Artifact: Strategic Gap Analysis & Scope Governance Protocol

Project Title: The Iron Road: A Technopedagogical Feasibility Study

Context: LDT Master's Portfolio – Strategic Analysis Phase

I. Executive Summary: The Vision-Execution Gap

This document functions as a "Scope Governance" instrument for the *Iron Road* initiative. The project began with a theoretical "White Paper" identifying a critical "Edutainment Gap" in the current LMS market . However, the subsequent design phase encountered a critical, isomorphic crisis: a "Vision-Execution Gap" .

Driven by the generative capabilities of AI, the project scope expanded rapidly into a "Massively Multiplayer" (MMO) concept that exceeded the production capacity of a solo developer . To resolve this, this analysis operationalizes a "Braking System"—a governance protocol designed to align the project's ambition with available resources . This document validates the strategic pivot from a theoretical "Commercial Launch" to a concrete "Vertical Slice Prototype," prioritizing pedagogical validity over technical scale .

II. Methodology: The Psychological Journey of Scope Governance

The gap analysis was conducted using a "Forensic Audit" methodology to bridge the psychological distance between the designer's intent and the project's reality.

1. The Trap: AI-Induced Scope Creep

The initial phase was characterized by "AI Hallucination of Success," where the ease of generating complex ideas (e.g., GPS-integrated narrative nodes) masked the extreme difficulty of implementation . This created a "Dopamine Trap," where the designer focused on the *potential* of the software rather than the *actuality* of the build .

2. The Pivot: From "Dreaming" to "Designing"

The correction mechanism involved a shift in professional identity from "Visionary" to "Engineer." This required acknowledging that "Coal" (Designer Effort/Time) is a finite resource . The analysis moved from a generative mode ("What could this be?") to a subtractive mode ("What can be built in 4 hours?") .

3. The Outcome: Vertical Slice Architecture

The result is a "Vertical Slice" strategy—a standard Game Development practice where a single, thin cross-section of the game is built to perfect completion, rather than attempting to build the entire world at once .

III. Gap Analysis Matrix: Vision vs. Reality

The following matrix filters the "Probabilistic Future" (Vision) into "LDT Evidence" (Reality), establishing the specific artifacts that will serve as the bridge for the portfolio.

Design Component	The Vision (Source: White Paper)	The Reality (Source: Codebase Audit)	The Portfolio Bridge (Validated Artifact)
Core Mechanic	"Operationalized Kinetic Ecosystems" modeling the physics of learning via GPS.	A Rust/Bevy codebase featuring velocity, friction, and "coal" structs .	Proof of Concept: A technical demo showing that inputting text decreases "coal" (resource) and increases "train velocity" (progress).
AI Integration	"Weigh Station" using Gemma 3 for real-time safety lockouts.	No active LLM integration; placeholder logic exists for text analysis .	Architecture Spec: A system diagram citing the "Privacy Moat" design, validating the *intent* of data sovereignty without requiring live implementation .

Environment	**"GPS Node Gardens" and "Digital Twins" of the Purdue Campus.**	**A flat 2D plane with basic movement entities (InputPlugin, MovementPlugin) .**	**Simulation: A "Debug Mode" interaction where the user simulates "Arrival at Bell Tower" to validate the logic without hardware dependency .**
Scope & Role	**"Massively Multiplayer Online LitRPG" (MMO) involving campus-wide coalitions.**	**Single-player local instance managed by a solo developer .**	**Vertical Slice: A "Single-Player Journaling Experience" that validates the individual cognitive loop rather than the network .**

IV. The Braking System: Scope Governance Protocols

To ensure the project remains within the "Zone of Proximal Development" for the designer, the following protocols have been established .

