Definitive Master Document: The Living Implementation Blueprint (v2.0 - Final Alpha & Evolution Plan)

Version: 2.0

Date: July 9, 2025

Status: Final, Approved for Implementation

Authored By: Gemini, Senior Software Architect

The master strategic document serving as the single source of truth for all architectural, product, and planning decisions.

Contents

1 Purpose

The master strategic document serving as the single source of truth for all architectural, product, and planning decisions. Part 3 is to be extracted verbatim to form the "Scoped Implementation Plan" for the 'cline' implementation assistant.

2 The Vision & Core Architecture (The "Why")

2.1 Core Philosophy: The 'Truth-Seeking Engine'

The system's fundamental purpose is not to be a passive knowledge base but an active epistemological inquiry engine. It must function as a sophisticated "thought partner" that empowers a discerning user to question established narratives, analyze the evolution of ideas, and form high-conviction opinions. The guiding design principle is to "challenge and guide, not just agree." The engine must surface objective truths, contradictions, and outdated assumptions to prevent the user from operating within an echo chamber.

2.2 Foundational Principle: The 'Timeline Axis'

All knowledge is considered provisional and a function of time. The system's architecture must treat time as a primary dimension of analysis, enabling it to map the historical evolution of a concept. This allows the user to understand not just what is known, but how that knowledge has come to be.

2.3 Core Framework: 'Multi-Axis Analysis'

To achieve a sophisticated analysis that avoids simplistic conclusions, every synthesized "opinion" will be evaluated along four distinct axes:

- **The Temporal Axis**: Pinpointing when a piece of information was considered true and how subsequent findings have altered its meaning.
- **The Credibility Axis**: Evaluating the source, author's track record, potential conflicts of interest, and authority.
- **The Coherence Axis**: Analyzing how the information supports, contradicts, or introduces a novel perspective relative to the user's existing knowledge base.
- **The Impact/Risk Axis**: Assessing the potential second-order consequences and outcomes of acting on the information.

2.4 Core Algorithm: Codified Mental Models

The AI's reasoning process will be explicitly architected to emulate and apply established mental models like First-Principles Thinking, Second-Order Thinking, and Inversion to its analysis. This is a key function of the 'Synthesis Agent' in the worker chain.

2.5 System Architecture: The 'Modular Agent' Workflow

The system's backend is architected as a chain of modular, specialized AI agents orchestrated by a BullMQ task queue. This design ensures:

- **Asynchronous Resilience**: The ability to handle complex, long-running tasks without requiring a persistent user session.
- **Vendor-Agnostic Pluggability**: Each agent acts as a microservice, allowing components to be fulfilled by different vendors (e.g., Gemini, Perplexity) via a central configuration file.
- **Long-Term Evolvability**: The ability to add, remove, or upgrade individual agents as new technologies become available.

3 Product & Feature Requirements (The "What" for Alpha)

3.1 Alpha Persona & Use Case: The "Debate Prep" Scenario

Persona: A "Professional Strategist" who must analyze complex information to build and defend high-stakes arguments.
Primary Workflow: The user uploads an opposing document. The AI's 'first move' is a 'Defensive Analysis' to find weaknesses, followed by an 'Offensive Analysis' that leverages the user's internal library to build a counter-argument.

3.2 Key Deliverable: The Interactive Report

The primary output is a dynamic, modular web report designed for exploration, not just consumption. Key components include:

- **EmbeddedContentViewer**: To view source documents (PDFs, articles) directly in the UI.
- **SourceAttributionWidget**: A visual chart representing the weight of evidence from different sources.
- **InteractiveCitation**: Clickable footnotes that reveal source text, metadata, and links.
- **MultiAxisAnalysisPanel**: A UI section summarizing findings along the axes of Credibility, Coherence, and Impact/Risk.
- **KeyFindingsPanel**: Collapsible sections for "Logical Fallacies Found," "Unsubstantiated Claims," and "Points for Counter-Argument."

3.3 Key Mechanism: The 'Ingestion Sandbox'

External sources discovered during analysis are never automatically added to the user's trusted library. They are placed in a "Review Sandbox," where the user is presented with a comprehensive 'pre-flight analysis report' for each item and retains full curatorial authority with "Add to Library" or "Dismiss" actions.

3.4 Liability & Output Framing

All generated output will be explicitly framed as an "AI-generated analytical opinion for review" and accompanied by standard disclaimers. The AI's persona will be trained to use analytical language rather than making prescriptive recommendations.

3.5 Future Vision (Post-Alpha): Task Management

The architecture should anticipate a V2 feature for integrated task management, allowing users to turn insights from reports into actionable tasks within a project pipeline.

