ENGINEERING OPERATIONS MANUAL & SPECIFICATION

SKILL BRAIN v2.1

The complete guide to orchestrating precision engineering skills, dedicated domain planners, dynamic responsive sizing, user-controlled virtual memory, and zero-AI-slop execution.

> **AXIOM OF ENGINEERING DISCIPLINE**
>
> We build real engineering products. Not AI-generated slop. Not template defaults. Not the average of everything the model has seen. Every decision here is a deliberate, mathematically grounded choice made for this project and this project only.

ARCHITECTURE VERSION	ACTIVE SKILL SUITE	CONFIGURATION ROOT
v2.1.0 (Production)	54 Specialized Skills	`.agents/` & `.agent`

Executive Reference & Commands

Skill Brain turns general-purpose AI coding agents into disciplined senior engineering partners. Rather than executing monolithic, hallucinated passes, the agent decomposes requests across 6 categories, confirms its plan, loads at most 2–3 precision skills simultaneously, and verifies all work with raw CLI evidence.

Dollar Commands & Shorthand Triggers ($) & Planners

You can invoke any category, dedicated pre-implementation planner, or specialized capability instantly using dollar commands ($):

Command	Shorthand	Target Orchestrator / Skill	Operational Role
`$orchestrate`	`brain:`	`supreme-orchestrator`	Apex router. Evaluates multi-domain requests, selects category routing, and requests user confirmation.
`$plan-frontend`	`plan-ui:`	`frontend-planner`	**Dedicated UI Planner:** Formulates complete subject inventory, fluid tokens, phone & desktop blueprints before code.
`$frontend`	`ui:`, `fe:`	`frontend-master`	Frontend suite with Taste Kill v2 + Anti-AI SOP + Dynamic Responsive Sizing baselines.
`$plan-backend`	`plan-api:`	`backend-planner`	**Dedicated Backend Planner:** Formulates domain schemas, responsive API contracts, and TDD red-green test plans.
`$backend`	`api:`, `be:`	`backend-master`	Backend suite enforcing Karpathy surgical simplicity + TDD loop + responsive contracts.
`$plan-debug`	`plan-fix:`	`debug-planner`	**Dedicated Debug Planner:** Formulates symptom, reproduction command, and candidate hypotheses. Code edits gated.
`$debug`	`fix:`	`debug-master`	Executes diagnostic plan, isolates root cause, and enforces verification before patches.
`$plan-anti-slop`	`audit-plan:`	`anti-slop-planner`	**Dedicated Anti-SOP Planner:** Pre-remediation 6-dimension code scanner & itemized repair plan.
`$anti-slop`	`gate:`, `audit:`	`anti-ai-sop-agent`	Runs 6-dimension quality gate audit producing a strict PASS / WARN / FAIL scorecard with `--fix`.
`$architect`	`arch:`, `isolate:`	`architecture-master`	Workspace isolation via git worktrees, subagent dispatch, and multi-session task mapping.
`$memory` or `$plan`	`memory:`	`memory-master`	User-controlled virtual memory: generates `plans/`, `phases/`, and live status dashboard.
`$taste`	`kill-slop:`	`design-taste-frontend`	Forces the 247-site study art direction: subject-grounded palettes, no stock templates.
`$typography`	`type:`	`typography-master`	Selects domain font archetype (Swiss, Editorial, Mono, Humanist) and fluid clamp scales.
`$icon`	`svg:`	`svg-icon-craftsman`	Crafts clean 24×24 grid-snapped vector geometry with uniform 1.5px/2.0px stroke.
`$verify`	`test:`	`test-verifier`	Enforces evidence before claims: requires raw CLI output before declaring any task complete.

Command	Shorthand	Target Orchestrator / Skill	Operational Role
`$git`	`git:`	`git-master`	**Apex Git Orchestrator:** Coordinates branch management, multi-commit splitting, and memory phase commit hooks.
`$commit`	`commit:`	`git-atomic-committer`	Analyzes diffs, splits changes into 2+ atomic commits across domain boundaries, Conventional format with test evidence.
`$branch`	`branch:`	`git-branch-manager`	Enforces strict kebab taxonomy (`feat/` , `fix/` , `phase/`), clean tree verification, rebase workflow.
`$phase-commit`	`phase-commit:`	`git-phase-committer`	Automatic Virtual Memory Phase commit hook: sequential 3-step commits with verification proof.

