Software Requirements Specification (SRS)

Project: AI Empower – Enterprise RAG Service

Version: 4.0 (High-Scale & Precision Release) **Date:** November 21, 2025 **Author:** Greg Burns (Coding Advisor) & Engineering Team

1. Introduction

1.1 Purpose

The purpose of Version 4.0 is to transition the AI Empower RAG Service from a synchronous, notebook-deployed proof-of-concept into a scalable, enterprise-grade platform. This release specifically addresses limitations regarding large document ingestion (timeouts on 500+ page books) and retrieval precision, while adhering to the client's constraint of using Firestore Native as the primary database.

1.2 Scope

The scope of V4 includes:

- **Architectural Overhaul:** Migrating infrastructure provisioning from scripts to Terraform.
- **Asynchronous Ingestion:** Decoupling file parsing from embedding using Google Cloud Pub/Sub to support massive files.
- **Advanced Indexing:** Implementing "Parent-Child" chunking to decouple vector search resolution (small chunks) from LLM context window (large chunks).
- **DevOps:** Establishing a local development environment and CI/CD pipelines.

1.3 Definitions, Acronyms, and Abbreviations

- **Parent Chunk:** A large text segment (e.g., 2000 chars) providing semantic context.
- **Child Chunk:** A smaller segment (e.g., 200-400 chars) derived from a Parent, optimized for vector similarity.
- **Pub/Sub:** Google Cloud Pub/Sub, an asynchronous messaging service.
- **IaC:** Infrastructure as Code (Terraform).

2. Overall Description

2.1 Product Perspective

V4 serves as the backend for AI Empower's medical education platform. It must integrate seamlessly with the existing V3 frontend (Streamlit) while replacing the backend processing logic to handle "Textbook-Scale" workloads.

2.2 User Classes and Characteristics

- **Tier 1 Clients (Institutions):** Upload entire libraries of medical textbooks. Expect zero timeouts and high availability.
- **Students:** Ask complex diagnostic questions. Expect precise answers derived from specific paragraphs, not vague summaries.
- **DevOps Engineers:** Need to deploy the system to new environments (Staging/Prod) reliability without manual notebook execution.

2.3 Operating Environment

- **Cloud Provider:** Google Cloud Platform (GCP).
- **Compute:** Cloud Run (replacing Cloud Functions for better concurrency control).
- **Database:** Firestore Native Mode (Vector Search).
- **Orchestration:** Terraform Cloud or Local State.

3. Functional Requirements

3.1 Asynchronous Ingestion Pipeline

FR-01: Job Dispatching

- **Description:** The system shall trigger a "Dispatcher" service immediately upon file upload.
- **Input:** PDF/PPTX file event from GCS.
- **Action:** The Dispatcher shall split the document into individual pages or logical sections.
- **Output:** The Dispatcher shall publish one message per page to a Pub/Sub topic (e.g., `ingestion-tasks`).
- **Constraint:** The Dispatcher must complete execution within 60 seconds.

FR-02: Parallel Processing

- **Description:** A "Worker" service shall subscribe to the `ingestion-tasks` Pub/Sub topic.
- **Scale:** The Worker service shall auto-scale (via Cloud Run) to process up to 100 pages concurrently.
- **Action:** Parse text, generate embeddings, and write to Firestore.

- **Resilience:** Failed pages must be retried automatically (Dead Letter Queue configuration).

3.2 Parent-Child Indexing Strategy

FR-03: Hierarchical Chunking

- **Description:** During ingestion, text shall be segmented into "Parents" (2000-4000 characters) and "Children" (200-500 characters).
- **Storage:**
  - Parent Chunks are stored as plain text documents in Firestore.
  - Child Chunks are stored with a `parent_id` reference and the Vector Embedding.
  - *Note:* Only Child Chunks are vectorized.

FR-04: Context Retrieval

- **Description:** The Retrieval API shall perform vector search against Child Chunks to find the "Best Matches."
- **Action:** Upon finding a Child Chunk, the system shall fetch its corresponding Parent Chunk.
- **Rationale:** This ensures the LLM receives full context (surrounding paragraphs) rather than a fragmented sentence.

3.3 Retrieval Enhancements

FR-05: Agentic Query Decomposition (Optional Candidate)

- **Description:** For multi-part questions (e.g., "Compare X and Y"), the system may decompose the query into sub-queries using Gemini before executing vector search.

4. Non-Functional Requirements

4.1 Performance & Scalability

- **Ingestion Speed:** A 500-page PDF must be fully indexed and searchable within 3 minutes (vs. >9 minutes/timeout in V3).
- **Latency:** Retrieval P95 latency shall remain under 3 seconds (excluding LLM generation time).

4.2 Maintainability (DevOps)

NFR-01: Infrastructure as Code

- All GCP resources (Buckets, databases, service accounts, IAM roles) must be defined in Terraform modules (`.tf` files).
- Manual console changes are strictly prohibited for Production environments.

NFR-02: Containerization

- All application code (Ingestion Worker, Retrieval API, Streamlit Frontend) must be packaged as Docker images.
- Deployment shall be managed via `gcloud run deploy` or GitHub Actions, not by copying code into a browser interface.

4.3 Reliability

- **Idempotency:** Processing the same file twice must not result in duplicate vector entries. The system must use deterministic IDs (e.g., hash of file content + page number) or check for existence.

5. Migration Strategy (V3 to V4)

5.1 Data Migration

- Existing data in `rag_knowledge_base_v3` (V3 format) is incompatible with Parent-Child indexing.
- **Action:** V3 data will be archived. All V3 documents must be re-ingested through the V4 pipeline to generate Parent/Child structures.

5.2 Service Cutover

- The V3 Cloud Functions will be spun down.
- The V4 Cloud Run services will be deployed behind a Load Balancer (or direct URL mapping) to take over traffic.