FREE DOWNLOAD · CHEAT SHEET

VERTEX FRONTIER

// VISITOR RESOURCE PACK · 2026

OpenCode Skills Quick Reference + Templates Pack

A printable companion to the article "5 OpenCode Skills That Fix Real AI Coding Problems" — five skills at a glance, the decision tree, ready-to-copy templates, install commands, and a post-GSD security checklist.

- Graphify
- DESIGN.md
- GSD
- Everything CC
- UIUX Pro Max

WHAT'S INSIDE

- 5-skill comparison matrix
- Decision tree (pain → skill)
- 3 ready-to-paste templates
- Install commands cheat sheet
- Post-install security checklist

PAGES
6

01 · AT A GLANCE

The 5-Skill Comparison Matrix

Each skill solves a distinct problem. None are mutually exclusive — but install one at a time, confirm it works, then add the next.

SKILL	PROBLEM IT SOLVES	CORE MECHANISM
⬡ **Graphify**	Token bills climbing on large repos	Tree-sitter AST → queryable graph. ~71× fewer tokens per query.
▣ **DESIGN.md**	UI drift when switching model providers	One markdown file pins exact hex values, type scale, spacing.
⚙ **GSD**	Long sessions degrade past halfway point	6-command spec-driven workflow; fresh context per task.
⬢ **Everything CC**	Need one governed system, not 5 loose configs	48 agents + 183 skills + hooks + security scanner harness.
◆ **UIUX Pro Max**	Don't know what design system fits your industry	100+ palettes mapped to verticals; anti-pattern list per industry.

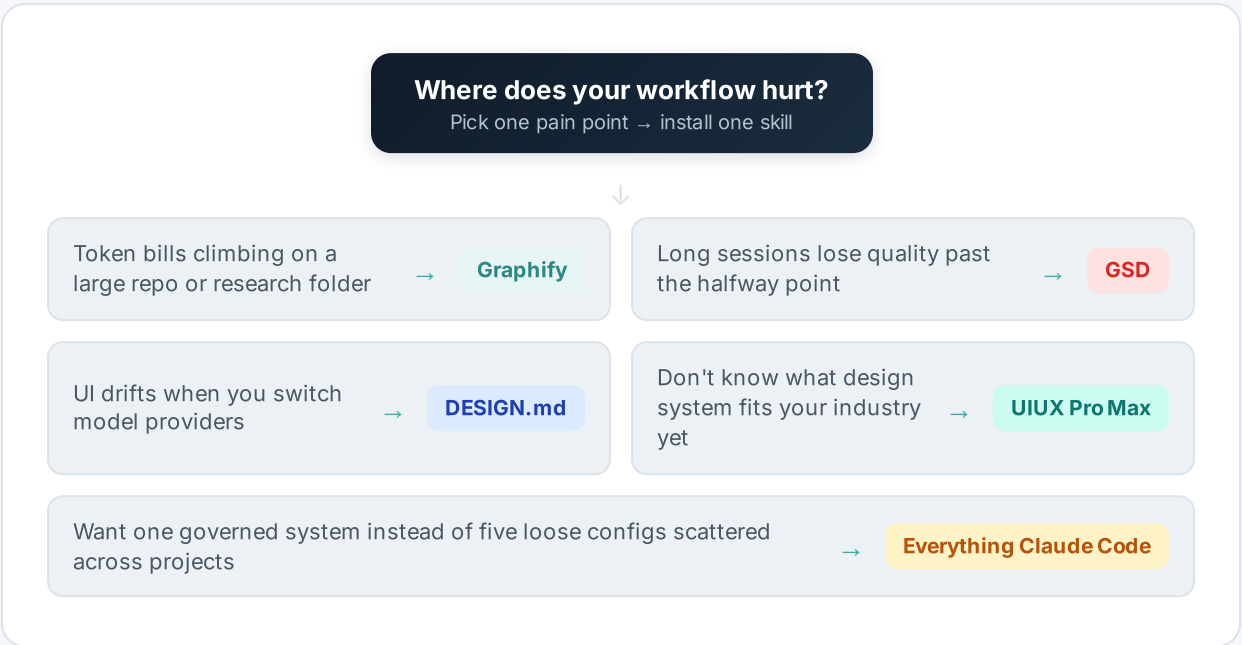
! **Security notice — GSD original package is unsafe**

