### **Technical Design Specification: Tri-Modal Constitutional Agent**

Version: 2.1 (Revised)

Date: 2025-06-14

Status: Proposed

This document supersedes v2.0. It details the implementation of the "Plan, Synthesize, Review" cognitive workflow described in Technical Research Report v13.md.

#### **1. Architectural Overview**

The system's core workflow is refactored to enhance its reasoning and generation capabilities for complex tasks.

New Workflow:

Route -> Create Plan -> [Iterative Synthesis Loop] -> Holistic Review -> Constitutional Refinement -> End

This multi-stage process ensures that a structured plan is created before content generation begins, that generation is focused on small, manageable tasks, and that a final review validates the coherence of the complete artifact.

#### **2. Module & Component Specifications**

* 2.1. GraphState Data Structure  
  The GraphState TypedDict is updated to support the new workflow.  
  Python  
  from typing import TypedDict, List, Optional, Dict  
    
  class GraphState(TypedDict):  
   # ... (previous fields: prompt, documents, task\_mode, etc.)  
   prompt: str  
   documents: List  
   task\_mode: str  
   output: str  
   critique: str  
   error: str  
   num\_revisions: int  
   log: List[str]  
    
   # --- NEW FIELDS for v2.1 ---  
   plan: Optional[List[str]] # The structured plan for generation.  
   working\_memory: Optional[Dict] # Stores content for each plan item during synthesis.  
   completed\_plan\_items: List[str] # Tracks progress through the synthesis loop.
* 2.2. Supervisor (supervisor.py)  
  The supervisor's orchestration logic is significantly enhanced.
  + **New Responsibility: Iterative Synthesis Loop:** After the create\_plan node runs, the supervisor will enter a new control loop. It will iterate through the list in state.plan, calling the execute\_synthesis node for each item and updating the state at each step.
* 2.3. Agent Core (agent\_core.py)  
  The node library is refactored to implement the new cognitive workflow. The previous monolithic generate\_\* and reflect\_\* nodes are deprecated in favor of this more specialized set.
  + **New Node: create\_plan(state):** Takes the user prompt and returns a new state with the plan field populated with a list of strings (e.g., section headings).
  + **New Node: execute\_synthesis(state):** Reads the *next* uncompleted item from state.plan. It performs a targeted retrieval and generates content *only for that item*. It returns a new state where working\_memory is updated with the new content and the item is marked as complete.
  + **New Node: assemble\_draft(state):** After the synthesis loop, this node compiles the contents of working\_memory into the output field in the correct order, creating the full draft.
  + **New Node: holistic\_review(state):** Takes the fully assembled state.output and performs a high-level review for narrative flow and consistency, populating the critique field. This critique will then be used in the existing constitutional refinement cycle.

#### **3. Data Flow and Workflow Logic**

The system's execution path is now as follows:

1. **Entry & Routing:** retrieve and route\_task operate as before.
2. **Planning:** The graph transitions to the new create\_plan node. The output is a state containing the generation plan.
3. **Synthesis Loop (Supervisor Logic):** a. The supervisor checks if there are uncompleted items in state.plan. b. If yes, it calls the execute\_synthesis node. c. It receives the updated state and loops back to step 3a. d. If no, the loop terminates.
4. **Assembly:** The graph transitions to the assemble\_draft node to create the final output.
5. **Review & Refinement:** a. The graph transitions to holistic\_review to generate a high-level critique of the full draft. b. The existing constitutional cross-critique cycle is then invoked to refine the draft based on the holistic review's findings.
6. **Finalization:** The final, refined output is saved and presented to the user.