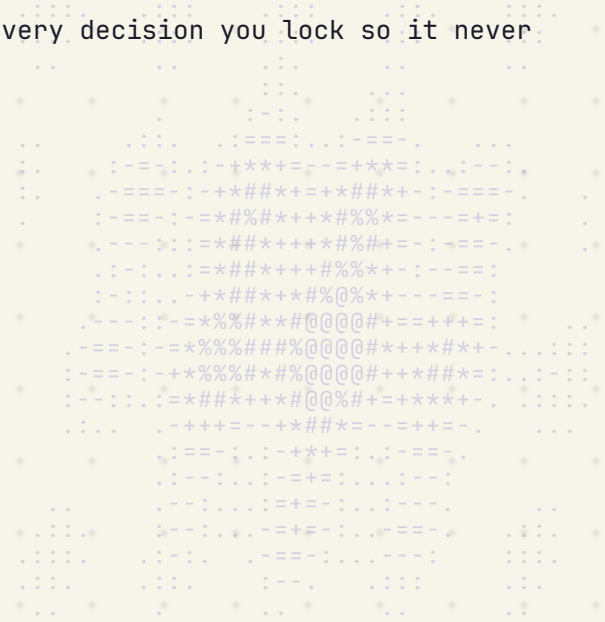
// SECTION 01 · CLAUDE SKILL

PROMPT MASTER

A free Claude skill that rewrites your prompt into an optimized format before answering, audits it against 35 credit-waste patterns, and locks the decisions you already made so you never re-explain yourself. Free on GitHub.

```
> boot opusjake_os
> resource: promptmaster
> version: v1.0 · 2026
> status: ready_
```

Every message you send re-sends the whole conversation. Five rounds of "no, I meant..." means you paid for the same context five times. Promptmaster ends the retries: it rewrites your raw ask into a prompt Claude gets right the first time, checks it against 35 ways prompts waste credits, and remembers every decision you lock so it never contradicts itself.



THE LEAK

The bill is not the answer. The bill is the retries.

When a prompt is vague, the model guesses. A wrong guess costs a correction, the correction re-sends everything before it, and the next attempt starts from a bigger, more expensive context. One fuzzy ask quietly becomes five paid turns.

The fix is boring and it works: make the first prompt specific enough that there is no second prompt.

```
raw:       write me something for instagram, idk

rewritten: Write a 30-day Instagram plan for a local coffee
           shop with 8k followers. One table: week, theme,
           3 post ideas. Under 400 words.
```

Same intent. One names the deliverable, the audience, the size, and the finish line. That is the whole trick, and Promptmaster does it for you on every request.

THE LOOP

Type `/promptmaster` before any request. Five steps run every time:

1. Load locked decisions. Your standing rules (voice, length, stack, emojis) get applied without being asked.

2. Rewrite the raw ask. Plain English, not a jargon template. The rewrite pins what gets produced, for whom, how big, under which constraints, and what "done" means. Missing facts get pulled from the conversation and your locks first. If something critical is genuinely unknown, you get one batched question with defaults. Never a drip.

3. Audit against the 35 patterns. The raw prompt is checked against every credit-waste pattern. The rewrite must clear all of them.

4. Print the receipt. Four lines showing what changed, so you learn the tight version yourself:

```
PROMPTMASTER
rewrote: "write me something for instagram, idk"
into:    "30-day IG plan, coffee shop, 8k followers,
          one table, <400 words"
flagged: #1 no deliverable, #2 no audience, #3 no length
applied: voice=casual (locked), emojis=never (locked)
```

5. Execute. You get the finished answer in the same turn, not homework.

35 WAYS PROMPTS WASTE CREDITS

The full audit list ships in the skill as `waste-patterns.json`. Check your own last prompt against it. Six families.

