Ghulam Ishaq Khan Institute of Engineering Sciences and Technology (GIKI)

Software Requirements and Design Specification (SR&D Specification)

for

CURAMIND
Advanced Medical QA Engine with Dynamic RAG Integration & Real-Time Clinical Knowledge Grounding

by

Member 1: Hamza Anjum 2020149

Member 2: Muhammad Ahmad 2022335

Member 3: Shaheer 2022424

Member 4: Anna Shah 2022099

Supervisor

Engr. Ahsan Shah

Table of Contents

Software Requirements and Design Specification (SR&D Specification)

CURAMIND
Advanced Medical QA Engine with Dynamic RAG Integration & Real-Time Clinical Knowledge Grounding

Revision History

Name	Date	Reason for Changes	Version

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to formally define the complete functional, non-functional, and systemic requirements of **CURAMIND**, an advanced RAG-powered biomedical question-answering system emphasizing real-time retrieval, clinical reasoning constraints, and transparent evidence-grounded generation.
CURAMIND is intended as a **research and educational tool**, not a diagnostic system, and supports biomedical inquiry by leveraging structured retrieval pipelines, SNOMED-CT–driven patient context modeling, and safety-aware LLM generation.

1.2 Document Conventions

- The SRS follows the **IEEE 830** standard structure.
- Functional requirements are denoted as **REQ-X**.
- **Bold** text highlights critical definitions.
- Biomedical terminology follows **SNOMED-CT, MeSH, UMLS** conventions.
- All diagrams are included in Appendix B.

1.3 Intended Audience

This document is intended for all stakeholders involved in the design, development, evaluation, and supervision of the CURAMIND system. It serves as

a comprehensive reference for understanding the system's goals, architecture, and technical requirements.

Primary Audiences:

1. **FYP Supervisor:**
   Responsible for overseeing the project's academic direction and ensuring that the scope, methodology, and system requirements align with FYP standards. The supervisor uses this document to monitor progress, verify completeness, and provide guidance during development.
2. **Evaluation Panel:**
   The panel reviews this SRS to assess the clarity, feasibility, and technical correctness of the system. They rely on the document to understand the system's purpose, capabilities, and compliance with academic, scientific, and engineering best practices during the final defense.
3. **Developers:**
   Developers refer to this document as the primary technical blueprint, using it to implement each module of the system according to the defined functional and non-functional requirements. It ensures consistent understanding of system behavior, interfaces, and constraints throughout the development lifecycle.
4. **Testers / QA Personnel:**
   Quality assurance teams use this SRS to design test plans, create test cases, and validate that the system meets all specified requirements. They rely on clearly defined functional requirements, performance expectations, and safety constraints to conduct thorough verification and validation.
5. **Medical Domain Experts:**
   Medical reviewers use the SRS to evaluate the biomedical relevance, accuracy, and practical usefulness of CURAMIND's outputs. They provide feedback on the appropriateness of clinical reasoning, evidence grounding, safety mechanisms, and alignment with biomedical standards.

1.4 Product Scope

CURAMIND is a modular biomedical information retrieval and reasoning engine designed to produce literature-grounded medical answers through:

- continuous ingestion of scientific sources (PubMed, DrugBank, CT.gov),
- hybrid semantic and lexical retrieval,
- SNOMED-CT–based demographic reasoning,

- safety filters for ensuring correctness of dosage, contradictions, and drug validity,
- transparent citation trails and evidence maps.

The system is intended strictly for **academic, research, and educational** use.

