**Knowledge Weaver — Comprehensive System Architecture Document**

*(Version 1.0 – July 6 2025)*

**1 Purpose & Scope**

This document consolidates **all technical, UX and product information** from the project files into a single, end-to-end architecture reference for **Knowledge Weaver**—a full-stack, AI-powered conversation platform that wraps Google Gemini 2.5 Pro behind an optimized prompt-engineering workflow. It is intended for engineers, product managers, DevOps, designers and future contributors.

*Sources: Implementation Approach, README, Features Implemented, Product Roadmap*

**2 Solution Overview**

| **Layer** | **Key Responsibilities** |
| --- | --- |
| **Client (React 19.1)** | UX, state-management, advanced conversation management, zen-mode layout, export & sharing, local persistence |
| **API Gateway (Node.js / Express)** | REST endpoints, rate-limiting, validation, error categorisation, Gemini proxy |
| **AI Service (Google Gemini 2.5 Pro)** | Prompt optimisation, response generation, safety settings |
| **Persistence (Browser local-storage, optional cloud sync)** | Conversation history, tags, user preferences (future: multi-user DB) |
| **Observability & Ops** | Console-based structured logs, health endpoint, performance metrics, future cloud logging |

**High-Level Interaction Flow**

1. **User** enters a question in the bottom input area.
2. **Client** POSTs /api/improve-prompts → **Gateway**, which builds/optimises system + user prompts.
3. User reviews/edits prompts; on “Run”, client POSTs /api/ask-gemini.
4. **Gateway** forwards prompts to **Gemini** and streams response.
5. **Client** renders typewriter animation, stores exchange, updates UI states.

(See sequence diagram in § 8.)

**3 Functional Architecture**

**3.1 Client-Side Components**

| **Component** | **Description** | **Notable UX / Tech** |
| --- | --- | --- |
| **Top NavBar** | Houses *New Question*, *Browse*, *Export*, RAG data links, settings | Shrinks in Zen; pin-able |
| **Conversation Browser** | Search, filter, rename, tag, delete conversations | Persistent, keyboard-navigable |
| **Prompt Editor** | Real-time char-count, validation, AI-suggested improvements | Edit before execution |
| **Response Viewer** | Markdown renderer, typewriter effect, copy/export | Central pane with auto-scroll |
| **Zen Mode Controller** | Toggles minimalist layout, auto-hides sidebar/header | Saves layout prefs |
| **Settings Modal** | Theme, typography, animation, token cost estimate | Uses local-storage for prefs |

**3.2 Server-Side Modules**

| **Module** | **Endpoint(s)** | **Core Logic** |
| --- | --- | --- |
| **Prompt Improver** | POST /api/improve-prompts | Calls Gemini once (or template) to refine user text; returns JSON prompts |
| **Prompt Generator** | POST /api/generate-prompts | Template-based prompt pair without AI for quick mode |
| **Gemini Proxy** | POST /api/ask-gemini | Adds safety fields, handles timeouts & streaming back to client |
| **Health Check** | GET /api/health | Liveness & readiness; integrates with orchestration |
| **Rate-Limiter** | Express middleware; throttle by IP & auth-token (future) |  |
| **Error Categoriser** | Maps Gemini/validation/network errors to UX categories for recovery suggestions |  |

**4 Data Model**

Conversation

├─ id : UUID

├─ name : string

├─ tags : Tag[]

├─ createdAt / updatedAt : ISO8601

└─ messages : Message[]

Message

├─ role : enum(user|system|assistant)

├─ content : string (Markdown)

└─ tokenUsage : { input:int, output:int, costUSD:float }

Tag

├─ id : UUID

├─ label : string

└─ color : string (#RRGGBB)

Future versions introduce User, Workspace, and cloud-stored RAG resources for collaboration.