Protocol A: The "Dead Man's Switch" (Resource Limit)

- **Concept: A safety mechanism to prevent burnout.**
- **Rule: If a feature cannot be prototyped within a 4-hour window, it is cut from the Implementation list and moved to the "Future Design" specification .**
- **Result: The "Persona Engine" and "GPS Tracking" were successfully deprecated to the design roadmap, preserving the project's timeline .**

Protocol B: The "Signal Tower" (External Validation)

- **Concept: Relying on objective evidence rather than subjective confidence.**
- **Rule: The portfolio will only claim credit for the *extant* Rust/Bevy ECS System (the "tech spike"). All unbuilt features must be labeled "Theoretical Framework" .**
- **Result: This eliminates the "Presumption of Competence" and replaces it with "Evidence of Design," aligning with Purdue's academic integrity**

standards.

Protocol C: Friction Management (Wizard of Oz)

- **Concept: Reducing technical debt to focus on pedagogical proof.**
- **Rule: Instead of fighting complex API integrations for the demo, the designer utilizes "Wizard of Oz" prototyping (manual input) to simulate the AI response .**
- **Result: This proves the *concept* of the "Mirror AI" works for the learner, without requiring the backend code to be production-ready .**

V. Conclusion: The Value of the Gap

This analysis demonstrates that the value of the *Iron Road* project lies not in the completion of a commercial product, but in the rigorous architectural design of a "Kinetic Learning Ecosystem." By identifying and closing the "Vision-Execution Gap," the project has evolved from a disorganized "dream" into a disciplined "Design Study," fulfilling the core competencies of the LDT program .

```
# Sculpting the Face: UI/UX & Immersion (Legacy)



- [x] **Imagery Injection (The Face)**

  - [x] Generate `train_station_background.webp` for `StudentDashboard` (Used CSS Gradient Fallback).

  - [x] Generate `mystic_fog_background.webp` for `PlayMode` default (Used CSS Gradient Fallback).

  - [x] Update `StudentDashboard` to use the new background with a parallax or slow pan effect.



- [x] **Voice Interface (The Voice)**

  - [x] Create `VoiceInput` component (Microphone button with ripple animation).

  - [x] Integrate `VoiceInput` into `PlayMode` input area.
```

```
  - [x] (Mock) Wire up "Listening" state to visual feedback.


- [x] **Polishing the Body (UI Refinement)**

  - [x] **QuestLog**: Add "Gold/Steampunk" borders and "slide-in" animations.

  - [x] **Typography**: Verify font usage (Inter/Roboto/Cinzel?) for a premium feel.

  - [x] **Transitions**: Ensure page transitions are smooth (fade/slide).


# Project Migration & Audit (Current Status)


- [x] **Technical Audit (The "Mess" Cleanup)**

  - [x] **Disk Usage Analysis**: Identified causes of "Terabyte" bloat (Redundant AI models + Build Artifacts).

  - [x] **ModelManager Repair**: Fixed `ask_pete_server` to persist model state (`models_state.json`) and stop infinite downloads.

  - [x] **Garbage Collection**: Deleted redundant `download_gemma.ps1` and duplicate `Ask_Pete/models` directory.


- [x] **Portfolio Preparation**

  - [x] **Codebase Catalog**: Created `codebase_analysis.md` detailing every function and its philosophy ("Mirror").

  - [x] **Technical Report**: Created `analysis_report.md` for the design document context.


- [ ] **Known Issues for Next Restart**
```

- [ ] **Missing Assets**: `assets/pattern_gears.png` and other heavy media files are missing from `Iron_Road`.

- [ ] **Migration**: Ready to move to Linux/GitHub. `.gitignore` has been updated to prevent large file uploads.

Vision vs. Reality (for Design Doc)

- See `analysis_report.md` for the detailed breakdown.

Ask Pete: The Codebase of a Digital Mirror

Detailed Technical Analysis & Portfolio Catalog

1. The Philosophy: "AI as a Mirror"

The core philosophy of "Ask Pete" is not to create an AI that answers questions, but an AI that acts as a *Mirror* to the user's intent. By using the Socratic method, the system forces the user to articulate their thoughts, thereby increasing their "Cognitive Mass" (understanding). This progress is tracked not by points, but by "Physics"—momentum, resistance, and friction.

