Design Document: S.M.A.R.T.

S.M.A.R.T. — Secure ‘doc’ Management And Retrieval Technology

Authors

Gurpreet K Hundal · gurpreet@sklose.com | Tiffany Valdecantos · tiv001@g.harvard.edu

Hellen Momoh · hem299@g.harvard.edu | Spiro Habasch · sph083@g.harvard.edu

1. Background and Motivation

Organizations increasingly rely on internal digital repositories—notes, policies, records—but conventional keyword-based search fails to deliver contextual understanding. SMART addresses this gap by providing an intelligent, secure, and attribution-aware retrieval system powered by LLMs and hybrid search while prioritizing data privacy and access control.

2. Scope and Objectives

SMART delivers:

- **Secure Document Storage**: Centralized, permissioned content repository (class notes, quizzes).
- **Semantic Search & Ranking**: Vector embeddings via LLM with vector and BM25 hybrid search.
- **Guardrails**: LLM powdered guardrails to avoid jailbreaking and inappropriate content.
- **LLM-Powered Summarization**: Relevant results reranked and structured via LLMs, supporting multilingual text.
- **Frontend UI**: Chatbot interface, authenticated via Google OAuth with SSL encryption.
- **Security + Audit Trail**: Logs at every access and retrieval point.

3. TechStack

No.	COMPONENT	TECHNOLOGY	PURPOSE
1	Frontend	React.js, Next.js, Tailwind CSS	Provides a responsive, modular user interface. All rendering is client-side to reduce backend exposure. No sensitive logic is handled on the frontend.
2	Authentication and encryption	Google OAuth2, SSL	Enterprise-grade identity verification. Ensures secure token-based access control with minimal attack surface. Delegates auth to trusted third party (Google), no passwords stored locally.

No.	COMPONENT	TECHNOLOGY	PURPOSE
3	API Server	FastAPI	High-performance async Python backend. Chosen for its compatibility with local models, secure routing, and full control over all I/O. Avoids opaque cloud platforms or closed-source runtimes.
4	Database	PostgreSQL + `pgvector` + `pgroonga`	Enables hybrid semantic and keyword search without relying on external vector DBs (e.g., Pinecone, ChromaDB). Local storage with full auditability and encryption support.
5	Object Storage	Google Cloud Storage (GCS)	Used only for secure document storage. Access is abstracted via signed links, preventing direct user access. Ensures scalability while maintaining fine-grained control.
6	Embedding Models	`all-MiniLM-L6-v2`, `all-mpnet-base-v2` (Hugging Face, local)	Lightweight and performant transformer models for encoding queries and documents into vector space. Hosted entirely locally, removing external API risks.
7	Reranker	`llama3:8b` via Ollama	Performs deep cross-encoder ranking. Hosted locally to ensure model weights and queries never leave the environment. Provides much better relevance over cosine alone.
8	LLM Generator	`Gemma3:12b`, `llama3:8b` via Ollama	Generates answers using rag. Chosen for open-weight licensing and strong reasoning under limited compute. Hosted securely offline.
9	Safety Filter	`llama-guard3:8b` via Ollama	Applies content safety and compliance filtering before LLM responses are returned. Local deployment ensures no user data leaves the system for moderation.
10	Language Detection	`langdetect`, `langid`, `polyglot`	Implements majority-vote detection with fallback to robust heuristics. No calls to Google Translate or any online classifier, fully offline detection logic.
11	Translation	`deep-translator`	Used only for multilingual fallback to English if needed. Translation happens locally unless explicitly extended; fails silently for secure environments.
12	CI/CD	GitHub Actions, Ansible, Helm, Google Kubernetes Engine	Used for continuous deployment. Runs: - Linters - Test unit & integration - Checks cluster (Ansible) - Deploys (Helm Chart) in GKE

4. SMART Schema

No.	TABLE	DESCRIPTION
1	class	Stores metadata about each class (e.g., course title, authors, term)
2	access	Manages access control for each user and class, linking user email to class access rights
3	document	Represents individual documents associated with a specific class
4	chunk	Contains individual document chunks + vector embeddings for retrieval
5	audit	Logs user queries, their embeddings, retrieved document IDs, chunks, responses, and timestamps
6	user_tokens	Manages OAuth tokens for user authentication and session renewal
7	chat_history	Records chat sessions per user, including conversation history, model used, and session timestamps

Indexing and Extensions:

- **Extensions:**
  - vectors: Supports vector search for embedding similarity queries.
  - pgroonga: Enables full-text search capabilities on text columns.