> **THE TOKEN EFFICIENCY LAW (MAX 2–3 SKILLS)**
>
> The model must **never load more than 2 to 3 skills at the same time**. Attempting monolithic passes burns context, degrades attention, and causes hallucinated regressions. The agent must pick the sharpest combination, confirm with you, and execute cleanly.

Dynamic Responsive Sizing & Universal Device Support

A critical mandate of Skill Brain is that **every UI component and page must adapt dynamically to every screen size** — from compact 320px mobile phones to 2560px ultra-wide desktop monitors. We divide this into two specialized layout disciplines, unified by a mathematical clamp engine.

1. Mobile Phone Perfection (`phone-layout-master`)

Targeting **320px – 480px** viewports with strict touch ergonomics:

- **Dynamic Height Units:** Mandatory `100svh` and `100dvh`. Never `100vh`, which clips under mobile address bars.
- **Thumb-Zone Ergonomics:** Primary actions and tabs live in the lower 40% of the screen. Touch targets strictly >= 44×44px.
- **Hardware Insets:** Top and bottom bars padded with `env(safe-area-inset-top)` and `env(safe-area-inset-bottom)` for notches, Dynamic Island, and home bars.
- **Zero Horizontal Overflow:** Global `overflow-x: hidden` and `overflow-wrap: break-word` prevent text blowouts.
- **Modal Transformation:** Dialogs convert into ergonomic bottom swipe sheets.

2. Desktop Workstation Density (`desktop-ui-architect`)

Targeting **768px – 2560px+** viewports with spatial hierarchy:

- **Maximum Content Bounds:** Running text constrained to `75ch`; grids bounded by `max-w-7xl` (1440px) with auto margins. Zero awkward 4K sprawl.
- **Container Queries (`@container`):** Components adapt to the width of their *parent container* (e.g. inside a sidebar or split pane), not just the global window.
- **Multi-Pane Spatial Hierarchy:** 3-column workstation layout (64px Nav Rail | 320px Master List | 1fr Detail Canvas) with independent scroll containers.
- **Auto-Fit Grids:** `repeat(auto-fill, minmax(min(100%, 320px), 1fr))` allows cards to wrap naturally without fixed column traps.
- **Keyboard Shortcuts:** `⌘K` command palette, `Esc` dismiss, visible focus rings.

The Mathematical Fluid Clamp Engine (`dynamic-ui-engine`)

Hardcoded pixel widths and arbitrary media query jumps are replaced with continuous fluid clamp mathematics that calculate optimal values across any screen width:

```
/* Universal Dynamic Responsive Tokens */
:root {
  /* Fluid Spacing Scale: 320px -> 1440px */
  --space-xs:  clamp(0.75rem, 0.7rem + 0.25vw, 0.875rem);
  --space-sm:  clamp(1rem, 0.9rem + 0.5vw, 1.25rem);
  --space-md:  clamp(1.5rem, 1.3rem + 1vw, 2rem);
  --space-lg:  clamp(2rem, 1.7rem + 1.5vw, 3rem);

  /* Fluid Typography Scale */
  --font-size-base: clamp(0.9375rem, 0.9rem + 0.19vw, 1.0625rem);
  --font-size-lg:   clamp(1.125rem, 1.05rem + 0.38vw, 1.375rem);
  --font-size-hero: clamp(2.25rem, 1.8rem + 2.25vw, 4rem);
}
```

Backend Responsive UI Contracts (`responsive-contract-enforcer`)

Responsive design is not merely a CSS concern — it begins at the backend contract layer. Backend engineers and API designers must honor these three contract rules:

1. **No Hardcoded Dimensions in Payloads:** APIs must never return fixed pixel coordinates (e.g. `width: 1200`). Send semantic layout tokens (`layout: "single-stack" | "auto-fill"`).
2. **Aspect Ratio & SrcSet Metadata:** Every media entity returned by an API must include explicit aspect ratios (e.g. `"16/9"`) and multi-resolution variant arrays to eliminate Cumulative Layout Shift (CLS).
3. **Adaptive Pagination:** Cursor-based stream pagination for mobile feeds; indexed pagination with total counts for desktop data tables.

The Virtual Memory Engine

Context windows are volatile and ephemeral; files on disk are permanent and auditable. The Virtual Memory Engine ensures complex software builds never drift or lose context across long sessions.