1.5 References

- **PubMed / MEDLINE**, National Library of Medicine, U.S. National Institutes of Health. Available: *pubmed.ncbi.nlm.nih.gov*.
- **ClinicalTrials.gov**, U.S. National Library of Medicine, Clinical Study Registry. Available: *clinicaltrials.gov*.
- **DrugBank**, Comprehensive Drug Reference Database. Available: *go.drugbank.com*.
- **SNOMED Clinical Terms (SNOMED-CT)**, SNOMED International. Available: *snomed.org/snomed-ct*.
- **Unified Medical Language System (UMLS) Metathesaurus**, National Library of Medicine. Available: *nlm.nih.gov/research/umls/*.
- **IBM Watson Health (Merative)**, Clinical Decision Support Technologies. Available: *merative.com/watson-health*.
- **PubMedQA Dataset**, Biomedical Question Answering Benchmark. Available: *pubmedqa.github.io*.
- **BioBERT**, Lee et al., "BioBERT: a pre-trained biomedical language representation model," GitHub Repository. Available: *github.com/dmis-lab/biobert*.
- **DeBERTa-v3**, Microsoft Research, "DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training," GitHub Repository. Available: *github.com/microsoft/DeBERTa*.
- **PyKEEN**, Ali et al., "PyKEEN: A Python Library for Training and Evaluating Knowledge Graph Embeddings," GitHub Repository. Available: *github.com/pykeen/pykeen*.
- **REDCap**, Research Electronic Data Capture Platform. Available: *project-redcap.org*.

2. Overall Description

2.1 Product Perspective

CURAMIND is designed as a standalone intelligent medical question-answering system that integrates multiple components into a single unified platform. In today's healthcare and medical-information environment, individuals often struggle to find accurate, reliable, and up-to-date information due to the vast amount of biomedical literature and complex terminology. CURAMIND addresses this need by providing a structured, transparent, and medically grounded solution that retrieves relevant information, analyzes it, and generates clear explanations supported by verified biomedical sources.

The system functions as a **complete end-to-end medical information engine**, combining OCR-based document processing, biomedical entity extraction, hybrid retrieval methods, patient-context filtering, and safety-validated LLM reasoning. Each module—such as ingestion, retrieval, reranking, reasoning, and explainability—operates as an independent unit, but all components interact seamlessly to deliver accurate and responsible medical answers to the user.

CURAMIND is built to ensure **reliable, safe, and consistent medical information presentation**. It integrates demographic filtering using SNOMED-CT, citation-backed reasoning, and strict safety checks for dosages, contraindications, and withdrawn medications. This allows the system to produce validated responses while avoiding unsafe or unsupported claims.

Overall, CURAMIND provides a unified platform that improves how users access medical knowledge by combining advanced retrieval techniques, context awareness, and safe answer generation. It enhances clarity, accessibility, and trust for anyone seeking credible medical information without requiring technical or clinical expertise.