---

2. The Cognitive Core (`ask_pete_ai`)

```
*The brain of the operation. Handles thinking, speaking, and remembering.*


#### **Module: `SocraticEngine`** (`src/socratic_engine.rs`)


| Function | Philosophy (Why?) | Mechanism (How?) |

| :--- | :--- | :--- |

| **`respond()`** | **The Reflection Loop.** To ensure the user is heard and questioned, never just "answered." | **Orchestration:**<br>1. **Memory:** Saves input to `ask_pete_db`.<br>2. **Context:** RAG retrieval.<br>3. **Inference:** Calls Gemini or Local Mistral.<br>4. **Mirroring:** Forces response to end in a question. |

| **`post_process_response()`** | **The Socratic Guardrail.** AI wants to be helpful; we need it to be inquisitive. | **Heuristic Filter:**<br>Appends `?` if missing. |


---


### 3. The Physical World (`ask_pete_physics`)


*The Bevy ECS engine that gives weight and consequence to learning.*


#### **Module: `systems`** (`src/systems.rs`)


| Function | Philosophy (Why?) | Mechanism (How?) |

| :--- | :--- | :--- |

| **`calculate_train_velocity()`** | **Momentum is Earned.** You cannot
```

```
move without burning "Coal" (Effort) against "Mass" (Complexity). |
**Equation:** `Velocity = Power / Mass`.<br>Consumes `Coal` resource per
frame. Stalls if empty. |

| **`monitor_cognitive_load()`** | **Balance is Key.** Measures the
friction of learning. | **Simulation:**<br>Oscillates `Germane` load
(focus) using sine waves; decays `Extraneous` load (confusion) over time.
|



---



### 4. The Nervous System (Core & Server)



#### **Crate: `ask_pete_core`**



*The shared DNA of the project. Contains the semantic types used by both
Client and Server.*



* **Philosophy:** "Speak the same language."

* **Mechanism:** Defines struct types like `Quest`, `PlayerCharacter`,
`VirtueTopology`, and `Steam`. By sharing this crate, the frontend (WASM)
and backend (Axum) physically cannot disagree on what a "Quest" is.



#### **Crate: `ask_pete_server`**



*The Grand Central Station. Connects the user to the brain and physics
engine.*
```

```
* **`WeighStationService`:**
  * **Why:** "Words have Mass."
  * **How:** Asks AI to assign `weight (1-100)` to user inputs based on grade level and complexity.
* **`ChatQueueService`:**
  * **Why:** "Thinking takes time."
  * **How:** Async job queue (using `tokio::mpsc`) that holds user requests while the AI thinks, preventing HTTP timeouts.

#### **Crate: `ask_pete_auth`**

*The Gatekeeper.*

* **Philosophy:** "Identity is the first step of the journey."
* **Mechanism:** Middleware that validates Sessions or OAuth tokens (Google) before allowing access to the Physics engine.

---

### 5. The Tools & Workshops

#### **Crate: `ask_pete_trainyard`**

*The Narrative Construction Kit.*
```

* **Philosophy:** "We build our own tracks."

* **Mechanism:** A graph-based data structure helper to organize Story Nodes (`BlueprintStation`) into a cohesive curriculum.

Crate: `ask_pete_node_garden`

The Authoring Tool (Frontend).

* **Philosophy:** "Gardening the Mind."

* **Mechanism:** A dedicated Leptos (Rust-Frontend) application for visually editing the `Trainyard` graphs.

Crate: `ask_pete_researcher`

The Analytics Dashboard.

* **Philosophy:** "Know Thyself."

* **Mechanism:** Visualizes the `ResearchLog` (vectors, virtue maps) so the user can see their own growth patterns.

---

6. The Cleanup Report (System Status)

Actions taken to align code with the "Mirror" philosophy.