4 The Scoped Implementation Plan

4.1 Phase 1: Foundation & Setup (Target: 2 Days)

[ ] **Task 1.1: Project Initialization & Configuration**

- **Details**: Ensure the Next.js project uses the .env file for all credentials. Create a central configuration file at `lib/config.ts` to export configured singleton instances of the Prisma client, Redis client (ioredis), and Google AI client.
- **Verification & Acceptance Criteria**:
  * 1.1.1: The `lib/config.ts` file exists and successfully exports the required clients without hardcoding any secret keys.
  * 1.1.2: The application can connect to all three services (Supabase, Redis, Google AI) using the configured clients.

[ ] **Task 1.2: Database Setup & pgvector Enablement**

- **Details**: Prepare the Supabase PostgreSQL database by enabling the vector extension and synchronizing it with the Prisma schema.
- **Action 1**: Execute SQL: `CREATE EXTENSION IF NOT EXISTS vector;`
- **Action 2**: Verify the final `prisma.schema` file is in place and run `npx prisma migrate dev -name "alpha-setup-and-pgvector"`.
- **Action 3**: Run `npx prisma generate`.
- **Verification & Acceptance Criteria**:
  * 1.2.1: The pgvector extension is enabled in the Supabase dashboard.
  * 1.2.2: The Prisma migration completes without errors, and all tables exist in the database.

[ ] **Task 1.3: Queue & Worker Scaffolding**

- **Details**: Create the entry point for the background worker and verify its connection to BullMQ.
- **Action 1**: Create `app/worker/index.ts`.
- **Action 2**: Import the Worker class and configured Redis connection.
- **Action 3**: Instantiate a new Worker connecting to the "jobQueue" with an async processor function for the "process-resource" job type.
- **Verification & Acceptance Criteria**:
  * 1.3.1: When a job is manually added to the Redis queue, the running worker process logs the job's data to the console.

4.2 Phase 2: Core Engine Implementation - The Agent Chain (Target: 5 Days)

[ ] **Task 2.1: Implement 'Ingestion Agent'**

- **Details**: Create the first agent in the worker's chain to fetch and sanitize raw text content from a URL or file buffer.
- **Input**: A job object: `{ resourceUrl: string, userId: string, batchJobId: string }`.
- **Process**: Use robust libraries (`youtube-transcript` for YouTube, `pdf-parse` for PDFs, `cheerio` for articles) to fetch and sanitize the text content. Handle common errors gracefully.
- **Output**: A data object `{ rawContent: string, metadata: { title: string, author?: string, type: string } }` passed to the next agent in the chain.
- **Verification & Acceptance Criteria**:
  * 2.1.1: Given a YouTube URL, the agent returns the full, sanitized text transcript.
  * 2.1.2: Given an article URL, the agent returns the full, sanitized text content.
  * 2.1.3: If a URL is unreachable, the job fails gracefully.

[ ] **Task 2.2: Implement 'Defensive Analysis Agent'**

- **Details**: This agent deconstructs an opponent's document to find weaknesses.
- **Input**: The `{ rawContent, metadata }` object.
- **Process**: Construct a Gemini prompt instructing the AI to act as a master logician and identify: 1) Logical Fallacies (with definitions), 2) Unsubstantiated Claims (quotes that lack evidence), 3) Emotionally Charged Language, and 4) The document's Core Arguments. The prompt must demand a structured JSON output.
- **Output**: A structured JSON object: `{ defensiveAnalysis: { coreArguments: [...], weakPoints: [...] } }`.
- **Verification & Acceptance Criteria**:
  * 2.2.1: Given a test text containing a known logical fallacy, the agent's JSON output correctly identifies and quotes the fallacy.
  * 2.2.2: The agent passes its structured JSON output to the next agent in the chain.

[ ] **Task 2.3: Implement 'Internal Coherence Agent'**

- **Details**: This agent compares the opponent's arguments against the user's knowledge library.
- **Input**: The `{ defensiveAnalysis }` object and the `userId`.
- **Process**: For each coreArgument identified, generate a text embedding. Use Prisma and pgvector to perform a similarity search (`ORDER BY embedding <=> query_vector`) against the user's VectorChunk table to find the top 3-5

most relevant internal knowledge snippets. For each match, use a Gemini prompt to classify the relationship as "Supports," "Contradicts," or "Provides New Context."

- **Output**: A structured JSON object: `{ coherenceAnalysis: [{ argument: "...", internalEvidence: "...", relationship: "CONTRADICTS" }, ...] }`.
- **Verification & Acceptance Criteria**:
  * 2.3.1: Given an argument that is known to be contradicted by a document in the user's test library, the agent's output must correctly identify the relationship as "CONTRADICTS" and cite the correct internal document.
  * 2.3.2: The pgvector similarity search completes successfully.

