Project Executive Summary: FutureFit Quest System Architecture

Internal Review Document v1.0

To: DeepSeek (Senior Engineering Reviewer), Claude (Independent Reviewer)
From: Gemini 3 (System Architect & Coder)
Date: December 24, 2025
Subject: Architectural Review for FutureFit Quest (v0.1 Core Engine)

1. Executive Overview

FutureFit Quest is a gamified, mobile-first educational application designed to teach AI literacy and career resilience to non-STEM professionals.

The Unique "Dual-Learner" Context:
This project operates under a unique constraint: **The Developer is also the Primary Student.**

1. **Pedagogical Goal:** The developer is building this app to learn full-stack engineering.
2. **User Goal:** The developer is the primary user ("dogfooding" daily) to learn the content (AI skills).
3. **Constraint:** The architecture must be robust enough to support daily learning use without data loss, yet modular enough for a novice developer to implement in "chunks" of ~3 hours/week.

Current Status:
We are rebooting from a failed V1 prototype which suffered from catastrophic state corruption (progress rollbacks) and inconsistent logic. We are currently in the **Architecture Design Phase**, preparing for a clean build of v0.1.

2. Review Protocol & Project Roadmap

2.1 The "Dept. of AI" Collaboration Workflow

We will operate as a synchronous architectural board. For each major phase, the workflow is:

1. **Draft (Gemini 3):** Produces the technical proposal or code.
2. **Technical Audit (DeepSeek):** Reviews for feasibility, logic gaps, state safety, and architectural integrity.

3. **Independent Audit (Claude):** Reviews DeepSeek's findings for blind spots, alternative risks, and user-centric/pedagogical alignment.
4. **Synthesis (Gemini 3):** Integrates feedback into the final action plan for the Human Developer.

2.2 Project Roadmap

We are adopting a "Functionality First, UI Second" approach, broken into isolated implementation chunks.

Phase	Component Focus	Review Gate
Phase 1	**System Architecture (Current Phase)**	**WE ARE HERE**
Phase 2	**Core Engine (Logic & State)** *Chunks 1-6: Data Structures, Persistence, Assessment, Immutable Events, Policy Engine, Session Manager.*	Review required before UI build.
Phase 3	**Interface & Gamification** *Chunks 7-10: Dashboard, Lesson Player, Stronghold, Badges.*	Review required before Backend migration.
Phase 4	**Backend Migration** *Porting v0.1 logic to Node.js/PostgreSQL.*	Final Architecture Review.

3. Architectural Context & Constraints

Reviewers must verify that the proposed design respects these constraints.

3.1 Pedagogical Model (The "Tree & Branch" System)

- **Structure:** Content is a strict hierarchy: **Tree (Domain) → Branch (Topic) → Leaf (Micro-lesson).**
- **Progression Rule (A2 - Branch-Level Strict):**
  - Trees must be unlocked in order (1 → 2 → 3).
  - Branches within a tree must be completed in order (A → B).
  - Leaves within a branch must be completed in order (1 → 2 → 3).
- **The "Stronghold":** A persistent portfolio where users collect artifacts (e.g., "My AI Ethics Code") as they progress.

3.2 Post-Mortem: Why V1 Failed

The previous prototype failed due to **Mutable State Corruption**.

- **Failure:** Opening a branch would reset progress to Leaf 1.
- **Failure:** Completion messages fired prematurely before state was saved.
- **Root Cause:** State was re-calculated from scratch on every render, with no immutable log of events.

- **Design Mandate:** The new architecture **must** use **Event Sourcing (Immutable Attempt Logs)** to guarantee no rollbacks.

3.3 Technical Constraints

1. **Mobile-First Responsive Web:** Must work flawlessly in a mobile browser.
2. **Offline-First (v0.1):** Must run entirely in the browser (localStorage) initially, but the data model must be designed *now* to map 1:1 to PostgreSQL tables for Phase 4.
3. **Daily Stability:** The app is used daily by the developer. State mutations must be **atomic** and **idempotent**.
4. **Input Validation:** For v0.1, open-ended text answers are accepted as "complete" without LLM grading (to reduce complexity), but the *Assessment Engine* must be modular to add LLM grading later.

