## Stage 1: Ingestion & Embedding Workflow

### 1.1 Session Upload Flow (Temporary Knowledge)

* **Upload Logic**
  + Local File Upload
  + Max 30 files per session
  + Max 5 files upload at a time
  + Files auto-expire after 7 days
* **Storage Strategy** \* **Raw files:** Local filesystem or MinIO (7-day TTL)
  + **Metadata & job status:** PostgreSQL
* **Async Processing** \* Celery workers for ingestion & parsing
* **Parsing Pipeline** \* Apache Tika (file type detection, text layer check)
  + Docling Jobkit (GPU, parallel parsing)
  + **Quality validation** \* If Docling fails / times out / low quality
    - Retry up to 3 times
    - Fallback to Unstructured or similar production-ready parser
* **Output** \* Canonical RAG JSON generated
  + Used only within session context

**Notes:** No manual chunking. No synonym customization. Fully automatic, best-practice enforced.

### 1.2 Knowledge Base Creation Flow (Persistent)

* **Logic**
  + Create Knowledge Base
  + Select domain / department / region
  + Upload bulk files (up to 5000 files)
* **Storage Strategy** \* **Raw files:** MinIO (persistent)
  + **Parsed JSON:** MinIO
  + **Metadata, routing info, status:** PostgreSQL
* **Async Processing** \* Celery workers with progress tracking
* **Parsing Pipeline** \* Apache Tika
  + Docling Jobkit (GPU)
  + **Quality validation** \* Retry up to 3 times
    - Fallback to Unstructured if required
* **Output** \* Canonical RAG JSON stored persistently

## Stage 2: Parsing & Chunking Flow

### 2.1 Session Upload Chunking (Automatic Only)

* **Logic**
  + Chunk data automatically
  + Uses **Advanced RAG** (default)
  + Section-aware chunking (Docling-based)
  + Chunk size: 200–400 tokens
  + Tables treated as standalone chunks
* **Quality Validation** \* Minimum chunk text length: 50 characters
  + Maximum empty chunks: 10%
  + Table extraction verification
  + Section hierarchy validation

**Notes:** No user control. Safe defaults enforced.

### 2.2 Knowledge Base Chunking (Configurable)

* **Logic**
  + Chunk data: Automatic (recommended) or Manual parameter tuning (optional)
  + User can select chunking mode:

| **Mode** | **Description** |
| --- | --- |
| **Naive RAG** | Fixed-size chunks (e.g., 500 tokens). No section awareness. Fast, low cost. |
| **Advanced RAG (DEFAULT)** | Section-aware chunking. 200–400 tokens. Page & heading metadata. Table-aware. |
| **Modular RAG (Expert Mode)** | Tree / section-level chunking. Parent-child relationships. Designed for complex, multi-hop queries. |

* **Manual Controls Allowed:** Chunk size, Chunk overlap.
* **Manual Controls Not Allowed:** Editing chunk text.

## Stage 3: Domain-Specific Synonym Management

### 3.1 Synonym Definition

* **Scope**
  + Per Knowledge Base (KB-level)
  + Shared across Knowledge Bases (Admin only)
* **KB-Level Synonyms (Example – HR KB)**
  + annual leave $\rightarrow$ AL, earned leave
  + probation $\rightarrow$ training period, initial period
* **Shared / Global Synonyms (Admin Managed)**
  + FY $\rightarrow$ financial year, fiscal year
  + TAT $\rightarrow$ turnaround time

### 3.2 Synonym Management Interface

* **KB Owners:** Can add/edit KB-level synonyms
* **Admins:** Can manage shared/global synonyms
* **UI Capabilities:** Simple key-value editor. No complex rule chaining. Export/import for bulk updates.

**Notes:** Synonyms stored in PostgreSQL. Versioned per KB. No document reprocessing required.

### 3.3 Synonym Usage Rules

* Synonyms applied **ONLY** at query time.
* No modification to chunks or embeddings.
* **Used in:** Query rewriting / expansion, BM25 keyword search (Elasticsearch), Query routing assistance (optional).

**Note:** Embeddings are NOT recomputed when synonyms change.

## Stage 4: Indexing Flow

* Chunked data processing using a unified chunk\_id.
* **Keyword Index** \* Elasticsearch (BM25)
  + Stores raw chunk text + metadata
  + Routing & domain filters applied
* **Vector Index** \* Primary: PostgreSQL + pgvector (HNSW)
  + Alternative: Qdrant (for dedicated vector workloads)
  + *Not recommended:* Milvus (operational complexity)

## Stage 5: Q&A / Retrieval Flow

* **User Query:** Select Session Documents OR Knowledge Base
* **Multi-turn Router:** \* Detect follow-up queries
  + Rewrite query into standalone form
  + Decompose or clarify if required
  + Apply domain / access routing
* **Final Rewritten Query:** Apply synonym expansion

### 5.1 Retrieval & Reranking

* **Keyword Retrieval:** Elasticsearch BM25 search $\rightarrow$ Fetch Top N results
* **Semantic Retrieval:** Vector DB similarity search $\rightarrow$ Fetch Top N results
* **Union:** Combine keyword + vector results
* **Reranking:** \* Cross-encoder reranker
  + Apply relevance score threshold (If below threshold $\rightarrow$ No Answer)
  + Select final top chunks

## Stage 6: Answer Generation & Monitoring

* Send final chunks to LLM.
* Generate grounded answer using **only** retrieved context.
* Attach citations (document\_id, page, section).

**Example Output:**

JSON

{  
 "answer": "...",  
 "sources": [  
 {  
 "document\_id": "...",  
 "page": 3,  
 "section": "..."  
 }  
 ]  
}