API Schema Definition: AI Empower RAG Service (V5)

Version: 5.0.0

Authentication: Public (Currently allows unauthenticated access for demo purposes).

Content-Type: application/json

---

1. Retrieval API

This is the primary endpoint for the chat interface. It handles vector search, context retrieval, and LLM generation.

POST /query

Submits a natural language query to the RAG system.

Request Body

Field	Type	Required	Description
query	string	**Yes**	The user's natural language question.
client_id	string	No	The tenant identifier for data isolation. Defaults to "default_client".
session_id	string	No	A unique session ID (UUID). Used to persist conversational memory across turns. Defaults to "default_session".

Example Request:

JSON

```
{
  "query": "What are the complications of a splenectomy?",
  "client_id": "medical_school_a",
  "session_id": "550e8400-e29b-41d4-a716-446655440000"
```

}

Response Body (200 OK)

Field	Type	Description
answer	string	The generative answer produced by Gemini 2.5 Pro.
context_used	string	The raw text of the retrieved "Parent" chunks used to ground the answer. Useful for citations or debugging.

Example Response:

JSON
{
  "answer": "Based on the provided context, the complications of a splenectomy include bleeding, thromboembolic events, and Overwhelming Post-Splenectomy Infection (OPSI).",
  "context_used": "\n[Source: surgery_handbook.pdf, Page: 412]\nComplications following splenectomy are rare but significant..."
}

Error Responses

- **400 Bad Request:** Missing query field in the JSON body.
- **500 Internal Server Error:** Upstream failure (Vertex AI, Firestore) or vector search timeout.

---

2. Ingestion Pipeline (Async)

The ingestion process is event-driven and does not have a synchronous REST API. Instead, it is triggered by file operations on Google Cloud Storage.

Trigger Mechanism

- **Event:** google.cloud.storage.object.v1.finalized
- **Target:** rag-dispatcher-v4 (Cloud Run Service)

Input Contract (GCS Object)

To trigger ingestion, a file must be uploaded to the GCS Bucket with the following path structure:

gs://[BUCKET_NAME]/uploads/[CLIENT_ID]/[FILENAME]

- **BUCKET_NAME**: ai-empower-rag-v4-uploads
- **CLIENT_ID**: The tenant ID (e.g., test_client). This MUST match the client_id sent to the Retrieval API.
- **FILENAME**: The file name (e.g., textbook.pdf).
- **Supported Formats:** .pdf, .pptx

Output Artifacts (Storage)

Successful ingestion produces two artifacts:

1. **Search Index (Firestore):**
    - **Collection:** rag_children (Vectors)
    - **Collection:** rag_parents (Context Text)
2. **Archive (GCS Parquet - V5 Feature):**
    - **Path:** gs://[BUCKET_NAME]/archive/[CLIENT_ID]/[FILENAME]/page_[N].parquet
    - **Format:** Apache Parquet file containing dataframe of embedding vectors and metadata.

---

3. System Health & Monitoring

Service Health Check

Cloud Run provides a default health check.

- **Endpoint:** GET / (on any service URL)
- **Response:** 200 OK (if container is running).

Log Channels

Component	Log Name	Success Indicator
Dispatcher	rag-dispatcher-v4	INFO: Successfully dispatched [N] tasks.
Worker	rag-worker-v4	INFO: V5 Archive successful for Page [N]: Wrote [X] records to GCS Parquet.
Retrieval	rag-retrieval-v4	INFO: HTTP Request: POST /query 200

---

4. Data Models (Internal Schema)

Firestore: rag_chat_history

Used for conversational memory persistence.

JSON

```
{
  "messages": [
    {
      "role": "user",
      "content": "What are the complications of a splenectomy?"
    },
    {
      "role": "assistant",
      "content": "Bleeding and OPSI are the primary risks..."
    }
  ]
}
```

Parquet Schema (V5 Archive)

The schema used for the .parquet files in GCS.

Column	Type	Description
id	String	Deterministic Hash ID.
client_id	String	Tenant ID.
embedding	List[Float]	768-dimensional vector.
content	String	Text content of the chunk.
source	String	Original filename.

page	Integer	Page number.
timestamp	Float	Unix timestamp of ingestion.