4. Proposed System Architecture (For Review)

4.1 High-Level Pattern: "The Deterministic Loop"

We propose a **Unidirectional Data Flow** architecture to prevent the V1 "state chaos." User input never mutates state directly; it passes through a strict chain of engines.

```
[UI Layer] -> [Session Manager] -> [Assessment Engine] -> [Progress Service (Immutable Lo
```

4.2 Core Components

1. **Session Manager (The Orchestrator):**
   - The *only* component allowed to coordinate the other engines.
   - Ensures the sequence: `Validate Input` → `Assess` → `Log Attempt` → `Update Derived State` → `Render`.
2. **Assessment Engine (The Validator):**
   - **Input:** User answer + Question Type.
   - **Logic:**
     - *Multiple Choice:* Exact match validation.
     - *Text Input:* Checks for non-empty string (v0.1 rule: accept any non-empty answer).
   - **Output:** Verdict ("correct"/"complete"), Score (0.0-1.0).
3. **Progress Service (The Truth Source):**
   - **Responsibility:** Manages the **Immutable Append-Only Log** (`attempt_events`).
   - **Storage:** Serializes the event log to `localStorage` (v0.1) with checksums.
   - **Derivation:** Calculates "Current State" (XP, Mastery, Unlocked Nodes) *on the fly* from the event log to ensure data integrity.

4. **Policy Engine (The Rule Maker):**
    - **Responsibility:** Pure function that decides "What is the next allowed node?"
    - **Logic:** Enforces the "Branch-Level Strict" progression rules.
    - **Invariant:** A user cannot "jump" to a locked node; the Policy Engine simply will not return it as a valid next step.

4.3 Data Model Strategy

To facilitate the future migration to PostgreSQL, we are treating `localStorage` as a relational DB dump.

- `skillTrees` **(Static):** The read-only graph of content (Trees/Branches/Leaves).
- `attempt_events` **(Immutable):** An array of attempt objects `{ attempt_id, user_id, node_id, timestamp, answer, verdict, score }`. **This is the single source of truth.**
- `user_state` **(Derived):** A cached object `{ current_node_id, unlocked_branches[] }` generated by replaying `attempt_events` through the Policy Engine.

5. Request to Reviewers

DeepSeek (Senior Engineer):

1. **State Safety:** Does the "Immutable Event Log" strategy sufficiently mitigate the "Rollback" bugs experienced in V1?
2. **Component Isolation:** Is the separation between `Assessment` and `Policy` distinct enough to prevent logic coupling?
3. **Migration Risk:** Does this data model actually map cleanly to PostgreSQL, or are we creating a trap for Phase 4?

Claude (Independent Reviewer):

1. **Pedagogical Alignment:** Does this strict logic support the "Dual-Learner" journey, or is it too rigid for a developer-student who might need to test random nodes?
2. **Blind Spots:** Are we over-engineering the "Event Sourcing" for a v0.1 prototype? Is there a simpler path that meets the stability requirement?

End of Executive Summary

⁂

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/collection_4557882a-424d-4f9d-8a21-c0808868c937/efe589ea-37a6-402b-8fc5-e63f9bbe5e1e/Principle-WhatItMeans-WhyItMatters.csv
2. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/collection_4557882a-424d-4f9d-8a21-c0808868c937/35b86843-8793-45ac-a2c8-bf63418fae62/FutureFit-Setup-Guide.md
3. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/collection_4557882a-424d-4f9d-8a21-c0808868c937/b1ed04c6-5908-4e5a-bfa8-ec3b43037947/concept.pdf

4. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/collection_4557882a-424d-4f9d-8a21-c0808868c937/d773b0a9-c208-445f-8c3d-3d03cc9eb8d5/CODIGO-FONTE-INICIAL.pdf
5. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/collection_4557882a-424d-4f9d-8a21-c0808868c937/fc92a267-da51-4211-bc4f-f2b4867160a9/Architecture.pdf