2.1.1 High-Level System Architecture Diagram

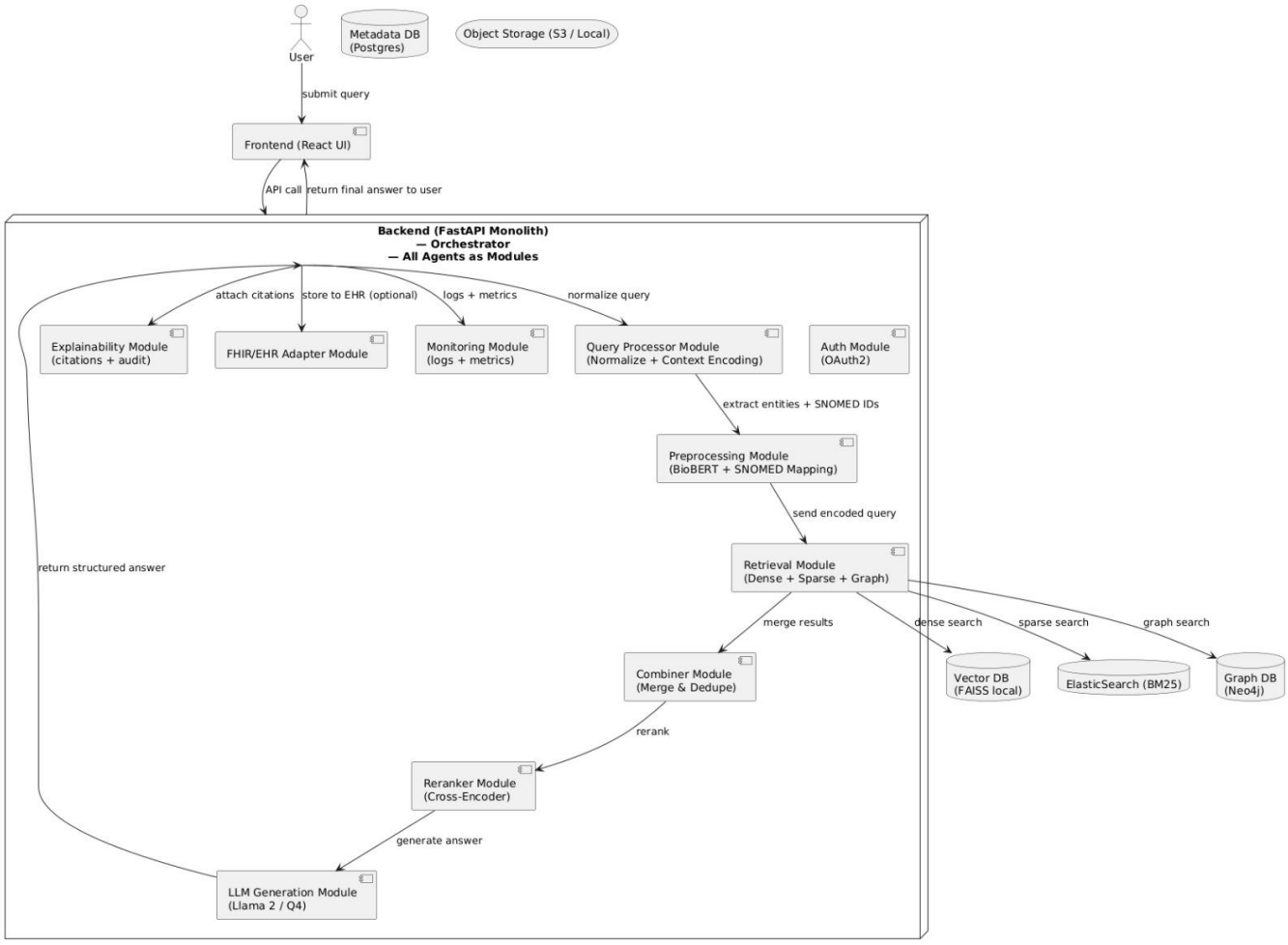

Fig 2.1 High-Level System Architecture Diagram

2.2Product Functions

- **Biomedical Document Acquisition and Processing:** CURAMIND continuously retrieves new biomedical documents using automated ingestion workflows. OCR, text parsing, and metadata extraction convert raw literature into structured, searchable knowledge.
- **Medical Entity Extraction and Metadata Tagging:** Using SciSpaCy and domain-specific models, medical entities such as diseases, genes, drugs, and clinical markers are identified. Extracted metadata supports fine-grained retrieval and patient-aware reasoning.
- **Vectorization and Semantic Indexing:** Documents undergo dense embedding generation using PubMedBERT/BioBERT, stored inside

FAISS/Weaviate for high-performance vector search. This underpins CURAMIND's semantic recall.

- **Query Processing and Hybrid Retrieval:** User queries are normalized and processed through dense, sparse, and graph-based retrieval pipelines. This ensures broad and clinically relevant evidence collection.
- **Patient-Aware Context Filtering:** Evidence is contextualized using SNOMED-CT demographic and disease associations. This personalizes biomedical information to match age, sex, conditions, and comorbidity profiles.
- **Evidence Re-ranking:** Cross-encoders refine the candidate set and prioritize clinically strongest evidence. This improves retrieval precision and reduces irrelevant noise.
- **Safety-Constrained LLM Answer Generation:** The LLM produces grounded answers that cite each supporting source. Safety mechanisms prevent hallucinations, incorrect dosage suggestions, or misleading biomedical claims.
- **Evidence Trace & Explainability:** CURAMIND attaches citations, highlighted passages, reasoning markers, and confidence scores. This ensures auditing, transparency, and academic scrutiny.
- **Audit Logging:** All interactions, retrieved evidence, and model decisions are stored for reproducibility, research assessment, and compliance-style traceability.

