AI Agent Resources

Resources, tools, and frameworks for AI coding agents and autonomous AI systems

- AI Agent Resources

AI Agent Resources

This page collects useful resources for effective **agentic engineering** — tools, frameworks, and skills that make coding agents more capable, reliable, and productive.

Each entry is something worth integrating into your agent workflow, whether as a skill, plugin, or standalone tool.

Impeccable

URL: impeccable.style · GitHub

Stars: 27k+

A design-focused AI agent skill that teaches your coding agent how design works. It loads seven reference files on every prompt (typography, color, motion, spatial, interaction, responsive, UX writing) to give your AI a shared design vocabulary.

Key features:

- **23+ design commands** — /typeset, /colorize, /animate, /layout, /critique, /audit, /polish, /distill, /adapt, and more
- **Design intelligence layer** — before any code runs, Impeccable establishes design context
- **Create → Evaluate → Refine → Simplify workflow** — structured design flow for coding agents
- **Codex asset producer agent** — v3.1.0 adds critique persistence and palette-first image flow
- **Cross-agent support** — works with multiple coding agent platforms

Use case: When your AI agent needs design-aware output — typography, color systems, spacing, motion, responsive layouts, and UX writing.

Ultrapowers

URL: github.com/danduma/ultrapowers

Stars: 1

A framework that adds a Product Manager role to your coding agent. It introduces PM-style thinking — requirements, prioritization, scope management — into the agent workflow.

Key features:

- **PM capabilities** — adds product management perspective to coding agents
- **Multi-agent support** — plugins for Claude, Codex, Cursor, and OpenCode
- **Structured agent hooks** — hooks, agents, commands, and scripts directories
- **Active development** — 454 commits, regularly updated

Use case: When you want your coding agent to think more like a PM — scoping work, managing priorities, and making product decisions before coding.

Playwright CLI

URL: github.com/microsoft/playwright-cli

Stars: 10k+

Microsoft's official CLI for common Playwright actions. Record and generate Playwright code, inspect selectors, and take screenshots — all from the command line.

Key features:

- **Record mode** — record browser interactions and generate Playwright code automatically
- **Code generation** — generate tests in multiple languages (TypeScript, Python, Java, C#)
- **Selector inspection** — inspect and test CSS selectors in the browser
- **Screenshot capture** — take screenshots and PDFs of web pages from the CLI
- **Multi-language support** — generated code for TypeScript, Python, Java, and C#

Use case: When you need to record browser interactions, generate Playwright test code quickly, or inspect selectors without opening the full Playwright Inspector.

Taste Skill

URL: tasteskill.dev · GitHub

A design-system skills pack for AI coding agents focused on producing premium, distinctive frontend output — the creator's pitch is "escape the generic AI slop." It ships as a collection of SKILL.md files that plug into any agent supporting the skills protocol (Cursor, Claude Code, Codex, Gemini CLI, v0, Lovable, OpenCode, and more).

Key features:

- **Modular skill set** — separate skills for different goals: `taste-skill` (default all-rounder), `gpt-tasteskill` (stricter for GPT/Codex), `image-to-code-skill` (image-first workflow), `redesign-skill` (visual audit), `soft-skill` (calm/expensive aesthetic), `minimalist-skill` (editorial), `brutalist-skill` (Swiss typography, raw structure), and `stitch-skill` (Google Stitch-compatible export)
- **Image generation skills** — `imagegen-frontend-web`, `imagegen-frontend-mobile`, and `brandkit` for producing design reference images with strong art direction
- **Execution guard** — `output-skill` prevents placeholder code and skipped sections
- **Open beta** — v2 is in open beta with downloadable files

Use case: When you want your AI agent to produce frontend output with a cohesive, high-end visual identity rather than generic default styling. Install the skill pack and pick the aesthetic that matches your project.

Layers

URL: layers.jamiemill.com · GitHub

A design-thinking skills pack by Jamie Mill that gives AI agents a structured framework for product design decisions. Rather than jumping straight to screens, Layers walks the agent through seven layers of product design — from surface-level visuals down to user needs, domain knowledge, and observed behavior.