Directory Architecture

```
.agents/memory/
├── plans/
│   └── main-plan.md       ← Active master plan
├── phases/
│   ├── phase-01-scaffold.md
│   ├── phase-02-core-logic.md
│   └── phase-03-ui-polish.md
├── archive/               ← Completed & verified
│   └── 2026-11-02-auth/
└── decisions/             ← Permanent ADRs
    └── ADR-001-jwt.md
```

Strict User Control Gate

Unlike fragile auto-memory tools that dump conversational noise to disk, Skill Brain memory is **100% user-controlled**:

- **Casual fixes & questions:** Handled in-context. Zero disk writes.
- **Complex multi-step features:** Agent asks or user triggers `$memory` or `$plan`.
- **Lifecycle:** Plans and phase files are active during execution, then automatically archived and purged when verified.

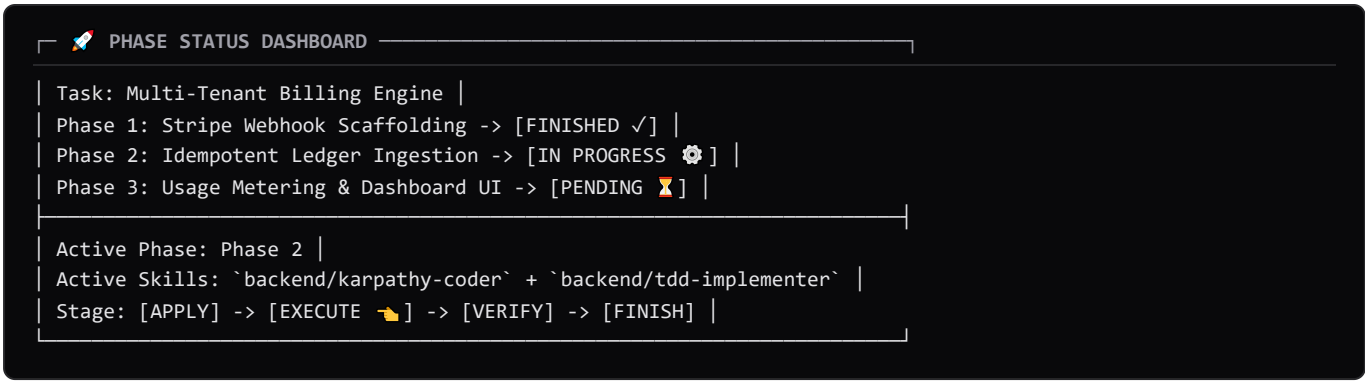
The 4-Stage Phase Lifecycle: [APPLY] → [EXECUTE] → [VERIFY] → [FINISH]

When virtual memory is active, work is divided into bite-sized phases. Every single phase proceeds through four rigorous stages:

Stage	Action	Standard & Expected Output
[APPLY]	Skill Pre-Selection	The agent loads the 2–3 specialized skills specified in the phase file and announces them in chat.
[EXECUTE]	Surgical Build	Executes step-by-step instructions. Implements tests first (TDD) or builds layout from subject inventory.
[VERIFY]	Evidence Validation	Runs test suites, linters, or visual inspectors. Pastes actual CLI terminal output into the chat.
[FINISH]	Status Checkpoint	Marks phase as finished in memory. Renders updated status dashboard and prompts for user confirmation.

The In-Chat Live Status Dashboard

During active phased execution, the agent **must prepend this live status widget** at the very top of every single response, giving you complete clarity on project state:



Automatic Cleanup & Archival Lifecycle

When the final phase passes all verification criteria:

1. The active `main-plan.md` and all `phase-*.md` files are moved to `.agents/memory/archive/[Date]-[TaskName]/`.
2. The active `.agents/memory/phases/` folder is completely cleared.
3. The workspace memory is pristine and ready for the next task without token bloat.

Category Deep-Dives: Frontend & Anti-AI SOP

The frontend subsystem is engineered around one core benchmark: **award-grade craft with zero AI tells and universal dynamic responsiveness**. It combines the 247-site empirical study from *Taste Kill v2* with modern typography, precision vector drafting, and 16.6ms frame-budget animation.