2.3 User Classes

- **Medical Doctors:**
  Provide expert medical evaluation of biomedical claims and verify evidence quality. They interpret the system's retrieved literature and assess its clinical consistency.
- **Researchers:**
  Use CURAMIND to analyze retrieval quality, benchmark hybrid search configurations, and evaluate LLM safety frameworks. Their role focuses on system analysis and improvement.
- **Students:**
  Interact with the system for educational learning, understanding medical literature retrieval, and exploring real-world RAG systems. They rely on transparent outputs and explanations.

2.4 Operating Environment

CURAMIND operates within a GPU-accelerated Linux environment**, primarily Ubuntu, to support computationally intensive** LLM inference **and** vector search operations**.**

Backend components:

- **Programming & Frameworks:** Python, PyTorch.
- **Model Serving:** NVIDIA Triton Inference Server for optimized LLM serving.
- **Data Ingestion:** Apache NiFi to manage ingestion pipelines and workflows.
- **Vector Indexing & Retrieval:** FAISS and Weaviate for efficient similarity search.
- **Document Processing:** Nougat OCR for extracting text from PDF-based biomedical documents.

User Interface:

- Streamlit-based frontend accessible via modern web browsers, designed for intuitive interaction with the system.

External Communication:

- REST/JSON protocols for secure interaction with biomedical APIs and optional FHIR mock EHRs.

Hardware Requirements:

- Stable GPU hardware with **≥16GB VRAM**.
- Reliable storage for embeddings, logs, and intermediate pipeline artifacts.
- Consistent internet access for external data retrieval and system updates.

This environment ensures **efficient processing, secure communications, and stable system operation** across all CURAMIND modules.

2.5 Design and Implementation Constraints

- **Data Privacy & Security:** CURAMIND shall follow HIPAA-style data handling principles. User-provided demographic or pseudo-patient information must be securely processed without long-term storage.
- **Auditability & Transparency:** All system interactions and internal processes shall support audit-style transparency, in alignment with FDA 21 CFR Part 11, ensuring actions, data flows, and reasoning traces are trackable and reproducible.
- **Approved Data Sources:** Only verified biomedical datasets and standardized vocabularies (e.g., PubMed, DrugBank, SNOMED-CT, UMLS) shall be used. Non-medical or unverified datasets are prohibited.
- **Biomedical Language Models:** System shall exclusively use biomedical LLMs (e.g., PubMedBERT, BioBERT, DeBERTa-v3) for embeddings, retrieval, and alignment to ensure accurate medical interpretation.
- **Computational Requirements:** GPU acceleration is mandatory for efficient LLM inference, dense vector embeddings, and real-time retrieval operations.
- **Interoperability:** CURAMIND shall maintain FHIR-compatible APIs for integration with EHR-like systems.
- **Modular Architecture:** Subsystems—including ingestion, retrieval, reranking, safety filtering, and answer generation—shall be modular and decoupled for independent development, testing, and upgrades.

2.6 User Documentation

- **End-user Manual:** Explains how to query the interface, interpret answers, view citations, and use patient context inputs.
- **Deployment Guide:** Provides installation instructions, environment configuration steps, GPU requirements, and setup flows.
- **API Documentation:** Details request/response formats, authentication, and endpoints for RAG, retrieval, and LLM reasoning modules.
- **Dataset Preparation Guide:** Describes steps for preparing biomedical literature, OCR processing, and ingestion pipelines.
- **Troubleshooting Notes:** Provide solutions to common issues such as retrieval failures, OCR errors, and GPU inference problems.

2.7 Assumptions and Dependencies

Assumptions

- **APIs remain accessible:** CURAMIND assumes continuous access to NIH, DrugBank, and CT.gov resources.
- **Efficient LLM operation:** Assumes quantized Llama-2/Llama-3 models provide reliable biomedical reasoning within VRAM constraints.
- **Internet stability:** Literature ingestion requires stable connectivity for updates.
- **Synthetic data viability:** Assumes the pipeline can generate synthetic biomedical QA training data when needed.

