# **Lean pre-coding sheet — 81/89**

**Component:** crates/vm\_app/src-tauri/src/main.rs (Tauri backend entry)  
 **Version/FormulaID:** VM-ENGINE v0

## **1) Goal & success**

**Goal:** Stand up the Tauri **backend entry point** that exposes minimal, safe commands to the UI for: loading local inputs, running the pipeline, and exporting reports—**offline and deterministic**.

**Success:** App builds and runs on Win/macOS/Linux; all commands read **only local files**, perform the **Doc 5** pipeline, and write canonical artifacts used by reports; no telemetry/network.

## **2) Scope**

**In:** Tauri main() setup, command registration, error mapping, and safe IPC surfaces to core crates (vm\_pipeline, vm\_report). FS/network policy is enforced (FS scope in tauri.conf.json next file; runtime network **disallowed**).

**Out:** UI (vite/web), icons/config (tauri.conf.json), map assets packaging—handled in files **82–89**.

## **3) Inputs → outputs (with schemas/IDs)**

**Inputs:** User-chosen local paths to DivisionRegistry, BallotTally, ParameterSet, optional Manifest; all are **local** per offline policy.

**Outputs:** Canonical Result and RunRecord (UTF-8, **sorted JSON keys**, LF, UTC) for report rendering.

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

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

No policy variables are **set** here; the backend **echoes** the ParameterSet used for runs and ensures determinism constraints from Doc 3 (ordering/rounding/RNG-seed).

## **6) Functions (signatures only — Tauri commands)**

rust

CopyEdit

#[tauri::command]

fn cmd\_engine\_info() -> EngineInfo; // FormulaID, EngineVersion, targets

#[tauri::command]

fn cmd\_load\_inputs(registry: PathBuf, ballots: PathBuf, params: PathBuf, manifest: Option<PathBuf>)

-> LoadedContextSummary; // echoes IDs/labels

#[tauri::command]

fn cmd\_run\_pipeline(registry: PathBuf, ballots: PathBuf, params: PathBuf, manifest: Option<PathBuf>, out\_dir: PathBuf)

-> RunSummary; // runs Doc 5 state machine, returns {result\_id, run\_id, label}

#[tauri::command]

fn cmd\_export\_report(result\_path: PathBuf, run\_record\_path: PathBuf, out\_dir: PathBuf, fmt: ReportFmt)

-> ReportPaths; // JSON/HTML via vm\_report

#[tauri::command]

fn cmd\_hash\_artifacts(result\_path: PathBuf, run\_record\_path: PathBuf) -> HashPair; // SHA-256

(Commands orchestrate **Doc 5** flow; reports follow **Doc 7**.)

## **7) Algorithm outline (backend)**

Initialize Tauri app; register commands; set panic hook to deterministic errors.

**cmd\_load\_inputs:** validate files exist and are inside allowed FS scope; probe IDs/labels only. (FS scope enforced in tauri.conf.json.)

**cmd\_run\_pipeline:** run **LOAD→VALIDATE→TABULATE→…→BUILD\_RESULT/RUN\_RECORD**; write artifacts canonically (UTF-8/LF/sorted keys/UTC).

**cmd\_export\_report:** generate HTML/JSON strictly from Result/RunRecord (and optional FrontierMap), with one-decimal presentation.

**cmd\_hash\_artifacts:** compute reproducibility hashes for UI verification.

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

UI → backend command → core crates follow **fixed pipeline order**; if **VALIDATE fails**, backend still packages **Invalid** Result/RunRecord (UI shows reasons).

## **9) Determinism & numeric rules**

No runtime network/telemetry; integer/rational comparisons and rounding/ordering rules come from core; RNG only with explicit rng\_seed, echoed in RunRecord.

## **10) Edge cases & failure policy**

**Block** any path outside allowed scope; reject URLs; do not follow symlinks out of scope.

Any network attempt is a **bug**; fail closed. Large map tiles treated as **data only**.

If report inputs missing, render **“Sensitivity: N/A (not executed)”** and proceed (UI text derives from Doc 7).

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

Launch app; invoke cmd\_run\_pipeline on a small Part 0 bundle → Result/RunRecord written; hashes stable across repeats/OS.

cmd\_export\_report yields HTML/JSON with **one-decimal** percents; no external assets loaded.

Attempts to read outside FS scope or any HTTP/DNS → error.