CaRMS Project Mastery Plan (Updated)

Last update: 2026-02-12 — semantic search foundation shipped.

What changed in this update

- Added pgvector-backed table `gold_program_embedding` via Alembic migration `20260212_0002`.
- New Dagster asset `gold_program_embeddings` that encodes `gold_program_profile.description_text` with `all-MiniLM-L6-v2`.
- New FastAPI endpoint `POST /semantic/query` for semantic search (optional LangChain QA summary when `OPENAI_API_KEY` is set).
- API contract + README refreshed to include the semantic endpoint and new gold table.
- Dependencies updated to include `langchain`, `langchain-community`, and `langchain-openai`.

Workstreams still planned

1) Match simulation / what-if engine (`carms/analytics/simulation.py`, `/analytics/simulate`).

2) Preference modeling (`carms/analytics/preferences.py`, `/analytics/preferences`).

3) AWS deployment path (Terraform for RDS + ECS + ECR + S3; CI plan).

Original mastery plan (for interview prep)

Goal and outcomes

Be able to explain: why the platform exists, how data flows, how one API call is served, how quality/performance/security are enforced, and what to improve in the first 30 days. Target: 6–9 hours.

Phase A — Conceptual understanding (1–2 hours)

1) Read README end-to-end. Output: "System in 90 seconds" note (purpose, layers, tech choices).

2) Read docs in order: `docs/architecture.md`, `docs/data-dictionary.md`, `docs/api-contract.md`, `docs/performance.md`. Maintain a two-column table (claim → where implemented).

3) Write 8–10 architecture bullets (business objective; ingestion/transform; serving/API; security; orchestration; performance; roadmap).

Phase B — Code tracing (2–3 hours)

1) Start at `carms/pipelines/definitions.py` (asset/check registration).

2) Trace one record bronze → silver → gold (inputs, transforms, outputs, edge cases).

3) Trace one API endpoint (`GET /programs`) route → dependency → model → response; repeat quickly for `/programs/{id}`, `/disciplines`, `/pipeline/run`.

4) Draw the data flow: source files → Dagster assets/checks → FastAPI routes; annotate migrations, indexes, auth/rate limiting.

Phase C — Runtime understanding (1–2 hours)

1) Run tests; intentionally break a silver rule; rerun tests; note which fail.

2) Full stack with Docker; hit key endpoints (`/health`, `/programs`, `/disciplines`, `/map/data.json`, `/pipeline/run`) and capture one 4xx.

3) Map Dagster materialization changes to API payload changes (cause/effect).

Phase D — Improvement readiness (1–2 hours)

1) Add one quality check (e.g., duplicate `program_stream_id`, province code validity, non-empty descriptions).

2) Add one performance improvement (e.g., trigram index for ILIKE).

3) Define one DS feature extension (similarity search, preference-aware ranking, scenario simulation) with data/fairness assumptions.

4) Draft "first 30 days" plan (Week 1 setup; Week 2 quality/observability; Week 3 performance benchmark; Week 4 DS prototype).

Self-eval checklist

- Explain bronze/silver/gold with a real row.
- Describe `/programs` filtering/pagination, auth, rate limiting.
- Name one migration and index.
- Run stack and troubleshoot a failed endpoint quickly.

- Propose one quality and one performance enhancement with rationale.
- Describe one DS extension with realistic scope and caveats.