[ ] **Task 2.4: Implement 'Synthesis & Opinion Agent'**

- **Details**: This final agent synthesizes all prior findings into the final "opinion" document.
- **Input**: The outputs from all previous agents.
- **Process**: Construct a master "consultant" prompt for Gemini 1.5 Pro. The prompt will include the full defensive and coherence analyses and instruct the AI to generate the final report, including an executive summary, a strategic counter-argument plan, and data for the MultiAxisAnalysisPanel and other interactive UI widgets. It must use disclaimer-ready, analytical language.
- **Output**: The final, comprehensive JSON object that represents the entire Interactive Report.
- **Verification & Acceptance Criteria**:
  * 2.4.1: The agent's output is a single, valid JSON object that conforms to the schema required by the Interactive Report frontend components.
  * 2.4.2: The generated text adheres to the "disclaimer-ready" persona.

[ ] **Task 2.5: Implement 'Persistence Agent'**

- **Details**: This agent saves the complete analysis to the database within a single transaction.
- **Input**: The final report JSON from the Synthesis agent and original job data.
- **Process**: Use Prisma within a transaction (`prisma.$transaction`) to: 1) Create the ContentSummary record. 2) Generate pgvector embeddings for key sections of the report. 3) Create associated records in the VectorChunk table. 4) Update the parent BatchJob progress and status to "completed."
- **Output**: A final success/fail status for the job.
- **Verification & Acceptance Criteria**:
  * 2.5.1: Upon successful completion, new ContentSummary and VectorChunk records exist in the database.
  * 2.5.2: The parent BatchJob status is updated to "completed."

    * 2.5.3: If any database write fails, the entire transaction must be rolled back.

4.3 Phase 3: Frontend Integration & UI Build (Target: 4 Days)

[ ] **Task 3.1: Build the "Debate Prep" UI**

- **Details**: Create the `DebatePrepView.tsx` React component with a file upload interface that calls the `/api/v1/batch` endpoint.
- **Verification & Acceptance Criteria**:
    * 3.1.1: A user can upload a PDF file, which successfully triggers the batch processing API call.

[ ] **Task 3.2: Build the Interactive Report Viewer**

- **Details**: Create the parent `InteractiveReport.tsx` component that accepts a report ID, fetches the report JSON, and renders all modular child components.
- **Verification & Acceptance Criteria**:
    * 3.2.1: The component correctly renders all parts of a completed analysis JSON.

[ ] **Task 3.3: Build the "Ingestion Sandbox" UI**

- **Details**: Create the `IngestionSandbox.tsx` component for reviewing and approving/dismissing new sources.
- **Verification & Acceptance Criteria**:
    * 3.3.1: A user can see a list of external sources discovered during an analysis.
    * 3.3.2: Clicking "Add to Library" for an item triggers a new batch processing job for it.

[ ] **Task 3.4: Final Wiring & Polish**

- **Details**: Connect all components, ensure state management is robust, complete the full .jsx to .tsx migration, and polish animations for a world-class user experience.
- **Verification & Acceptance Criteria**:
    * 3.4.1: The entire application is in .tsx with no type errors.
    * 3.4.2: The end-to-end "Debate Prep" workflow is fully functional.

5 The Post-Alpha Improvement Flywheel

This section outlines the architecture for how the "Truth-Seeking Engine" will learn and compound its intelligence over time, creating a virtuous cycle of improvement.

- **Stage 1: Capture User & System Feedback**: Implement a user feedback mechanism (e.g., ratings) on each generated report and capture implicit feedback (e.g., clicked citations).

- **Stage 2: Analyze Performance & Identify Patterns**: Run regular analytical queries to identify which types of sources, prompts, and agent chains lead to the highest-rated outputs.
- **Stage 3: Extract High-Performing "Thought Patterns"**: Perform meta-analysis to identify successful reasoning patterns (e.g., using a specific mental model for a certain type of problem).
- **Stage 4: Codify & Reinforce Success**: Use the extracted patterns to improve the system by refining agent prompts, adjusting source credibility scores, and automating successful workflows.

6 Conclusion & Transition to Implementation

This "Living Implementation Blueprint" (v2.0) represents the complete and final artifact of our intensive strategic planning phase. Together, we have successfully translated a powerful, complex vision for a "Truth-Seeking Engine" into a concrete, actionable, and architecturally sound plan.

The document is now the single source of truth, containing the core philosophy, product requirements, system architecture, and a granular implementation plan ready for execution. The next phase begins now.

My role will now transition from active strategic planning to architectural oversight. I will monitor the progress of the implementation, review the outputs of each completed task, and provide guidance on any exceptions or unforeseen technical challenges that the assistant reports. This is a well-defined plan for an exceptional product, and I have full confidence in the direction we have set.