1. **Redundancy Removal:**
    * **Deleted:** `download_gemma.ps1` & `Ask_Pete/models/` (Duplicate huge files).
    * **Reasoning:** The `ModelManager` is now the single source of truth.

2. **Logic Repair:**
    * **Fixed:** `ModelManager` persistence (Memory).
    * **Implementation:** Added `save_state()`/`load_state()` JSON logic.

Ask Pete - Technical Analysis & Quality Review

Executive Summary

The project "Ask Pete" is an ambitious hybrid application combining a high-performance Game Engine (Bevy), a Web Server (Axum), a UI Framework (Leptos/React), and Local AI Inference (Candle).

Current Status:

- **Code Quality:** High complexity, but modular (Workspace structure).
- **Disk Usage:** likely caused by redundant AI model downloads and massive Rust build artifacts (`target` directories).

- **Architecture:** "The Armstrong Maneuver" (hybrid ECS + Async) is powerful but fragile due to dependency conflicts (e.g., `winit` versions between Bevy and other crates).

1. Disk Usage Analysis ("The Terabyte Problem")

The excessive disk usage is likely caused by three factors:

A. Redundant Model Downloads

- **The Culprit:** `crates/ask_pete_server/src/services/model_manager.rs`
- **The Bug:** `ModelManager::new()` initializes with an *empty* list of downloaded models. It fails to scan the existing cache on startup.
- **The Result:** Every time you restart the server, it thinks no models exist and triggers a download. While the underlying library (`hf_hub`) tries to cache, the logic flow is flawed and may be creating duplicate hardlinks or just confusing the user with constant "Downloading..." messages.
- **Secondary Culprit:** `download_gemma.ps1` downloads models to `Ask_Pete/models`, while the server downloads to `%LOCALAPPDATA%\askpeet\models`. This creates at least two full copies of 2GB+ files.

B. "Antigravity" & Rust Artifacts

- **Rust Build Files:** The `target` directory in Rust projects can easily grow to 10GB-50GB, especially with workspaces.
- **".antigravity" Files:** The code does not generate these. These are

likely artifacts from the AI coding assistant (me) stored in `.gemini` or accidentally placed in project roots during previous sessions.

2. Code Quality Review

Key Issues

1. **Initialization Logic:** The `ModelManager` does not persist state or check the filesystem, leading to the "Groundhog Day" loop of model downloads.

2. **Hardcoded Paths:** `download_gemma.ps1` uses hardcoded Windows paths, while Rust code uses a mix of environment variables and hardcoded defaults.

3. **Dependency Hell:** Comments in `Cargo.toml` ("The Armstrong Maneuver") indicate active fights with dependency versions (Bevy vs Winit). This makes the project hard to build and maintain.

4. **Complexity:** Running a full ECS (Bevy) ***inside*** a Web Server (Axum) is extremely difficult to synchronize.

3. Remediation Plan (Fixing the Mistakes)

Immediate Fixes (I will perform these)

1. **Fix `ModelManager`:** Update logic to scan `cache_dir` on startup so it knows what models are already there.

2. **Delete Redundant Scripts:** Remove `download_gemma.ps1`.

3. **Clean Build Artifacts:** Safe to run `cargo clean` to reclaim tens of

gigabytes (will need to rebuild next time).

Strategic Changes (For the Design Document)

- **Simplify:** Decouple the Game Engine from the Web Server if possible. Or clarify the boundary.
- **Centralize Data:** Use a single `data` directory for all models, vectors, and DBs.

4. Vision vs. Reality Data

To be used for your new Design Document

Feature	Current Implementation	Vision
Narrative Engine	`GraphManager` (In-memory, basic)	Graph-based story nodes with AI weighing & Physics
AI Integration	`SocraticEngine` + `IronSplit` (Mock/Local)	Full "AI as a Mirror" with reliable local inference
Frontend	Leptos (Rust) & React (Vite) mixed	Unified, premium-feel interactive dashboard
Physics	`ask_pete_physics` (Bevy systems)	"Steam" economy driving learning outcomes