Dependencies

- **PubMed, DrugBank, ClinicalTrials APIs:** Critical for biomedical updates and ingestion.
- **Nougat OCR:** Required for extracting text from non-digital PDFs.
- **Weaviate Vector DB:** Supports semantic retrieval and indexing pipelines.
- **PyKEEN:** Required for knowledge graph embeddings and relational augmentation.
- **GPU Availability:** Necessary for high-speed inference, reranking, and retrieval.

3. External Interface Requirements

3.1 User Interfaces

CURAMIND provides a **Streamlit-based interface** designed for intuitive interaction. Users can:

- Enter medical or pseudo-patient queries.
- Inspect evidence and reasoning trails.
- Review patient metadata.
- Explore citation and source trails.

The interface emphasizes **MedPaLM-style clarity** to ensure accessibility for both medical professionals and non-technical users. Layouts, menus, and visualizations are structured for minimal cognitive load and efficient workflow navigation.

3.2 Hardware Interfaces

CURAMIND requires access to **CUDA-enabled GPUs** to support:

- LLM inference acceleration.
- Dense vector embedding generation.
- Real-time retrieval and reranking.

Local or cloud storage must be available to maintain:

- Biomedical documents and datasets.
- Precomputed embeddings.
- Logs and pipeline artifacts.
- Intermediate computational results.

3.3 Software Interfaces

The system interfaces with **external biomedical resources and software** to ensure authoritative and structured data:

- **APIs:** PubMed/MEDLINE, ClinicalTrials.gov, DrugBank.
- **Ingestion and storage tools:** Weaviate, Apache NiFi.
- **Optional integration:** FHIR servers for structured clinical communication.

These interfaces allow **real-time data ingestion**, vector indexing, and retrieval while maintaining semantic and clinical fidelity.

3.4 Communications Interfaces

All communications follow **secure, standards-compliant protocols**:

- **Transport:** HTTPS and REST-based exchanges.
- **Message format:** JSON for structured data exchange.
- **Security:** TLS encryption ensures the protection of pseudo-patient metadata, queries, and logs.

This ensures end-to-end **data confidentiality, integrity, and interoperability** across all system modules and external services.

4. System Features

4.1 Dynamic Biomedical Data Ingestion

4.1.1 Description and Priority

Enables continuous acquisition of biomedical documents with OCR, entity extraction, and structured storage.
Priority: High

4.1.2 Stimulus/Response Sequence

External API request → Fetch documents → OCR extraction → NLP parsing → Storage into retrieval index.

4.1.3 Functional Requirements

- **REQ-1:** Periodic retrieval of biomedical updates must be supported.
- **REQ-2:** OCR processing via Nougat must be performed.
- **REQ-3:** Entity extraction must utilize SciSpaCy.
- **REQ-4:** All processed chunks must be stored with timestamps.

4.2 Hybrid Retrieval Engine

4.2.1 Description and Priority

Retrieves relevant evidence using dense embeddings, sparse indexing, and biomedical knowledge graph traversal.
Priority: High

4.2.2 Stimulus/Response Sequence

User query → Text normalization → Hybrid retrieval → Cross-encoder reranking.

4.2.3 Functional Requirements

- **REQ-5:** Dense vector retrieval using PubMedBERT/BioBERT must be available.
- **REQ-6:** Lexical BM25 search must support keyword-oriented queries.
- **REQ-7:** Knowledge graph embeddings must support relational recall.
- **REQ-8:** Retrieval latency must support near-real-time usage.

4.3 Patient Context Encoding

4.3.1 Description and Priority

Incorporates demographic and clinical data using SNOMED-CT ontologies to filter and prioritize evidence.
Priority: High

4.3.2 Functional Requirements

- **REQ-9:** Patient demographic inputs (age, sex, comorbidities) must be supported.
- **REQ-10:** Evidence must be filtered based on patient profile relevance.