The Frontend Skill Suite

Skill Name	Primary Trigger / Command	Core Capability & Engineering Benefit
`frontend-planner`	`$plan-frontend`, `plan-ui:`	**Pre-Build UI Blueprint:** Subject inventory, fluid tokens, phone & desktop layouts, and anti-slop check before code.
`phone-layout-master`	`phone:`, `mobile-ui:`	Mobile phone perfection (320px–480px, `100svh`, 44px touch targets, safe-area insets, bottom sheets, zero horizontal scroll).
`desktop-ui-architect`	`desktop:`, `workstation:`	Desktop application UI (768px–2560px, multi-pane split views, container queries `@container`, `max-w-7xl` bounds).
`dynamic-ui-engine`	`dynamic-ui:`, `responsive:`	Master responsive sizing engine. Fluid CSS clamp math, aspect-ratio stability, zero cumulative layout shift.
`design-taste-frontend`	`$taste`, `kill-slop:`	The complete Taste Kill v2 engine. Replaces template defaults with subject-matter inventory, researched composition directions, and an inspect-and-fix loop.
`ui-taste-enforcer`	`taste:`, `art-direct:`	Anti-slop art direction: forbids generic defaults, conducts 8–12 point client subject inventory, enforces the "competitor test" on visual concepts.
`ui-design-principles`	`design-plan:`	Two-pass design workflow: compact token plan → brief uniqueness review → surgical build → screenshot critique.
`typography-master`	`$typography`, `type:`	Domain font archetypes (Swiss, Editorial, Mono, Humanist), mathematical scale ratios, optical tracking compensation, and tabular figures.
`svg-icon-craftsman`	`$icon`, `svg:`	Hand-crafted SVG icons. Strict 24×24 grid, 1.5px/2.0px stroke, integral pixel snapping, `currentColor` decoupling. No AI bezier curve distortion.
`design-engineering`	`polish:`, `emil:`	Emil Kowalski polish philosophy: 9 interactive states (default, hover, active, focus, disabled, loading, error, success, empty), micro-interactions.
`animation-builder`	`animate:`	Builds purposeful web animations from scratch with proper easing curves, spring physics, and reduced-motion fallbacks.
`animation-performance`	`jank:`, `fps:`	Maintains 16.6ms frame budget (60fps). Restricts animation to composite-only GPU properties (transform, opacity). Eliminates layout thrashing.

The Anti-AI SOP Quality Standard

The Anti-AI SOP suite acts as a ruthless filter across all text, code, vectors, and UI layouts:

BANNED AI TELLS (PURGED IMMEDIATELY)

- **Conversational Throat-Clearing:** "Certainly!", "I'd be happy to", "As an AI language model..."
- **AI Vocabulary:** "Crucial", "Vital", "Delve", "Tapestry", "Testament", "Beacon", "Landscape".
- **Color Clichés:** Anthropic cream (`#F4F1EA`) + terracotta (`#D97757`), near-black + acid-green.
- **UI Widgets:** Pill badges above headlines, uppercase tracked eyebrow tags, 3-card centered icon rows.
- **Chrome:** Soft grey card shadows, gradient background blobs, "→" appended to every button.
- **Vectors:** 64-node wandering bezier curves, floating decimals, mismatched stroke weights.

REAL ENGINEERING REPLACEMENTS

- **Prose:** Direct, declarative statements. Conclusion first, backed by numbers and exact file references.
- **Palette:** Derived directly from the subject's materials, physical place, or authentic photography.
- **Layout:** Asymmetric editorial grids, dense spec sheets, or blueprint compositions.
- **Surfaces:** 1-step tone differentiation (e.g. 3% background delta) or sharp 1px borders instead of drop shadows.
- **Icons:** Razor-sharp 24×24 vectors with 1.5px stroke snapped to whole pixels.

Category Deep-Dives: Backend & Debugging

The Backend Subsystem: Karpathy Simplicity & TDD

The backend subsystem is built on Andrej Karpathy's core observation: **LLMs tend to overcomplicate code by adding speculative flexibility, unnecessary abstractions, and defensive bloat**. Skill Brain enforces minimal blast radius and strict test-driven development.

