CLAUDE.md — Operating Instructions for Agents in This Repo

This repository is the **human-facing, concept-first curriculum** for the Dream pursuit doctrine. It is a documentation repo — no deployable code, no build system beyond the optional render scripts. Every change here is a change to teaching material, so the bar is pedagogical quality and doctrinal accuracy.

`AGENTS.md` is the cross-tool agent contract for this repo — the tool-agnostic operating instructions any agent (Claude, Codex, Copilot, a CI-driven agent) reads before touching it. This CLAUDE.md is the project-specific view; read both, and keep both current when the repo changes.

The two layers (the core rule)

Everything in this repo lives in one of two layers:

1. **The doctrine** — the durable concepts: where the money lives, how solicitations work, the pursuit pipeline, gates and governance, scoring and price-to-win, the ORBITAL structure, the Dream → ORBITAL → World lifecycle. These are the source of truth and they **must not change** with the tooling.
2. **The tools** — software, products, portals, AI systems. The tools layer is **referenced conceptually only**, never taught as a dependency.

The stack-agnostic rule: no course material may require knowledge of a specific product or system. This includes ERPNext, Dream Dash, Claude, the DNA repository internals, dream-track/catcher/analyzer, or any other tool. You MAY say "a system of record" or "an AI layer that encodes this doctrine." You may NOT make the course depend on any of them. If an edit would make a lesson unteachable without a product, the edit violates the rule — see `doctrine/08-tools-change-concepts-dont.md`.

What changes where

- **The doctrine never changes to chase the stack.** When the software layer changes — a new portal, a new AI capability, a renamed product — **the doctrine does not get rewritten.**
- **When the stack changes, update the align map, never the doctrine.** The `align/` directory is the fully-populated, DNA-version-pinned concept system fusion map: the concept column is durable, the DNA/stack column is swappable, and it now includes the Shipley×DNA fusion map, the DNA corpus index, and the book canon. If a concept gains a new machine-side encoding or a tool mapping changes, that change belongs in `align/`, not in `doctrine/`.
- **Literacy facts must be accurate.** The `literacy/` layer is where government facts live (UEI, CAGE, NAICS, SAM.gov, Grants.gov, FAR,

SBIR/STTR, HUBZone, SBA). If you are not certain of a fact, state it generally rather than inventing specifics. Accuracy is the point of this layer.

- **The teaching case is the shared anchor.** `case-study/ravonics.md` is the same canary company the machine side uses for its golden-thread doctrine validation. Keep its published facts (certifications, identifiers, NAICS codes, partner) consistent with what is documented in the wider ecosystem; do not invent private details.

Structure

- `doctrine/` — one concept per file, each ending with a 3-question self-check, ~600–1200 words, inviting tone.
- `literacy/` — plain-language reference: glossary, acronym decoder, RFP/NOFO reading maps, money-flow map.
- `modules/` — session-by-session teaching plans: `module-template.md` is the contract; `shared-core.md` is the six-session spine; `strategic-initiative.md`, `mba.md`, `mpa.md` are the audience tracks.
- `course/` — the 14-week syllabus, the assessment rubric, the capstone brief.
- `case-study/` — the Ravonics teaching case.
- `align/` — the fully-populated concept system (fusion) map, DNA-version-pinned; the only place in the repo that knows what the tools are.
- `render/` — `build-docx.sh` (required to work) and `build-pdf.sh` (best-effort) produce documents under `dist/`.

Editing guidelines

- **Keep the tone.** Inviting, plain-language, concrete examples, short paragraphs, headings, diagrams. This is a college course, not a technical manual. Match the voice of the existing files.
- **Keep the pipeline in the room.** Every concept connects back to the pursuit pipeline (`doctrine/03`). Preserve that connective tissue.
- **Every doctrine doc ends with a self-check.** If you add a doctrine doc, it must end with a 3-question self-check.
- **Keep the manifest honest.** `manifest.yaml` registers every document with its path, title, audience, and concept. When you add, rename, or remove a doc, update the manifest in the same change.
- **Bump VERSION** for meaningful content changes (semver: patch for fixes, minor for additions).
- **Render after you edit.** Run `render/build-docx.sh` (and `build-pdf.sh` if possible) so the `dist/` output stays in sync with the markdown. The build scripts must remain idempotent and must not fail the repo.
- **Small, committed slices.** Commit and push coherent slices with clear messages. Do not sit on uncommitted work.

Definition of done

A content change is done when: (1) the markdown is correct and consistent with the doctrine; (2) `manifest.yaml` reflects the actual file list; (3) VERSION is bumped if warranted; (4) the render scripts run and produce `dist/` output; and (5) the change is committed and pushed. This is a docs repo — no pipeline, no deployment; the push is the release.