# **Lean pre-coding sheet — 87/89**

**Component:** crates/vm\_app/ui/src/main.ts (UI bootstrap & renderer entry)  
 **Version/FormulaID:** VM-ENGINE v0

## **1) Goal & success**

**Goal:** Bootstrap the UI, wire **offline** data flow to the Tauri backend, and render the fixed **Doc 7** report sections in order with **one-decimal** presentation only (no outcome math in UI).

**Success:** On app launch, UI mounts, calls backend commands to run/inspect a local bundle, and renders from Result/RunRecord (+ optional FrontierMap) only; **no network/CDN** assets; stable behavior across OS.

## **2) Scope**

**In:** App initialization, IPC calls, DOM mounting, formatting helpers (percent to **one decimal**), optional MapLibre glue to read **local** public/map/style.json.

**Out:** Any policy/algorithm computation (lives in core); any remote fetch; packaging/security (handled by Tauri config).

## **3) Inputs → outputs**

**Inputs:** Backend command responses (summaries and artifact paths) and canonical JSON from Result, RunRecord, optional FrontierMap.

**Outputs:** Rendered DOM for **Doc 7A** sections; optional map view sourced from local style/tiles; **no** modification of artifacts.

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

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

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

ts

CopyEdit

async function bootstrap(): Promise<void>; // mount app root, call engine\_info

async function loadBundle(paths: {registry?: string; ballots?: string; params?: string; manifest?: string;}): Promise<LoadedContextSummary>; // via Tauri cmd

async function runPipeline(paths, outDir): Promise<RunSummary>; // returns {result\_id, run\_id, label}

async function readArtifact<T>(path: string): Promise<T>; // fs read via backend; JSON parse

function renderReport(result: Result, run: RunRecord): void; // strictly presentation per Doc 7A order

function formatPercent(numer: bigint, denom: bigint): string; // one-decimal display only

function initMap(containerId: string, styleUrl: string): void; // MapLibre using local style/tiles

These call Tauri commands defined in the backend entry; UI stays read-only.

## **7) Algorithm outline (render flow)**

**bootstrap** → mount #app; query engine\_info.

**loadBundle** (optional) → echo IDs/labels.

**runPipeline** (or open existing artifacts) → obtain paths.

**readArtifact** Result + RunRecord; optionally FrontierMap.

**renderReport** strictly follows **Doc 7A** section order; percentages formatted to **one decimal**; include the mandatory approval-gate sentence when ballot\_type=approval.

If map assets present, **initMap** with **local** style/tiles; otherwise hide map panel.

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

UI never computes outcomes; it **invokes** the backend pipeline and **displays** returned artifacts. Failures surface as banners; on **Invalid** runs, all sections still render with reasons.

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

UI must **not** re-round or double-round data; use raw values from Result and apply presentation rounding **once** (one decimal). No network/CDN; assets are local; timestamps shown as UTC.

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

Missing artifacts → show empty state and instructions to run again; do **not** fetch remotely.

Any attempt to load remote fonts/styles/tiles is a **bug**; remove and use packaged assets.

If JSON parse fails, show a deterministic error with the file path; never continue with partial data.

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

App launches offline; **no HTTP/DNS**; UI renders all **Doc 7A** sections from local artifacts with one-decimal percentages.

Map initializes only with **local** style/tiles; otherwise panel hidden gracefully.