Skill Name	Primary Trigger / Command	Core Capability & Engineering Benefit
`backend-planner`	`$plan-backend`, `plan-api:`	**Pre-Build Backend Blueprint:** Entity models, responsive API contracts, tracer-bullet ticket graph, and TDD plan.
`responsive-contract-enforcer`	`contract:`, `api-layout:`	**Mandatory Backend UI Contract:** Eliminates hardcoded pixel widths in API responses; requires aspect ratios & srcsets.
`karpathy-coder`	`surgical:`, `simple:`	Minimal blast radius. No unrequested features, no single-use abstractions, no premature configurability. Standard library first.
`tdd-implementer`	`tdd:`, `test-first:`	Red-Green-Refactor loop. Write failing test first, witness failure, write minimal code to pass, refactor cleanly.
`spec-writer`	`to-spec:`	Synthesizes conversation into a formal technical spec: problem, scope, functional requirements, and error states.
`ticket-breaker`	`to-tickets:`	Deconstructs specs into dependency-ordered tracer-bullet tickets with explicit acceptance criteria.
`codebase-designer`	`deep-module:`	John Ousterhout deep module design: simple interfaces hiding complex implementations. Minimizes caller cognitive load.
`architecture-improver`	`audit-arch:`	Scans repository for circular dependencies, god modules, and leaking abstractions; generates visual improvement plan.
`swift-writer`	`swift:`	Modern Swift 6: compile-time data race safety, `@MainActor`, value-type modeling, Swift Testing framework.
`mcp-builder`	`mcp:`	Designs Model Context Protocol tool servers (FastMCP / TypeScript SDK) with clean schemas and deterministic outputs.

The Debugging Subsystem: Planning Gate & Evidence

Debugging in Skill Brain follows an uncompromising Iron Law: **You are strictly forbidden from modifying code until the root cause is isolated with verifiable evidence**.

The Mandatory Diagnostic Plan Template (/plan-debug)

```
### 🔍 Diagnostic Plan
1. **Symptom:** Unhandled rejection: JWT signature expired during token refresh.
2. **Reproduction:** `npm test tests/auth/refresh.spec.ts`
3. **Candidate Hypotheses:**
   • H1: Token expiry timestamp compared using client local time instead of UTC.
   • H2: Refresh token rotation race condition when two requests fire concurrently.
4. **Investigation Steps:**
   [ ] Inspect timestamp parsing at `src/auth/jwt.ts:42`
   [ ] Add isolated regression test reproducing clock drift
5. **Next Step:** User confirmation required to proceed with investigation.
```

Skill Name	Primary Trigger / Command	Core Capability & Engineering Benefit
`debug-planner`	`$plan-debug`, `plan-fix:`	**Pre-Investigation Blueprint:** Gathers symptom, deterministic reproduction command, and candidate hypotheses before code touch.
`root-cause-debugger`	`$debug`, `debug:`	Finds root cause FIRST. Symptom fixes are banned. Gathers evidence, forms ONE falsifiable hypothesis, proves or disproves it.
`bug-diagnoser`	`diagnose:`	Systematic diagnosis for hard bugs and regressions: git bisect automation, temporary targeted logging, minimal reproductions.
`test-verifier`	`$verify`, `verify:`	Evidence before assertions. Runs actual test commands and pastes raw CLI stdout/stderr. "It should work" is rejected as proof.
`code-reviewer`	`review:`	Dual-axis review: Standards compliance (repo rules, clean code) vs. Spec compliance (did we actually build what was requested?).

Category Deep-Dive: Git, Commits & Branch Lifecycle

A core foundation of Skill Brain is that **Git history is a first-class engineering deliverable**. Monolithic, ambiguous, or AI-sloppy commits ("update files", "misc improvements") are strictly prohibited. The Git subsystem enforces automated multi-commit splitting based on domain blast radius, strict branch naming hygiene, and automatic Virtual Memory phase synchronization.

The Git Master Subsystem

Conventional Commits Standard

Every commit generated by Skill Brain follows strict structural guidelines:

- **Header:** `<type>(<scope>): <imperative summary>` (≤ 72 characters, lowercase verb).
- **Types:** `feat`, `fix`, `refactor`, `perf`, `test`, `style`, `docs`, `chore`, `phase`.
- **Body:** Concise bullet points explaining *what* changed and *why*.
- **Footer:** Immutable CLI verification proof (e.g., `Verified: pytest passed (14 tests)`).