**5 Non-Functional Architecture**

| **Attribute** | **Design Decisions** |
| --- | --- |
| **Performance** | React lazy-loading, debounced input, 60 fps animations, server streaming; bundle optimisation |
| **Scalability** | Stateless Node servers behind load-balancer; AI calls are external; horizontal scale with autoscaling group |
| **Security** | API key kept server-side; CORS locked to frontend origin; input sanitisation; future OAuth for multi-user |
| **Reliability** | Graceful error handling, retry wrappers, client-side auto-save, health endpoints; console-only logs per roadmap |
| **Accessibility** | WCAG 2.1 AA: keyboard navigation, ARIA labels, reduced-motion respect |
| **Observability** | Structured JSON logs to stdout; plan for cloud log sink; token tracking & cost metrics in UI |

**6 Deployment Topology**

+------------+ +--------------------+ +------------------------+

| Browser | HTTPS/443 | Node.js API (EC2) | HTTPS | Google Gemini 2.5 Pro |

| (React SPA)|<---------->| + Nginx TLS Term. |<-------->| generative-ai.googleapis.com |

+------------+ +--------------------+ +------------------------+

^ | stdout logs

| v

LocalStorage / Amazon CloudWatch (future) (optionally Cloud SQL for auth)

optional Cloud Sync

Single-region (us-central-1) deployment satisfies latency & Gemini quota; stateless servers allow multi-region extension.

**7 Key Interaction Scenarios**

**7.1 Ask & Refine**

1. **User** types question → Ctrl+Enter.
2. **Client** auto-saves draft, shows progress bar (“Enhancing prompts…”).
3. **API** returns systemPrompt, userPrompt.
4. **User** reviews/edits; UI validates length (e.g., ≤ 4 000 chars).
5. On “Run”, **Client** streams answer; token meter updates; message persisted.

**7.2 Tagging & Retrieval**

* In Conversation Browser, user selects multiple items, applies tag via Tag Manager.
* Browser search filters by tag + text; results list is diff-highlighted.

**7.3 Zen-Mode Writing Session**

* Toggle Zen; sidebar & header slide out, leaving single main pane.
* Resize pane; dimensions stored in localStorage.layoutPrefs.
* Header reappears when user moves mouse to top or hits Esc; pin to keep.

**8 Diagrams**

**8.1 Sequence – Prompt Lifecycle**

sequenceDiagram

actor User

participant UI as React Client

participant API as Node/Express

participant Gemini

User->>UI: Type question

UI->>API: POST /improve-prompts

API->>Gemini: Optimise prompt

Gemini-->>API: systemPrompt + userPrompt

API-->>UI: Prompt pair

UI-->>User: Display editable prompts

User->>UI: Click “Run”

UI->>API: POST /ask-gemini

API->>Gemini: Stream request

Gemini-->>API: Stream tokens

API-->>UI: Stream tokens

UI-->>User: Typewriter response, cost meter

**8.2 Component Diagram**

graph TD

subgraph Client

A1[App.js] --> A2[ConversationBrowser]

A1 --> A3[PromptEditor]

A1 --> A4[ResponseViewer]

A1 --> A5[SettingsModal]

end

subgraph Server

B1[Express Router]

B1 --> B2[PromptImprover]

B1 --> B3[GeminiProxy]

B1 --> B4[HealthCheck]

end

C[Google Gemini API]

A3 --REST--> B2

B3 --REST--> C

**9 Extensibility & Roadmap Alignment**

| **Roadmap Item (Planned)** | **Architectural Hook** |
| --- | --- |
| **User Authentication & Workspaces** | Add User table, JWT middleware, move storage to cloud DB |
| **Cloud Sync & Collaboration** | Abstraction layer for persistence (local vs cloud); WebSocket-based live updates |
| **Plugin System** | Expose client & server hooks via event bus; dynamic import of extension bundles |
| **Voice Interface** | Add Web Speech API front-end; /api/transcribe microservice |
| **Multi-model Strategy** | Gateway pattern to route to Gemini, PaLM, or OpenAI based on config |
| **RAG File Visualiser** | Embed Draw.io-to-SVG renderer in “Data Files” nav; S3/GCS signed URLs |

