whole repo, not just the files you think are relevant. It needs the full picture to catch downstream effects a narrower context would miss.

-> **Architectural planning**
Ask for a written plan before any code changes — the new structure, the migration steps, and anything that could regress. Review this plan yourself before it goes any further.

-> **Cost and constraint awareness**
Explicitly ask it to flag external costs — API usage, infra changes, anything that affects your bill — as part of the plan, not as a surprise after the fact.

HANDOFF PROMPT

HANDOFF PROMPT — ASTRA TO SOL

```
Ingest the full codebase before writing this plan.
1. Describe the current architecture relevant to this change
2. Propose the new structure and every file it touches
3. List external costs this change introduces or removes
```

```
4. Flag anything downstream that could break
5. Write this as a step-by-step implementation plan for Sol to follow

Do not write implementation code. Plan only.
```

[ ]	Astra given the full codebase, not a partial context window
[ ]	Written plan reviewed by you before implementation starts
[ ]	External and infra costs called out explicitly in the plan

STAGE 03

ChatGPT Pro — Sol

included

ROLE

Implements the plan Astra wrote, turning the architecture into working code.

Once Astra's plan is approved, Sol executes it. This is a deliberate split — the model that plans is not the same model that implements, which means the implementation is being checked against a plan rather than improvised on the fly.

Sol's output then goes straight to Opus 5.5 for the second review pass, closing the loop: plan, implement, review, all by different models.

WHAT TO USE IT FOR

-> **Implementation against a written plan**
Hand Sol the plan Astra produced, not a fresh description of the task. It should be executing a spec, not making architectural decisions of its own.

-> **Flag deviations from the plan**
If Sol needs to deviate from Astra's plan for a technical reason, that deviation should be called out explicitly, not silently absorbed into the diff.

[ ]	Sol implements from Astra's written plan, not from scratch
[ ]	Any deviation from the plan is flagged in the output, not hidden
[ ]	Completed work is handed to Opus 5.5 before it is considered done

WHY SPLIT ACROSS TWO PROVIDERS

The two plans combined cost $120 a month. That gets access to the strongest models from both companies, resets often enough that usage limits are never actually a problem, and a workflow where three different models check each other's work rather than one model reviewing itself.

DO	DON'T
Split planning, implementation, and review across different models	Let one model plan, implement, and review the same change
Give the planning model the full codebase, not a narrow slice	Subscribe to every provider's top tier without a reason for each
Use the strongest available model for the final review pass	Skip the review pass because the implementation 'looks fine'
Track usage resets so you always know which model has headroom	Pay four figures a month chasing marginal model differences

MONTHLY STACK CHECKLIST

Run this before you assume your subscriptions are pulling their weight.

[ ]	Anthropic Pro active — Opus 5.5 used for UI work and final review
[ ]	Every implementation diff gets a second review pass
[ ]	ChatGPT Pro active — Astra used for planning, not implementation
[ ]	Astra given full codebase context before writing any plan
[ ]	Written plan reviewed by you before Sol starts implementing
[ ]	Sol implements from the plan — deviations flagged, not hidden
[ ]	Opus 5.5 review pass happens after every Sol implementation
[ ]	Usage resets tracked so no model runs out mid-project

[ ]	Total spend still $120/mo — no unused top-tier subscriptions

$120/month. Zero usage walls. Three models checking each other's work so nothing slips through.

Comment 'MODEL' for the full guide.