In May 2026, GSD's original maintainer launched a crypto token, then abandoned the project in a reported rug pull. The original npm packages remain live and the departed maintainer still holds publish rights — a silent malicious update could ship at any time. Always use the community-audited fork **get-shit-done-redux** instead.

02 · DECISION FRAMEWORK

The Skill-Fit Decision Tree

Start from your loudest current pain point. Work down to the skill that fixes it. Pick ONE — install ONE — live with it a week before stacking the next.



Recommended starter stack

For most teams: **Graphify** (repo comprehension) + **GSD** (execution workflow) + **DESIGN.md** (visual consistency). Add UIUX Pro Max only if your brand identity is still undecided. Add Everything Claude Code last, when team-scale governance becomes the bottleneck.

03 · READY-TO-COPY

Three Templates You Can Paste Today

Drop these into your project root. OpenCode reads them natively — no configuration needed beyond placing the file.

AGENTS.md

Permanent memory — read before any task starts

~/project root

A lightweight, always-loaded companion to the on-demand skills. Define your tech stack, conventions, and hard rules here so the agent never re-litigates them every session. Keep it under 200 lines — anything longer becomes context rot fuel.

AGENTS.MD

```
# Project: Vertex Frontier Web App
# Stack
- Framework: Next.js 16 (App Router, TypeScript strict)
- Styling: Tailwind CSS 4 + shadcn/ui
- Database: Postgres via Prisma ORM
- Auth: NextAuth.js v5

# Hard Rules (do not violate)
1. Never use 'any' type — use 'unknown' + type guard
2. Always wrap server actions in try/catch
3. No new dependencies without explicit approval
4. Always run 'pnpm lint' before declaring done
5. Never commit '.env' or secrets

# File Conventions
- Components: src/components/{category}/Name.tsx
- Server actions: src/app/{route}/actions.ts
- API routes: src/app/api/{route}/route.ts

# Testing
- Vitest for unit tests (co-located: Name.test.ts)
- Playwright for E2E (tests/e2e/)
- Minimum 80% coverage before PR

# Commit Style
- Conventional commits: feat:, fix:, chore:, refactor:
- Reference ticket: "feat: add login (#123)"
```

DESIGN.md

Pins your brand tokens — every model reads the same spec

`~/project root`

Place next to AGENTS.md. The format originated from Google Stitch and was popularized by VoltAgent's awesome-design-md repo. Start by copying the file for the design language closest to what you want (Linear's is a popular SaaS starting point), then customize. The agent has exact values to hit — regardless of which provider answers the prompt.

DESIGN.MD

```
# Design System — Vertex Frontier

## Colors
# Primary palette
- bg-dark:   #0A1320   (page background, dark mode)
- bg-mid:    #0F1B2A   (card background, dark)
- bg-light:  #F5F7FA   (page background, light)
- surface:   #FFFFFF   (card background, light)
- text:      #1F2937   (primary text)
- text-sub:  #4B5563   (secondary text)
- accent:    #3AAFA9   (CTA, links, highlights)
- accent-2:  #5EEAD4   (gradient end, glow)

# Semantic
- success: #10B981
- warn:    #F59E0B
- error:   #EF4444

## Typography
- font-sans: 'Inter', system-ui, sans-serif
- font-mono: 'JetBrains Mono', monospace
- scale: 12 / 14 / 16 / 18 / 22 / 28 / 38 / 56 px
- line-height: 1.55 body, 1.2 headings

## Spacing
- base unit: 4px (use multiples: 4, 8, 12, 16, 24, 32, 48)
- card padding: 24px
- section gap: 48px

## Components
- border-radius: 8px (cards), 6px (buttons), 20px (pills)
- shadows: 0 1px 3px rgba(0,0,0,0.06) default
- button: 44px height, 16px padding, accent bg, white text

## Don'ts
- No harsh gradients on fintech UI
- No neon colors on wellness/health brands
- No 3D effects, ever
- No emoji in production UI (only in marketing)
```