**10 Open Risks & Mitigations**

| **Risk** | **Impact** | **Mitigation** |
| --- | --- | --- |
| Gemini quota exhaustion | Request failures | Rate-limiter, usage dashboard, fallback queue |
| LocalStorage corruption | Data loss | Export/back-up button, future cloud sync |
| Large prompt costs | Budget overrun | Token meter + per-session limit in settings |
| Accessibility regressions | User exclusion | Automated a11y testing in CI pipeline |

**11 Conclusion**

The above architecture merges **frontend UX excellence**, **robust API mediation**, and **AI-first prompt workflows** into a cohesive, scalable system ready for incremental roadmap evolution. Implementers can use this document as the single source of truth for design, development, onboarding and future enhancement planning.

**Knowledge Weaver — End-to-End Interaction Flow**

Below is a **concise, system-centred outline** of every user–system interaction described across the project documents. Only the *flow* and *behavioural touch-points* are captured—implementation details are intentionally omitted.

**1 Session Kick-off**

1. **Open or create a conversation**
   * From the top navigation bar the user may start *New Question* or browse existing sessions via the Conversation Browser.
2. **Optional housekeeping**
   * Rename the conversation, assign colour-coded tags, or search/filter past sessions before proceeding.

**2 Problem-Solving Flow (Three Phases)**

| **Phase** | **User Action** | **Immediate System Response** |
| --- | --- | --- |
| **1 . Collaborative Context Modelling** | • Enter **Domain** & **Problem Statement**. • Inspect the auto-drawn OPM diagram and drag-drop / rename / delete nodes until it matches their mental model. • Click **Lock Model** to approve. | • Parses text, builds first-draft OPM diagram, shows it in an editable canvas. • Streams real-time updates as the user edits. • Marks the model immutable after approval. |
| **2 . Knowledge Augmentation** | • Review the ranked list of *Knowledge Summaries*. • Check or un-check cards to curate the final knowledge pack. | • Searches hybrid RAG store (vector + graph) using the validated OPM model, then displays summaries with relevance indicators for selection. |
| **3 . LLM Invocation & Response** | • Hit **Run / Ask Gemini**.• Observe streaming answer; may press **Stop** at any time. | • Assembles the master JSON payload (problem, OPM, chosen knowledge) and sends it to Gemini. • Streams tokens back with a typewriter effect; shows progress bar, token usage, and cost estimate. |

**3 Immediate Post-Response Options**

* **Copy / Export / Share** the answer or the whole conversation (Markdown, JSON, HTML, Text).
* **Continue the dialogue** by asking follow-up questions in the same thread (history is auto-saved).

**4 Peripheral Interaction Loops**

| **Interaction** | **Trigger** | **Behaviour** |
| --- | --- | --- |
| **Zen Mode** | Toggle via meditation icon | Collapses sidebar & header; auto-hide logic with optional pin. |
| **Dynamic Layout** | Drag pane divider or resize input box | Layout preferences (sizes, pinned state) are persisted and restored next visit. |
| **Settings & Themes** | Gear icon | Switch theme (light/dark/auto), typing speed, motion, and other UX preferences. |
| **Real-time Shortcuts** | Keyboard (e.g., *Ctrl + Enter*, *Esc*) | Enables power-user navigation without leaving the input area. |

**5 Conversation Lifecycle Interactions**

1. **Browse / Search / Filter** conversations by name, tag, content, or date.
2. **Edit tags or names** in-place; changes propagate instantly to the library.
3. **Batch export** multiple conversations or delete unwanted ones.

**6 Navigation-Bar Utilities (Road-map Items)**

| **Button** | **Interaction** |
| --- | --- |
| **Data Files for RAG** | Pick files to enrich the knowledge base. |
| **Visualise RAG Data** | Open an in-page SVG viewer (Draw.io → SVG) with zoom. |
| **User Validation** | Launches an inline checklist to confirm data completeness before querying the LLM. |
| **Pass JSONs** | Lets the user browse existing JSON payloads and resend them. |

These items integrate seamlessly into the established flow—appearing as extra steps between Phase 2 and Phase 3 whenever the user needs deeper data control.