// VAGUE ASK

#	PATTERN	THE FIX
1	no deliverable named	name the artifact: one table, a script, a file
2	no audience	say who reads it and what they know
3	no length or size	cap it: under 400 words, 10 ideas
4	no done-when	state the finish line
5	"something like" hedging	commit to one version, change it after
6	open verdict fishing	ask the decision you actually face
7	style matters, no example	paste a sample, say "match this"
8	buried constraint	list every known constraint up front
9	unrelated asks fused	one prompt per job, or number them
10	"make it better", no direction	name the failure

// RE-EXPLAINING

#	PATTERN	THE FIX
11	re-pasting the same context	state it once, lock it, reference it
12	restarting instead of continuing	carry a 5-line summary + your locks
13	repeating a decision already made	lock it the first time
14	"like you said before", no quote	paste the 2 lines you mean
15	correcting instances, not the rule	correct the rule once, then lock it

// OVERFEEDING

#	PATTERN	THE FIX
16	whole file for one function	paste the function plus its callers
17	unsummarized transcript dump	work from a 10-line summary
18	irrelevant attachments	attach only what the task reads
19	screenshot of text	paste the text itself
20	paraphrased error message	paste the exact error
21	duplicated instructions	say each rule once, in one place
35	redundant context	add only what is new

// UNDERSPECIFIED PROCESS

#	PATTERN	THE FIX
22	no stack or tool pinned	pin it: plain HTML, no frameworks
23	no do-not-touch list	say what is frozen
24	missing environment facts	state OS, versions, plan limits
25	full rewrite when a diff would do	change only the hook, keep the rest
26	no output cap on long generation	cap it, ask for more if needed
27	no priority when tradeoffs exist	rank speed vs cost vs polish

// CONVERSATION CHURN

#	PATTERN	THE FIX
28	one-question drip	batch 4 questions in one message
29	permission-seeking turn	skip "can you...", just ask
30	resending all to change one word	send the delta
31	vague rejection	name the miss, say what to keep
32	politeness padding	spend those tokens on constraints

// WRONG TOOL FOR THE TURN

#	PATTERN	THE FIX
33	big model for a lookup	route trivia to a small model
34	regenerate instead of edit	fix the flawed section only

// RUN THE AUDIT

You do not need the skill installed to use this section. Before you send anything long or expensive, scan the six tables and fix what they flag. The skill just does it for you, every time, without being asked.

```
> audit last_prompt
> flagged: 4 patterns (#1, #3, #8, #32)
> estimated retries avoided: 3
> verdict: rewrite before sending
```

Most prompts trip three or four. Zero is the goal, and after a week with the receipt header you start writing the zero-flag version by hand.

LOCKED DECISIONS

The second half of the skill is memory. Say a preference once and lock it:

```
lock: voice = casual, like I talk
lock: length = under 200 words
lock: emojis = never
```

Locks live in a `decisions.md` file in your project. Every future request applies them automatically, and here is the part that saves real money: a request that contradicts a lock gets blocked, not silently absorbed.

```
This contradicts a locked decision
(length: under 200 words, locked 2026-08-14).
Say "unlock length" to change it,
or I keep the lock.
```

No more discovering in message twelve that Claude forgot the voice you set in message one. `decisions` prints the table. `unlock <key>` releases one.

INSTALL

Fastest path: paste the repo link into Claude and ask it to install the skill.

```
https://github.com/Jakeschincariol/promptmaster-skill
Install this skill, then confirm /promptmaster works.
```

Manual path for Claude Code:

```
git clone https://github.com/Jakeschincariol/promptmaster-skill
cp -R promptmaster-skill/.claude/skills/promptmaster ~/.claude/skills/
```

No Claude Code? Paste the repo's `SKILL.md` at the top of any Claude chat and it runs as a mode. Then use it:

```
/promptmaster write me something for instagram, idk
```

Tonight's version: take the last prompt you sent, run it against the six family tables above, and count the patterns it trips. Mine tripped four. That number is what you are paying for.

Free, open source, MIT licensed. Next skill drops at opusjake.ai.