

AI Empower RAG Service: SRS & TDD - V5.0

Version: 5.0 (Storage Decoupling Release) **Date:** November 25, 2025 **Purpose:** To implement the archival plumbing required for **Library Scale (V6)** by decoupling vector storage from Firestore, while maintaining the real-time search performance and cost efficiency of the current **Textbook Scale (V4)** system.

1. Executive Summary: V5 Architectural Shift

Version 5.0 is an **incremental update** to the V4 Production system. It introduces a secondary, archival indexing path to eliminate the future risk of high Firestore vector storage costs.

- **Core Goal:** Achieve **Storage Decoupling** by writing all Parent/Child data to GCS in Parquet format, *in addition* to writing the Child vectors to Firestore.
- **Search Engine:** **Firestore Native Vector Search remains the primary, real-time query engine.**
- **Cost Impact:** Minimal, as no new dedicated compute is introduced.
- **V6 Readiness:** High. The entire data layer is now ready for a "flip" to Vertex AI Vector Search when required.

2. Functional Requirements (FR)

ID	Requirement	Description	V5 Implementation Detail
FR-01	Asynchronous Ingestion	Dispatcher queues jobs to Pub/Sub.	No Change. Retains the V4 Fan-Out model.
FR-02	Parent-Child Indexing	Document chunks are hierarchically linked (Parent: 2000 chars, Child: 400 chars).	No Change. Retains V4 logic.

FR-03	Conversational Memory	System retains the last 4 user/assistant turns via Firestore.	No Change. Retains V4 logic.
FR-04	Storage Decoupling	The ingestion Worker must output vector data to a cost-efficient storage format for future scaling.	NEW: Worker service must install pyarrow and pandas . After writing to Firestore, it must export the Parent/Child data block to a .parquet file in GCS.

3. Technical Design Document (TDD) Updates

3.1 Worker Service Redesign (**src/ingestion-worker/**)

The core Python Worker must be updated to handle the dual-write process (Firestore + Parquet).

src/ingestion-worker/requirements.txt (Update)

```
google-cloud-storage
google-cloud-firestore
google-cloud-pubsub
google-cloud-aiplatform
langchain==0.1.0
langchain-community==0.0.10
langchain-google-vertexai
pypdf
flask
unicorn
pandas          # NEW: Required for DataFrame/Parquet
pyarrow         # NEW: Required for Parquet file format
```

Worker Logic Flow (New Step)

The logic must add a final data export step:

1. Read Pub/Sub message.
2. Parse PDF page into Parent/Child chunks.
3. **Accumulate Data:** Collect all Parent/Child data and vectors into a temporary list/DataFrame.

4. Dual Write:
- Write Parent/Child to **Firestore** (for V5 search).

○ Write accumulated data to **GCS as Parquet** (for V6 archive).

3.2 Data Schema Updates

A new secondary schema is introduced for archival.

Collection / Store	Purpose	Fields
Firestore (Primary)	Real-Time Search & Memory (Same as V4)	rag_parents, rag_children, rag_chat_history
GCS Parquet (Secondary)	Archive / V6 Source	client_id, parent_id, embedding (full list), content, source, page

4. V6 Roadmap: Full SmartRAG Implementation

Version 6.0 shifts the system from a Textbook Scale to a **Library Scale** (100K+ documents) production solution by replacing the high-cost, instant-indexing service (Firestore Vector Search) with a dedicated, cost-efficient Vector Search Layer.

Feature Area	V5 (Current State)	V6 (Target Library Scale)	New Technology Required
Vector Storage	Firestore + GCS Parquet Archive	GCS Parquet Archive (Firestore vectors disabled)	N/A (Plumbing is ready)
Retrieval Engine	Firestore's built-in KNN	Managed Vector Search Layer	Vertex AI Vector Search (Required for performance)

Indexing Process	Real-time, Transactional Write	Batch Index Builder Service (Cloud Run Job)	Dedicated Cloud Run Job service for reading Parquet and pushing to Vertex AI Index.
Multimodality	PDF/PPTX (Text only)	Video, Audio, Image Support	Vertex AI Video/Audio Processing API (to transcribe/extract frames).

5. Non-Functional Requirements (NFR)

- **Maintainability (IaC):** The introduction of new dependencies (`pandas`, `pyarrow`) must be added to the Terraform configuration to remain compliant with the Infrastructure as Code requirement.
- **Cost Control:** The dual-write approach maintains the low operational cost baseline of V4 while simultaneously mitigating the future risk of escalating Firestore vector indexing fees.
- **Performance:** All ingestion steps remain asynchronous, preserving the target ingestion speed of **500 pages in <3 minutes**.