

RAG System Design: Comprehensive Specification

1. High-Level Architecture

The system is a modular **Retrieval-Augmented Generation (RAG)** ecosystem. It separates document "understanding" from real-time "answering."

- **Frontend:** Vercel (Next.js) handles the UI and streaming chat.
 - **Backend:** FastAPI on Railway manages orchestration and the ingestion queue.
 - **Database:** Pinecone serves as the high-dimensional vector store.
-

2. The Ingestion Pipeline (The "Docling" Flow)

This layer transforms raw files into "Knowledge Atoms."

2.1 Multi-Format Normalization

Supports `.pdf`, `.docx`, `.txt`, `.json`, and `.csv`. All files are normalized into a unified Markdown-style structure before chunking.

2.2 Hybrid Chunking & Contextualization

- **Chunking:** Uses the `HybridChunker` to split text into ~512 token segments.
 - **Hierarchy:** Preserves document structure (e.g., `Resume > Experience > Tech Stack`).
 - **Contextualization:** Every chunk is enriched with a `.contextualize()` summary, providing the LLM with a "macro-view" of the data.
-

3. Namespace Routing & Storage

To handle mixed-topic documents, the system employs **Per-Chunk LLM Routing**.

- **Logic:** Each chunk's `context_summary` is analyzed by an LLM (gpt-4o-mini).
 - **Namespaces:** `projects`, `experience`, `personal`, and `education`.
 - **Retrieval:** If a user asks a broad question, the system pulls from the **Summary Layer** (Level 1). For specific queries, it hits the **Detail Layer** (Level 2).
-

4. Technical Stack Summary

Component	Technology
Parsing	Docling (IBM)
Embeddings	OpenAI text-embedding-3-small
Vector DB	Pinecone (Serverless)