Branch Taxonomy & Hygiene

All feature work must take place on dedicated branches with kebab-case taxonomy:

- `feat/<feature-name>` — User-facing capabilities and modules.
- `fix/<issue-name>` — Bug fixes, crashes, and regression patches.
- `phase/<N>-<name>` — Direct correlation with a memory phase.
- `refactor/<target>` — Architectural improvements without feature changes.
- **Rebase Workflow:** Branches sync via `git pull --rebase origin main` for linear history.

The Multi-Commit Splitting Engine (`git-atomic-committer`)

When changes span multiple domains or exceed change size thresholds, Skill Brain automatically splits the changeset into **2 or more cohesive atomic commits**:

Example: Workspace diff contains API changes, test suite, and UI styling # Commit 1: Core Domain Logic git add src/auth/service.py src/auth/tokens.py git commit -m "feat(auth): implement sliding window JWT refresh rotation\n\n- Rotate token on each verified exchange\n- Invalidate expired refresh tokens with 401\n\nVerified: pytest tests/test_auth.py\nRefs: Phase 2" # Commit 2: Test Suite & Fixtures git add tests/test_auth.py git commit -m "test(auth): add unit test coverage for token expiry edge cases\n\n- Verify token rotation lifecycle\n- Verify rejection of revoked refresh tokens\n\nVerified: pytest passed (10 passed)\nRefs: Phase 2" # Commit 3: UI Feedback & Tokens git add src/ui/components/SessionBanner.tsx git commit -m "feat(ui): display session expiration warning banner\n\n- Add toast trigger at 60s to token expiry\n- Adhere to 44px touch targets on mobile\n\nRefs: Phase 2"

Virtual Memory Phase Commit Integration (`git-phase-committer`)

Whenever a phase finishes in `.agents/memory/phases/`, the agent triggers `git-phase-committer` before proceeding. It commits application code, test fixtures, and memory markdown files in a structured 3-step sequence, embedding the exact CLI verification evidence into the git record.

Skill	Command / Trigger	Operational Role
`git-master`	`$git`, `git:`	Apex Git orchestrator: Coordinates atomic commit generation, branch management, and phase commit gates.
`git-atomic-committer`	`$commit`, `commit:`	Inspects diffs, enforces multi-commit splitting across domain boundaries, writes Conventional Commit messages.
`git-branch-manager`	`$branch`, `branch:`	Enforces strict kebab branch taxonomy, stashes dirty trees, conducts rebase syncs, and prunes merged branches.
`git-phase-committer`	`$phase-commit`	Automatic Virtual Memory Phase commit hook: stages and commits phase code, tests, and memory status sequentially.

Real-World Operations Walkthrough

Scenario 1: Dedicated Pre-Build UI Planning

User Prompt: `/plan-frontend build a responsive analytics dashboard and metrics stream.`

EXECUTION STEPS UNDERTAKEN BY FRONTEND PLANNER

1. **Planning Activation:** Recognizes `$plan-frontend` → activates `frontend-planner`.
2. **Subject Inventory:** Queries domain context — extracts high-frequency latency telemetry and audit trails.
3. **Dual-Viewport Responsive Architecture:**
   - **Phone (320px–480px):** `phone-layout-master` specifies single metric cards, bottom action sheet for filter drawers, and `100svh` viewport bounding.
   - **Desktop (1024px–1440px+):** `desktop-ui-architect` specifies 3-pane workstation with container queries (`@container`) and sticky tabular data streams.
4. **Fluid Clamp Scale:** Establishes mathematical typography scale using Technical Swiss fonts (`typography-master`).
5. **User Approval Gate:** Presents the complete UI Architecture Plan and prompts: "*Shall I proceed to build?*"

Scenario 2: Diagnosing an Intermittent Database Timeout

User Prompt: `/debug our checkout endpoint intermittently throws 504 Gateway Timeout under moderate load.`

DIAGNOSTIC GATE FOLLOWED BY THE SYSTEM

1. **Planning First:** The agent activates `debug-planner` and does **NOT** modify database connection pools or query timeouts.
2. **Diagnostic Plan Presentation:** Outlines observed symptoms, logs to inspect (`src/db/pool.ts`), and candidate hypotheses (N+1 query vs. unreleased transaction lock).
3. **User Approval:** User types "*Proceed*".
4. **Investigation:** Adds temporary profiling logging around transaction blocks. Discovers an open transaction inside a third-party payment gateway HTTP call.
5. **Root Cause Stated:** "*Root cause: Database connection held open across external Stripe network call at checkout.ts:88.*"
6. **Surgical Fix:** Moves transaction boundary to commit order state BEFORE making external HTTP call (`karpathy-coder`).
7. **Evidence Verification:** Runs load test script (`test-verifier`), pasting stdout showing 50 concurrent checkouts completing in 140ms with 0 timeouts.