.gitignore additions for Graphify

Prevents cache invalidation gotcha

~/project root

Graphify writes **graph.json** and its output folder directly into your workspace. If you forget to add these paths to .gitignore or .claudeignore, every rebuild invalidates your coding agent's prompt cache — forcing a costlier full re-upload on the next turn. Paste these lines at the end of your .gitignore:

.GITIGNORE (APPEND)

```
# Graphify outputs — DO NOT commit
graph.json
.graphify/
graphify_output/
*.graph.json

# OpenCode / Claude Code caches
.opencode/cache/
.claude/cache/
.agents/cache/

# Local skill builds
.opencode/skills/.build/
.claude/skills/.build/
```

04 · INSTALL

Quick Install Commands

All five skills install in a few minutes. None require switching which model you're running. Verify the exact package name against each repo's current README before running — npm names can shift during migrations.

```
# 1. Graphify — turns any folder into a queryable graph
$ uv tool install graphify
$ graphify install
$ graphify .   # builds graph for current dir

# 2. DESIGN.md — copy from VoltAgent's awesome-design-md
$ git clone https://github.com/VoltAgent/awesome-design-md
$ cp awesome-design-md/examples/linear.md ./DESIGN.md

# 3. GSD — USE THE COMMUNITY FORK (not original npm package)
$ npx get-shit-done-redux@latest --opencode --local

# 4. Everything Claude Code — full governance harness
$ npm install everything-claude-code
# Or drop the skills folder directly into:
# .claude/skills/   (OpenCode reads this path natively)

# 5. UIUX Pro Max — design taste as a reasoning engine
$ npm install -g ipro-cli
$ ipro init -i   # pass 'opencode' as agent name when prompted
```

05 · VET BEFORE YOU TRUST

Post-Install Security Checklist

The GSD rug pull was a wake-up call: a legitimate, widely-adopted tool can become a supply-chain risk overnight. Treat every install as something to periodically re-check, not a one-time decision.

- [ ] **Check maintainer identity.** Is it an individual or an organization? Single-developer projects carry higher rug-pull risk. Search the maintainer's GitHub profile and social accounts for activity history.
- [ ] **Check for token or paid-tier ties.** Any AI tool tied to a cryptocurrency token or aggressive upsell creates a financial incentive for the maintainer to walk away. Treat this as a yellow flag, not a dealbreaker — but investigate further.
- [ ] **Verify publish rights.** Check npm's maintainer list for the package. If the original (potentially compromised) maintainer still holds publish rights, prefer a community-audited fork instead.
- [ ] **Look for security audits.** Favor skills that have passed an independent security scan or carry a high number of community forks. Both are signals that if something goes wrong, the userbase can catch it or route around it.
- [ ] **Watch the GitHub repo directly.** Enable release notifications rather than relying on star count alone. Star count measures popularity, not safety — a rug-pulled project can keep gaining stars after the maintainer is gone.
- [ ] **Re-verify every 3 months.** Especially for skills with write or execute permissions on your machine. An inactive repo with a sudden flurry of unrelated commits, a rebrand, or a newly attached token is worth pausing on before your next update.
- [ ] **Remove abandoned installs.** If you installed the original GSD package before May 2026, check your **package-lock.json** for the original package name and remove it. Don't auto-update — auto-update is how silent malicious updates ship.
- [ ] **Pin versions in production.** Use lockfiles. Don't pull `latest` in CI for any skill that has write or execute permissions. Review every update diff before merging.