The Seven Layers:

1. **Surface** — visual design, colors, typography
2. **Interaction structure & flow** — navigation, states, edge cases
3. **Conceptual model** — objects, relationships, mental models
4. **Product & service strategy** — outcomes, bets, positioning
5. **User needs** — job stories, priorities, pain points
6. **The domain** — terminology, constraints, expertise
7. **Observed behaviour** — analytics, usage patterns, real-world data

Key features:

- **Diagnostic command** — `/layers-orient` audits all seven layers and tells you which is the bottleneck
- **Layer-specific skills** — individual `/layers-*` commands for deep dives (e.g., `/layers-user-needs`, `/layers-conceptual-model`, `/layers-interaction-flow`, `/layers-domain`)
- **Structured output** — produces markdown and mermaid diagrams (job stories, strategy trees, object maps, decision inventories)
- **MCP integrations** — can write decisions directly to Linear, Notion, Figma, or GitHub
- **Open source** — installable via `npx skills add jamiemill/layers-skills`

Use case: When you're starting a new feature or redesign and need your AI agent to think through the *right* design decisions before touching a pixel. Especially valuable when your team can't agree on direction — the framework surfaces what you're actually disagreeing about.

Superdesign

URL: app.superdesign.dev

An AI-powered design tool that generates production-ready website designs from text prompts, screenshots, or existing URLs. It combines a prompt library with AI generation and offers Cursor/Claude Code/VS Code integration for seamless handoff to development.

Key features:

- **Multi-modal input** — generate designs from text prompts, recreate from screenshots ("Recreate Screenshot"), or import from existing sites ("Import from Site")
- **Prompt library** — curated design templates including brutalist e-commerce, high-contrast landing pages, SaaS layouts, and editorial styles — each with detailed design system specifications (colors, typography, spacing, grid)
- **Design system integration** — use your own design system or let Superdesign generate one
- **AI model selection** — supports Gemini 3 Flash and other models for generation
- **Animation & effects** — explore and apply scroll-triggered animations, transitions, and visual effects
- **IDE integration** — works with Cursor, Claude Code, and VS Code for direct code generation

Use case: When you need rapid design exploration or want to generate a polished landing page / component library from a prompt. Great for early-stage projects, design sprints, or when you want to quickly validate a visual direction before committing to implementation.

Archify

URL: github.com/tt-a1i/archify

Stars: 523 · **Forks:** 33

A Claude Skill that generates polished architecture, technical workflow, sequence, data-flow, and lifecycle diagrams from plain-English descriptions. It outputs a single self-contained HTML file you can open in a browser, toggle between dark/light themes, copy to clipboard, or export at high resolution (4× PNG, JPEG, WebP, SVG).

Key features:

- **Natural language to diagrams** — describe your architecture or process in English, get a rendered diagram
- **Diagram types** — architecture diagrams, technical workflows, approvals, tool call chains, CI/CD pipelines, runbooks, request call chains, data pipelines, PII boundaries, and state machines
- **Theme toggle** — one-click switch between dark and light modes, persisted across sessions
- **Export options** — copy to clipboard (paste into Slack, Notion, GitHub) or export as 4× PNG, JPEG, WebP, or SVG
- **Single HTML output** — no dependencies, works offline, embeddable anywhere

Use case: When you need to quickly visualize system architecture, data flows, or technical processes and want a clean, exportable diagram without opening a drawing tool. Especially useful in the middle of coding sessions when you need to document or share a design.

Vikunja CLI

URL: github.com/jo-nike/vikunja-cli

Stars: 1 · **Forks:** 1

A command-line interface for Vikunja, the open-source to-do and project management application. All output is JSON, making it easy to integrate with scripts and other tools.

Use case: When you want your agent to interact with Vikunja task lists programmatically — creating, querying, updating, and completing tasks via a CLI that outputs structured JSON for reliable parsing.