Scenario 3: Adding a Multi-Phase Feature with Virtual Memory

User Prompt: `/memory we need to build an OAuth2 Google and GitHub authentication system with role-based permissions.`

VIRTUAL MEMORY LIFECYCLE

1. **Memory Master Activation:** `memory/phase-planner` creates `.agents/memory/plans/main-plan.md`.
2. **Phase Decomposition:** Decomposes into 3 distinct files:
   - `phase-01-oauth-providers.md` (Skills: `backend/spec-writer` + `backend/tdd-implementer`)
   - `phase-02-rbac-schema.md` (Skills: `backend/domain-modeler` + `backend/codebase-designer`)
   - `phase-03-session-middleware.md` (Skills: `backend/karpathy-coder` + `debugging/test-verifier`)
3. **Live Status Dashboard:** Prepend status board in chat showing Phase 1 in progress.
4. **Phase Execution & User Approvals:** Each phase runs through `[APPLY] -> [EXECUTE] -> [VERIFY] -> [FINISH]` with user confirmation at each checkpoint.
5. **Final Archival:** When all 3 phases pass tests, the system archives the plan to `.agents/memory/archive/2026-11-02-oauth2/` and cleans active phases.

Complete Skill Catalog Index (54 Skills)

Category	Skill Name	File System Location

Orchestrators & Planners	`supreme-orchestrator`
	`frontend-master`
	`frontend-planner`
	`backend-master`
	`backend-planner`
	`debug-master`
	`debug-planner`
	`anti-slop-planner`
	`architecture-master`
	`memory-master`
	`git-master`
	`git-atomic-committer`
		`.agents/skills/orchestrators/supreme-orchestrator/SKILL.md`
		`.agents/skills/orchestrators/frontend-master/SKILL.md`
		`.agents/skills/frontend/frontend-planner/SKILL.md`
		`.agents/skills/orchestrators/backend-master/SKILL.md`
		`.agents/skills/backend/backend-planner/SKILL.md`
		`.agents/skills/orchestrators/debug-master/SKILL.md`
		`.agents/skills/debugging/debug-planner/SKILL.md`
		`.agents/skills/anti-ai-sop/anti-slop-planner/SKILL.md`
		`.agents/skills/orchestrators/architecture-master/SKILL.md`
		`.agents/skills/orchestrators/memory-master/SKILL.md`
		`.agents/skills/orchestrators/git-master/SKILL.md`
		`.agents/skills/git/git-atomic-committer/SKILL.md`

Category	Skill Name	File System Location

Frontend	`phone-layout-master`
	`desktop-ui-architect`
	`dynamic-ui-engine`
	`design-taste-frontend`
	`ui-taste-enforcer`
	`ui-design-principles`
	`typography-master`
	`svg-icon-craftsman`
	`animation-builder`
	`animation-reviewer`
	`animation-auditor`
	`animation-performance`
	`animation-opportunities`
	`apple-feel`
	`mobile-native-feel`
	`ui-stress-tester`
	`ui-library-picker`
	`ui-prototyper`
	`design-engineering`
	`canvas-art`
	`theme-system`
	`web-artifact-builder`
	`webapp-tester`
		`.agents/skills/frontend/phone-layout-master/SKILL.md`
		`.agents/skills/frontend/desktop-ui-architect/SKILL.md`
		`.agents/skills/frontend/dynamic-ui-engine/SKILL.md`
		`.agents/skills/frontend/design-taste-frontend/SKILL.md`
		`.agents/skills/frontend/ui-taste-enforcer/SKILL.md`
		`.agents/skills/frontend/ui-design-principles/SKILL.md`
		`.agents/skills/frontend/typography-master/SKILL.md`
		`.agents/skills/frontend/svg-icon-craftsman/SKILL.md`
		`.agents/skills/frontend/animation-builder/SKILL.md`
		`.agents/skills/frontend/animation-reviewer/SKILL.md`
		`.agents/skills/frontend/animation-auditor/SKILL.md`
		`.agents/skills/frontend/animation-performance/SKILL.md`
		`.agents/skills/frontend/animation-opportunities/SKILL.md`
		`.agents/skills/frontend/apple-feel/SKILL.md`
		`.agents/skills/frontend/mobile-native-feel/SKILL.md`
		`.agents/skills/frontend/ui-stress-tester/SKILL.md`
		`.agents/skills/frontend/ui-library-picker/SKILL.md`
		`.agents/skills/frontend/ui-prototyper/SKILL.md`
		`.agents/skills/frontend/design-engineering/SKILL.md`
		`.agents/skills/frontend/canvas-art/SKILL.md`
		`.agents/skills/frontend/theme-system/SKILL.md`
		`.agents/skills/frontend/web-artifact-builder/SKILL.md`
		`.agents/skills/frontend/webapp-tester/SKILL.md`