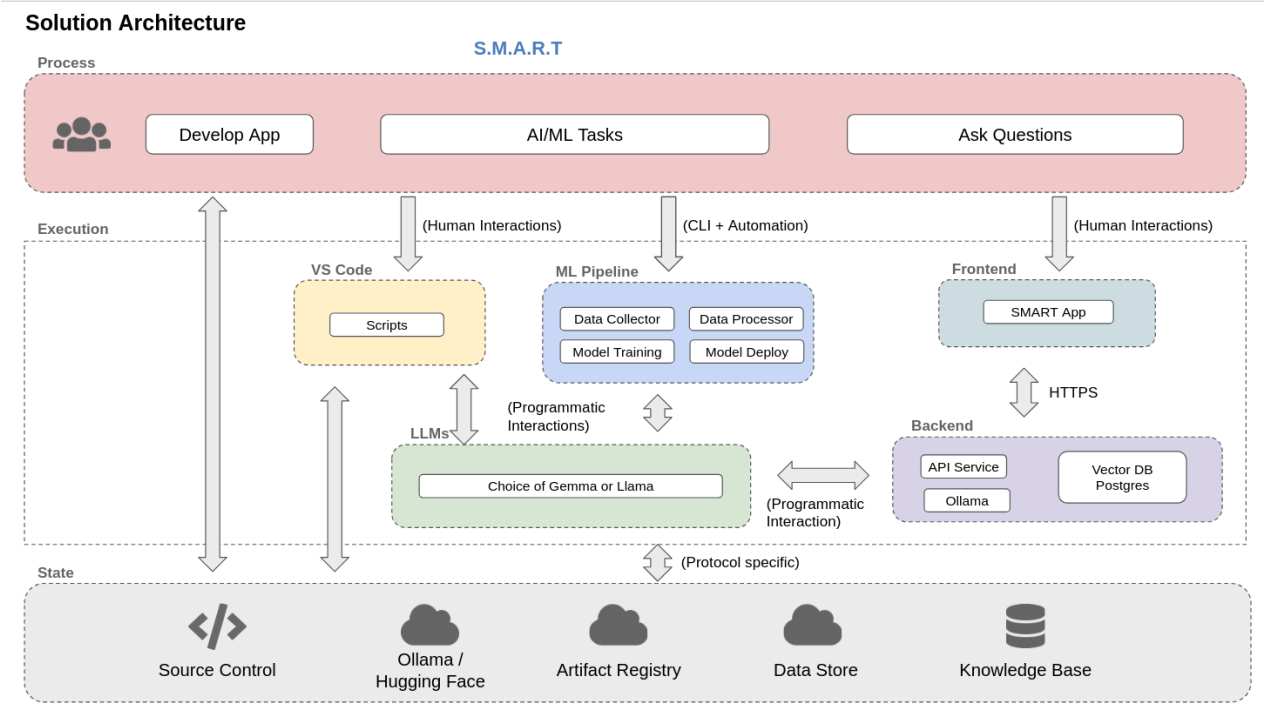
- **Indexes:**
  - pgroonga_chunk_text_index: Full-text search on chunk_text in the chunk table.
  - idx_chat_history_chat_id: Optimizes queries by chat_id.
  - idx_chat_history_session_id: Optimizes queries by session_id.
  - idx_chat_history_model: Optimizes queries by model.
  - idx_chat_history_dts: Optimizes queries by dts (descending).

5. SMART Artifacts

Below are the artifacts of SMART deployed via docker compose.

Artifacts	SMART
postgres	- port 5432 - Persistent, starts the database postgres
datapipeline	- Creates semantic chunking and stores the metadata and chunks in the postgres database - Dependent on postgres for storing the chunks and metadata
ollama	- Starts the Ollama server and pulls models: llamaguard3:8b", "llama3:8b" "gemma3:12b - Stores the downloaded models in persistent directory
api	- port: 9000 - Dependent on ollama and postgres - Creates the RAG pipeline, handling query processing, ranking, guardrails, language detection, and embedding generation - Hosts APIs and utility endpoints for downstream services - Configured with CORS to connect to the frontend service running on port 3000
frontend	- port: 3000 - Hosts the frontend application, including all necessary scripts and assets - Integrated with Google OAuth for user authentication and session management - Dependent on api service - Configured with SSL for secure communication

6. Solution Architecture



7. Technical Architecture