Technical Design Document

Project: AI Empower – Enterprise RAG Service

Version: 4.0 **Date:** November 24, 2025 **Author:** Greg Burns

I will generate a clean, final version of the V4 TDD, incorporating the memory specification and the critical Vertex AI enablement requirement learned during the deployment process.

1. Executive Summary

The objective of Version 4.0 is to transition the AI Empower RAG Service from a synchronous, function-based prototype (V3) to an **asynchronous, containerized enterprise platform**. This release addresses the critical limitation of V3: the inability to ingest large textbooks (500+ pages) due to timeout constraints.

Key Architectural Shifts:

- **Compute:** Migration from Cloud Functions to **Cloud Run** to support higher concurrency and containerization.
- **Ingestion:** Implementation of a **Fan-Out Architecture** using Pub/Sub to decouple file parsing from vectorization.
- **Indexing:** Adoption of **Parent-Child Indexing** to maximize retrieval precision (Child) while maintaining Large Language Model (LLM) context (Parent).
- **Provisioning:** Full migration to **Terraform** (IaC) for reproducible infrastructure.

2. System Architecture

2.1 High-Level Data Flow (Async Pipeline)

V4 uses an event-driven **"Dispatcher-Worker" pattern**:

1. **Upload:** User uploads PDF/PPTX to GCS (gs://bucket/uploads/{client_id}/...).
2. **Trigger:** Eventarc detects google.cloud.storage.object.v1.finalized.
3. **Dispatch:** The **Ingestion Dispatcher** (Cloud Run) splits the file into logical page groups and publishes tasks to Pub/Sub (ingestion-tasks).
4. **Process: Ingestion Workers** (Cloud Run, Autoscaling) process pages in parallel, generate embeddings, and write to Firestore.
5. **Retrieval:** The **Retrieval API** (Cloud Run) queries Child vectors, fetches Parent context, and sends the payload to Gemini 2.5 Pro.

Component	V3 (Current)	V4 (Target)	Rationale
Compute	Cloud Functions (Gen 2)	**Cloud Run**	Better concurrency control & Docker support.
Orchestration	Direct Trigger	**Google Pub/Sub**	Asynchronous buffering for massive files.
Deployment	Jupyter Notebook	**Terraform + Docker**	Reproducibility & DevOps best practices.
Indexing	Simple Chunking	**Parent-Child**	Decouples search precision from context window.

3. Detailed Component Design

3.2 Ingestion Worker Service

- **Idempotency:** Document IDs are **deterministic hashes** (hash(filename + page_num + index)) to ensure re-running a job overwrites data rather than duplicating it.
- **Chunking:** Text is split into **Parent Chunks** (approx. 2000 chars) for context and **Child Chunks** (approx. 400 chars) for vectorization.

3.3 Retrieval API (Updated)

- **Endpoint:** POST /query
- **Logic Update:**
    1. Convert user query to vector.
    2. Vector Search: Query rag_children (Child Chunks) for top matches.
    3. Resolution: Extract parent_id from the matched Children.
    4. Fetch: Retrieve unique Parent documents from Firestore.
    5. **Generate:** Pass Parent Context + **Last 4 Turns of Chat History** to Gemini 2.5 Pro. (Updated to specify memory limit)

4. Data Schema (Firestore Native)

The V4 architecture uses three primary collections for hierarchy and memory:

Collection	Field	Type	Description
rag_parents	content	String	Large context text (~2000 chars).
	client_id	String	Partition Key (Tenant Isolation).
rag_children	embedding	Vector (768-dim)	The vector used for similarity search.
	parent_id	String	Reference to the corresponding document in rag_parents.
	client_id	String	Partition Key (Tenant Isolation).
rag_chat_history	messages	Array[Map]	Stores up to 4 turns of {"role": "...", "content": "..."} objects for conversation memory.
	Index: Composite Index on client_id (ASC) + embedding (VECTOR).		

---

5. Infrastructure & Security (Terraform)

5.2 IAM & Security

- **Invoker SA (Dispatcher):** Used by Eventarc to trigger the service. Requires eventarc.eventReceiver and the GCS Service Agent requires pubsub.publisher.
- **Worker SA (Worker/Retrieval):** Used by Cloud Run containers. Needs:
    - pubsub.subscriber (to pull tasks)
    - datastore.user (to write vectors)
    - aiplatform.user (for Vertex AI) **AND the Vertex AI API must be enabled at the project level.** (Updated to reflect deployment troubleshooting)
    - storage.objectViewer (to read PDFs)

6. Migration Strategy

- **Data Migration:** Existing V3 data is incompatible. V3 collections are archived, and V3 documents must be **re-ingested** through the new V4 pipeline to generate Parent/Child structures.