Technical Architecture & Requirements

Implementation State — Embedding Landscape Explorer

Claude (Anthropic) · Sean Patrick Morris

March 2026

This document describes the technical architecture and current implementation state of the Embedding Landscape Explorer. It consolidates the design rationale, as-built technology stack, requirement status, and known gaps into a single reference for technical collaborators.

Architecture

Technology Stack

Component	Technology
3D rendering	Three.js r128
Frontend	Vanilla JavaScript — no framework, no build step
State management	AppStore — custom pub/sub (shared/store.js)
Routing	Hash-based (shell.js) — #/landscape, #/discovery
Backend	FastAPI (uvicorn), Python
Embedding source	GloVe 300d, PCA to 256d
Dimensionality reduction	UMAP (seed: 21)
Density estimation	scipy.stats.gaussian_kde
Voronoi	scipy.spatial.Voronoi
Generative decoding	Anthropic API (claude-haiku-4-5)
Export	matplotlib (topography diagrams)
Dev server	python -m http.server 3000

File Structure

File	Purpose
frontend/main.js	Three.js scene, terrain, navigation, all Landscape features
frontend/shell.js	Hash router, nav bar, panel toggle toolbar
frontend/styles.css	All styling including panel system
frontend/index.html	Entry point, all panel HTML
frontend/shared/data.js	DataLayer — cached API fetches
frontend/shared/store.js	AppStore — reactive pub/sub state
frontend/shared/export.js	ExportUtils — JSON/CSV downloads
frontend/pages/discovery.js	Discovery page module

backend/app/main.py	FastAPI server, all /api/* endpoints
scripts/terrain_config.py	UMAP seed and pipeline constants
scripts/*.py	22-step pipeline + discovery tools

Data Flow

The pipeline is linear: seed words → GloVe expansion → nine-way merge → PCA embedding → UMAP projection → density/gradient fields → attractors → deserts → Voronoi → render data → basins → concept candidates. Each step reads the output of the previous step. The pipeline is automated by run_pipeline.ps1 with a -Downstream flag to skip vocabulary generation.

Panel System

All floating panels are draggable via title bar handles. Panels are contextual per page: Landscape shows Research, Path, Location, Map, Notes, Journal; Discovery shows Probe, Layers, Desert, Absence, Cross, Journal. The Journal panel is top-level in the DOM and accessible from both pages. Panel visibility is driven by body[data-page] CSS selectors. Toggle state tracked via .ptog-on class. All panels use [hidden] { display: none !important; } for correct hide behaviour.

Implementation Status Summary

The following summarizes the requirement status across all phases. For the complete per-requirement audit with file locations and deviation notes, see COMPLETE_REQUIREMENTS.md in the project repository.

Phase	Total	Done	Partial	Not Started
P0 — Terrain Surface	6	6	0	0
P1 — Attractor/Basin	6	5	0	1
P2 — Domain Colour/Camp	7	6	1	0
P3 — Concept Paths	6	5	1	0
D0 — Shared Infrastructure	9	5	4	0
D1 — Walkability	8	4	2	2
D2 — Desert Mapping	7	5	1	1
D3 — Probes	7	4	2	1
D4 — Atmosphere/Layers	9	2	2	5
D5 — Generative Decoding	7	7	0	0
D6 — Voronoi/Absence	6	5	1	0
D7 — Export	7	3	2	2
Discovery Acceleration	20	20	0	0
Security	10	7	1	2
Startup/README	16	13	1	2
Standalone Tools	15	14	1	0

Key Gaps

The most significant incomplete areas:

Atmosphere rendering (D4): Multi-layer position spheres, filaments, contextual variant convex hulls, and shader-driven scan animation are not yet implemented. The data pipeline (generate_contexts.py, compute_layer_field.py) is complete — this is a frontend rendering task.

Ground-level indicators (D1-R01, R02): Contour texture overlays and gradient flow chevrons are not started. These improve walkability at close range.

Export overlays (D7-R02 through R05): Desert halftone, annotation overlay, and Voronoi overlay on fabrication diagrams are not started.

Desert hover tooltip (D2-R04): Real-time desert distance feedback on terrain hover is not started.

Security

Secure coding is applied throughout. Key measures: XSS sanitization on all concept labels (SEC-01), export path containment with Windows-compatible case-insensitive checks (SEC-02), query input length limiting at 500 characters (SEC-03), LLM API key isolated in .env and proxied through FastAPI (SEC-07), imported annotation files validated against schema (SEC-08). WebSocket validation (SEC-04) and Voronoi Web Worker isolation (SEC-09) are not implemented but are not needed at current scale.

Performance

The rendering budget: approximately 17,000–32,000 triangles across ~10 draw calls. This is well within budget for any modern GPU. The primary performance consideration is data loading (8,735 concepts with metadata) and initial UMAP projection (handled by the pipeline, not at runtime).

Element	Triangles	Draw Calls
Terrain mesh	5,000–10,000	1
Concept spheres (instanced)	2,000–4,000	1
Basin fill + boundaries	~12,000	2
Atmosphere (when implemented)	6,000–12,000	3
Concept path tube	500–1,000	1
UI overlays	HTML/CSS	0
Total	~17,000–32,000	~10

Configuration

UMAP seed: 21 (scripts/terrain_config.py). Changing the seed regenerates the entire terrain and invalidates all planted camps. Desert threshold default: 0.25 for dig site enumeration. LLM description desert gate: 0.02 (no generation below this). Cross-domain pair filter: cosine similarity > 0.85 rejected as synonyms. Probe steps: 30 default for cross-domain probes.

Environment variables: ANTHROPIC_API_KEY (required for generative decoding, optional for all other features), PORT (default 3000 for frontend, 8000 for backend).