Backend	`responsive-contract-enforcer`
	`karpathy-coder`
	`tdd-implementer`
	`implementation-planner`
	`spec-writer`
	`ticket-breaker`
	`spec-implementer`
	`domain-modeler`
	`codebase-designer`
	`architecture-improver`
	`swift-writer`
	`pr-writer`
	`mcp-builder`
		`.agents/skills/backend/responsive-contract-enforcer/SKILL.md`
		`.agents/skills/backend/karpathy-coder/SKILL.md`
		`.agents/skills/backend/tdd-implementer/SKILL.md`
		`.agents/skills/backend/implementation-planner/SKILL.md`
		`.agents/skills/backend/spec-writer/SKILL.md`
		`.agents/skills/backend/ticket-breaker/SKILL.md`
		`.agents/skills/backend/spec-implementer/SKILL.md`
		`.agents/skills/backend/domain-modeler/SKILL.md`
		`.agents/skills/backend/codebase-designer/SKILL.md`
		`.agents/skills/backend/architecture-improver/SKILL.md`
		`.agents/skills/backend/swift-writer/SKILL.md`
		`.agents/skills/backend/pr-writer/SKILL.md`
		`.agents/skills/backend/mcp-builder/SKILL.md`

Category	Skill Name	File System Location

Debugging	`root-cause-debugger`
	`bug-diagnoser`
	`test-verifier`
	`code-reviewer`
	`issue-triager`
		`.agents/skills/debugging/root-cause-debugger/SKILL.md`
		`.agents/skills/debugging/bug-diagnoser/SKILL.md`
		`.agents/skills/debugging/test-verifier/SKILL.md`
		`.agents/skills/debugging/code-reviewer/SKILL.md`
		`.agents/skills/debugging/issue-triager/SKILL.md`

Architecture	`subagent-executor`
	`git-isolator`
	`large-task-mapper`
	`research-agent`
	`setup-wizard`
	`skill-creator`
		`.agents/skills/architecture/subagent-executor/SKILL.md`
		`.agents/skills/architecture/git-isolator/SKILL.md`
		`.agents/skills/architecture/large-task-mapper/SKILL.md`
		`.agents/skills/architecture/research-agent/SKILL.md`
		`.agents/skills/architecture/setup-wizard/SKILL.md`
		`.agents/skills/architecture/skill-creator/SKILL.md`

Memory	`phase-planner`
	`session-continuity`
	`context-handoff`
		`.agents/skills/memory/phase-planner/SKILL.md`
		`.agents/skills/memory/session-continuity/SKILL.md`
		`.agents/skills/memory/context-handoff/SKILL.md`

Anti-AI SOP	`anti-slop-enforcer`
	`text-tone-enforcer`
	`ui-tells-eliminator`
	`anti-ai-sop-agent`
		`.agents/skills/anti-ai-sop/anti-slop-enforcer/SKILL.md`
		`.agents/skills/anti-ai-sop/text-tone-enforcer/SKILL.md`
		`.agents/skills/anti-ai-sop/ui-tells-eliminator/SKILL.md`
		`.agents/agents/anti-ai-sop-agent/SKILL.md`

Git	`git-master`
	`git-atomic-committer`
	`git-branch-manager`
	`git-phase-committer`
		`.agents/skills/orchestrators/git-master/SKILL.md`
		`.agents/skills/git/git-atomic-committer/SKILL.md`
		`.agents/skills/git/git-branch-manager/SKILL.md`
		`.agents/skills/git/git-phase-committer/SKILL.md`