The Problem Breakdown (First Principles)

1. The Fundamental Goal
The user has a Mental Image (A). The Video AI needs a Technical String (B).
Your software's only purpose is to minimize the friction of moving from A → B.
2. The Current Failure (The "Wizard" Fallacy)
You currently try to solve this by inserting a Schema (C) in the middle:
Mental Image (A) → Schema/Form (C) → Technical String (B)

- **The Assumption:** You assume every "Mental Image" (A) can be decomposed into the same "Schema" (C) (Subject + Action + Location).
- **The Reality:** A storyboard (A) *cannot* be decomposed into your Schema (C) because it has no "Action."
- **The Result:** The Schema (C) becomes a **Blocker**, not a bridge. You are forcing the user to play "Translation Games" just to satisfy your UI.

3. The First Principles Solution
To support "Infinite Use Cases," you must remove the Schema (C) from the user's path. The user should never interact with your database fields.
The Correct Flow:
Mental Image (A) → Raw Input → INTELLIGENT INTERPRETER → Technical String (B)

The Architecture Implied by This Analysis

If we accept that "The Schema is the Problem," then the entire "Wizard" architecture in your codebase is technical debt.

1. The Input Mechanism: "Unstructured Ingestion"

- **Principle:** Do not force structure *before* understanding intent.
- **Implementation:** Replace WizardVideoBuilder (and its 4-step process) with a **Single Input Interface**.
  - *Not* a form with labels like "Subject."
  - A raw text area (or audio input) where the user dumps their brain.
  - *Example Input:* "I need a shot of a storyboard panning left."

2. The Processing Mechanism: "Post-Hoc Structuring"

- **Principle:** Structure is an *output* of understanding, not an *input* requirement.
- **Implementation:** The AI (your PromptOptimizationService) receives the raw blob. *It* decides what the structure is.
  - It looks at "Storyboard panning" and decides: *"This is an artifact shot. I need to define camera movement."*
  - It looks at "Man running" and decides: *"This is an action shot. I need to define speed and environment."*
- **Crucial Change:** The validation logic (subject && action) in useQuickFillForm.js must be deleted. It tries to validate the *User's Input* against a rigid model. Instead, you should only validate the *AI's Output* (does it generate a valid video prompt?).

3. The Feedback Loop: "Correction vs. Filling"

- **Principle:** It is easier for a human to *correct* a wrong guess than to *fill* a blank form from scratch.
- **Implementation:** The system immediately generates a candidate prompt. The user then interacts with the **Canvas**, not the **Wizard**.
  - If the AI guessed "Action" where there was none, the user deletes that text in the editor. They don't have to go back a step in a wizard.

Conclusion: The "Anti-Wizard"

The problem isn't that you have "4 boxes" (Narrative, Commercial, etc.). The problem is that you have **boxes at all**.

Your Strategy Pivot:

1. **Delete** StepQuickFill, StepCoreConcept, and StepAtmosphere. They are all "Schema Enforcers."
2. **Build** a "Smart Canvas." The user types *anything*, hits Enter, and the PromptOptimizationService does the heavy lifting of structuring it *behind the scenes*.
3. **Success Metric:** Not "Did the user fill all fields?" but "Did the user hit 'Generate' on the result?"