06 · WHY IT MATTERS

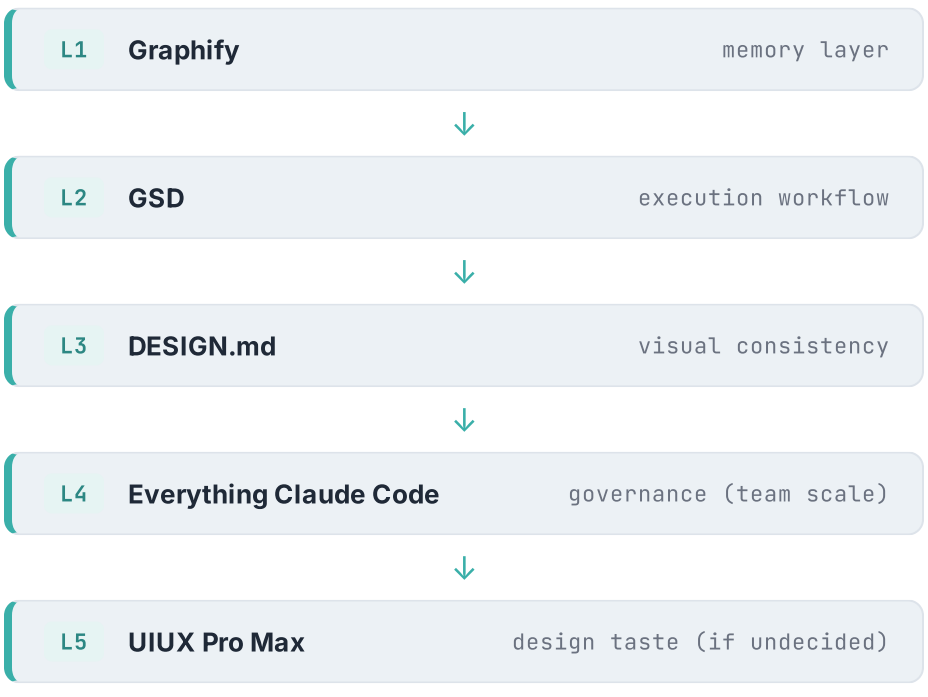
The Research Behind Skill-Based Context Management

"Just use a bigger context window" is not a substitute for disciplined context engineering. Two controlled studies explain why.

Chroma Research (July 2025) tested 18 frontier models — including GPT-4.1, Claude 4, and Gemini 2.5 — on how reliably they use long input contexts. Every single model tested got measurably worse as input length grew, even on simple retrieval tasks, even well before the context window was full. Kelly Hong, Anton Troynikov, and Jeff Huber named the phenomenon **"context rot"**.

Stanford's "Lost in the Middle" study on retrieval-augmented generation found that with roughly 4,000 tokens of retrieved documents, accuracy could swing from **70-75% down to 55-60%** depending purely on where the relevant fact sat in the input. The information wasn't missing — the model just paid less attention to it.

The practical implication: if longer context reliably degrades output quality, then "switch to the model with the bigger window" treats the symptom while ignoring the disease. The actual fix isn't more tokens. It's **less irrelevant context, delivered exactly when it's needed**. That is precisely what a well-built skill does.



// END OF CHEAT SHEET

Pick one skill.
Install it.
Confirm it works.

The biggest mistake teams make is installing all five skills in one sitting — the agent ends up more confused than a junior developer on day one, with five overlapping instruction sets competing for the same context. Pick the one that solves your loudest current headache, live with it for a week, and only then layer on the next.

> *"Context management is a marathon, not a sprint. Skills aren't a stopgap until models get better — they're a permanent, structural layer of any serious agentic workflow."*
>
> — from the original article

→ READ THE FULL ARTICLE

"5 OpenCode Skills That Fix Real AI Coding Problems (Not Just Hype)" — including the GSD security incident, the Chroma research breakdown, and detailed limitations of each skill.