# **Pre-Coding Essentials (Component: crates/vm\_pipeline/Cargo.toml, Version/FormulaID: VM-ENGINE v0) — 48/89**

## **1) Goal & Success**

Goal: Declare the **pipeline** crate that orchestrates the fixed state machine (load→validate→tabulate→allocate→aggregate→apply rules→frontier→ties→label→build result→build run record).

Success: Builds offline, deterministically, and links only to required crates (vm\_core, vm\_io, vm\_algo); respects numeric/order/RNG constraints from platform doc.

## **2) Scope**

In scope: package metadata; dependencies on vm\_core (types/variables/RNG), vm\_io (canonical JSON, loaders), vm\_algo (tabulation/allocation); feature flags if any (e.g., frontier, mmp).

Out of scope: algorithm implementations (live in vm\_algo), report rendering (Doc 7), UI packaging.

## **3) Inputs → Outputs**

Inputs: workspace toolchain & lockfile, the three internal crates above.

Outputs: one lib target exposing pipeline entry points used by CLI/app; it ultimately produces **Result** and **RunRecord** artifacts downstream, per pipeline spec.

## **4) Entities/Tables (minimal)**

## **5) Variables (only ones used here)**

None at manifest level. Numeric/ordering/RNG rules are enforced by called code per Docs **3**/**5**; this crate just depends on them.

## **6) Functions (signatures only)**

None (manifest). Pipeline functions exist in src/\*.rs and map to Doc 5 functions and artifacts (LoadedContext, UnitScores, UnitAllocation, AggregateResults, LegitimacyReport, FrontierMap, TieLog, Result, RunRecord).

## **7) Algorithm Outline (bullet steps)**

Not applicable to the manifest; state machine is fixed in code by this crate and must match Doc 5 order exactly.

## **8) State Flow (very short)**

Compile → expose pipeline API used by CLI/app. At runtime, the state machine follows Doc 5 and produces **Result** and **RunRecord**; **FrontierMap** is optional.

## **9) Determinism & Numeric Rules**

Follow workspace profiles (e.g., codegen-units=1, deterministic builds) and **offline** policy (no network at runtime). Math/ordering/RNG rules live in callee crates per Docs 3/5.

## **10) Edge Cases & Failure Policy**

Dependency drift or feature mismatches that would allow networked crates or floating-point presentation in core must be rejected (keep deps minimal; rely on vm\_io for canonical JSON and hashing).

## **11) Test Checklist (must pass)**

cargo build --locked -p vm\_pipeline succeeds on supported OS/arch.

Pipeline integration tests (in this crate’s src later) produce **Result**/**RunRecord** objects matching Doc 1/5 field expectations.

Order and stop/continue semantics exactly match Doc 5 §2.