Technical Design Document

Project: AI Empower – Enterprise RAG Service

Version: 3.0 (Current State) **Date:** November 21, 2025 **Author:** Greg Burns

1. Executive Summary

This document outlines the architecture for Version 3.0 of the Enterprise RAG Service. This release transitions the system from a single-tenant prototype to a Multi-Tenant, Multimodal Platform capable of serving distinct clients with strict data isolation.

Key Strategic Shift: V3 focuses on Logical Isolation and Reasoning Depth. We have introduced Gemini 2.5 Pro to handle complex medical queries and added PowerPoint ingestion. Architectural constraints regarding Firestore's lack of sparse indexes have pushed Hybrid Search to the V4 roadmap.

A live version of the app is available at https://test-rag-frontend-v3.streamlit.app/.

2. System Architecture

The architecture follows an **Event-Driven, Serverless Microservices** pattern on Google Cloud Platform (GCP).

2.1 High-Level Data Flow

1. **Ingestion:** `User Upload (Streamlit)` -> `GCS Bucket (Client Path)` -> `Eventarc` -> `Ingestion Function` -> `Firestore`.
2. **Retrieval:** `User Query` -> `Retrieval API` -> `Semantic Cache` -> `Vector Search (Filtered)` -> `Gemini 2.5 Pro` -> `Response`.

2.2 Component Stack

- **Compute:** Google Cloud Functions (2nd Gen).
- **Orchestration:** Eventarc (Audit Log Triggers).
- **Storage:**
    - **Blob:** Google Cloud Storage (Standard Class).
    - **Vector/NoSQL:** Firestore (Native Mode).
- **AI Models:**
    - **Embedding:** `text-embedding-004` (768 dimensions).

    - **Reasoning:** `gemini-2.5-pro` (Complex reasoning & medical context).
- **Frontend:** Streamlit (Hosted on Streamlit Cloud).

3. Technical Specifications

3.1 Ingestion Service (V3)

Name: `rag-ingestion-svc-v3`

Trigger: `google.cloud.storage.object.v1.finalized` on bucket `test-rag-backend-v3-bucket`.

Key Features:

- **Multi-Tenancy:** Automatically extracts `client_id` from the object path: `uploads/{client_id}/{filename}`.
- **Format Support:**
    - **PDF:** Parsed via `PyPDFLoader`.
    - **PPTX:** Parsed via `UnstructuredPowerPointLoader`.
- **Data Structure:**
    - Text is chunked (1000 chars, 100 overlap).
    - Embeddings are wrapped in strict `Vector()` objects.
    - Writes are batched (Max 50 docs/transaction) to ensure stability.

3.2 Retrieval Service (V3)

Name: `rag-retrieval-api-v3` **Trigger:** HTTP (Public Endpoint)

Key Features:

- **Isolation:** Enforces strict data access using Firestore `FieldFilter("client_id", "==", requester_id)`.
- **Semantic Caching:** Hashes `{client_id}:{query}` to serve sub-200ms responses for repeated questions.
- **Conversational Memory:** Persists the last 4 turns of chat history in Firestore to support context-aware follow-up questions.
- **Performance:** Configured with `min_instances=1` (Warm Pool) to eliminate cold-start latency.
- **Robustness:** Implements Strict Lazy Loading for heavy libraries to prevent container health-check failures.

3.3 Database Schema (Firestore)

Collection: `rag_knowledge_base_v3` (Vector Store) | Field | Type | Description | | :--- | :--- | :--- | | `client_id` | String | **Partition Key**. Tenant identifier (e.g., "tier1_medical"). | | `source` | String | Filename. | | `content` | String | Text chunk. | | `embedding` | Vector | 768-dim vector (Indexed). |

Collection: `rag_chat_history` (Memory) | Field | Type | Description | | :--- | :--- | :--- | | `messages` | Array[Map] | List of `{"role": "user/assistant", "content": "..."}` objects. |

Collection: `rag_semantic_cache` (Performance) | Field | Type | Description | | :--- | :--- | :--- | | `response` | Map | Cached JSON response. | | `timestamp` | DateTime | TTL tracking. |

4. Security & Permissions

4.1 Service Accounts

- **Compute SA:** Runs the functions. Has `storage.admin` (for metadata validation), `datastore.user`, and `aiplatform.user`.
- **Storage Agent:** Has `pubsub.publisher` to emit events to Eventarc.
- **Frontend SA:** A dedicated `rag-frontend-uploader` account with `storage.objectCreator` allows the Streamlit app to securely upload user files without making the bucket public.

4.2 Frontend Authentication

The Streamlit app authenticates with GCP using a Service Account Key stored securely in Streamlit Cloud Secrets (`st.secrets["gcp_service_account"]`).

5. Deployment Strategy

- **Infrastructure as Code:** The entire stack is provisioned via a single, linear Jupyter Notebook (`Enterprise_RAG_V3_Final.ipynb`).
- **Idempotency:** All setup scripts use `|| echo` or Python logic to prevent failure if resources already exist.
- **Recovery:** The deployment logic includes retry loops for IAM permission propagation (handling GCP's eventual consistency).

6. Architectural Decision Log (V3 Changes)

Feature	Status	Rationale
Gemini 2.5 Pro	**Adopted**	Chosen over Flash to handle complex medical reasoning (e.g., "diagnose X based on symptoms Y and Z").
PPTX Support	**Adopted**	Replaced Video support as Lecture Slides are higher priority for the client's immediate needs.
Hybrid Search	**Deferred**	Moved to V4. Firestore Native lacks BM25 (Keyword) indexing. Implementing this now would require migrating to Vertex AI Search, breaking the notebook-based deployment model.

7. Future Roadmap (V4.0 Candidates)

- **Hybrid Search (Keyword Filtering):** Implement "Fake" Hybrid Search by using Gemini to extract keywords during ingestion and filtering by `array-contains-any` during retrieval. This provides keyword precision without leaving Firestore.
- **Parent-Child Indexing:** Split documents into large "Parent" chunks for context and small "Child" chunks for precise vector matching. Retrieve the Parent when the Child is found.
- **Async Ingestion:** Decoupling parsing via Cloud Pub/Sub for massive parallel processing of large textbooks.
- **Map-Reduce Summarization:** Handling distinct summarization requests for entire chapters/books.