4.4 Safety-Constrained LLM Answer Generation

4.4.1 Description and Priority

Generates medically grounded answers with safety validation, citation enforcement, and contradiction detection.
Priority: High

4.4.2 Functional Requirements

- **REQ-11:** Dosage checks must rely on DailyMed.
- **REQ-12:** Contradiction detection must utilize BioClinicalBERT-style models.
- **REQ-13:** Sentence-level citations must accompany all outputs.
- **REQ-14:** Withdrawn drugs must be filtered out automatically.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system shall provide timely responses for biomedical retrieval.
- The system shall support scalability without performance degradation.
- The ingestion pipeline shall maintain consistent throughput during scheduled runs.

5.2 Safety Requirements

- The system shall minimize hallucinations and unsupported clinical claims.

- Dosage-related outputs shall be validated using authoritative databases.
- All LLM outputs shall pass through a safety filter.

5.3 Security Requirements

- Sensitive pseudo-patient data shall be encrypted at rest.
- Communications shall utilize industry-standard encryption in transit.
- Access control mechanisms shall ensure users only access allowed components.

5.4 Software Quality Attributes

- The UI shall be intuitive and easy to use for non-technical users.
- The system shall maintain high availability except for planned maintenance.
- Components shall be modular for ease of updates and testing.
- The system shall support portability across GPU-capable environments.

5.5 Business Rules

- Only validated biomedical data sources may be used.
- Synthetic data must be clearly labeled.
- Real-world clinical usage requires expert review.

6. Other Requirements

CURAMIND must **enforce HIPAA-style privacy practices** for all user-provided demographic or pseudo-patient inputs. Persistent logs shall be maintained to support **auditability, oversight, and reproducible research**.

All biomedical sources and datasets must be **legally licensed** and used in accordance with regulatory and ethical guidelines. Long-term retention of data, embeddings, and system states shall follow structured strategies to **ensure reproducibility and scientific traceability**.

System components—including LLMs, embedding snapshots, and ingestion pipelines—must be **systematically archived** to enable:

- Controlled experimentation.
- Repeatable scientific workflows.
- Transparent tracking of model evolution and data transformations.

7. System Requirements Summary

7.1 Functional Requirements Summary

- Biomedical ingestion, OCR, and parsing
- Entity extraction and metadata creation
- Hybrid dense/sparse/graph retrieval
- SNOMED-CT patient context integration
- Cross-encoder reranking
- Safety-constrained LLM reasoning
- Citation and explanation generation
- Audit logging and traceability

7.2 Non-Functional Requirements Summary

- Performance
- Safety
- Security
- Reliability
- Usability and Extensibility

Appendix A: Glossary

Term	Description
RAG (Retrieval-Augmented Generation)	A technique where an LLM retrieves external information and uses it to generate grounded responses.
FHIR (Fast Healthcare Interoperability Resources)	A global healthcare standard defining data formats and APIs for exchanging EHR information.
SNOMED-CT	A clinically validated medical terminology used to encode conditions, symptoms, and procedures.
PICO	A research framework representing Population, Intervention, Comparison, and Outcome.
LLM (Large Language Model)	A deep learning model trained on large-scale text (e.g., GPT, Llama-2).
OCR (Optical Character Recognition)	A technique that converts scanned or PDF text into machine-readable format.
UMNSRS	A biomedical benchmark for semantic similarity and relatedness between medical terms.
Embeddings	Numeric vector representations of text enabling semantic search.
Knowledge Graph	A graph database representing entities and relationships.
DailyMed	A database containing official FDA drug labeling.
ClinicalTrials.gov	A registry of clinical studies used for evidence extraction.
Triton Inference Server	NVIDIA's high-performance model serving platform.
ETL	Extract, Transform, Load — a data ingestion and processing workflow.
Vector Database	A database optimized for similarity search using embeddings.

Appendix B: System Diagrams

- **Use Case View (Use Case Diagram)**

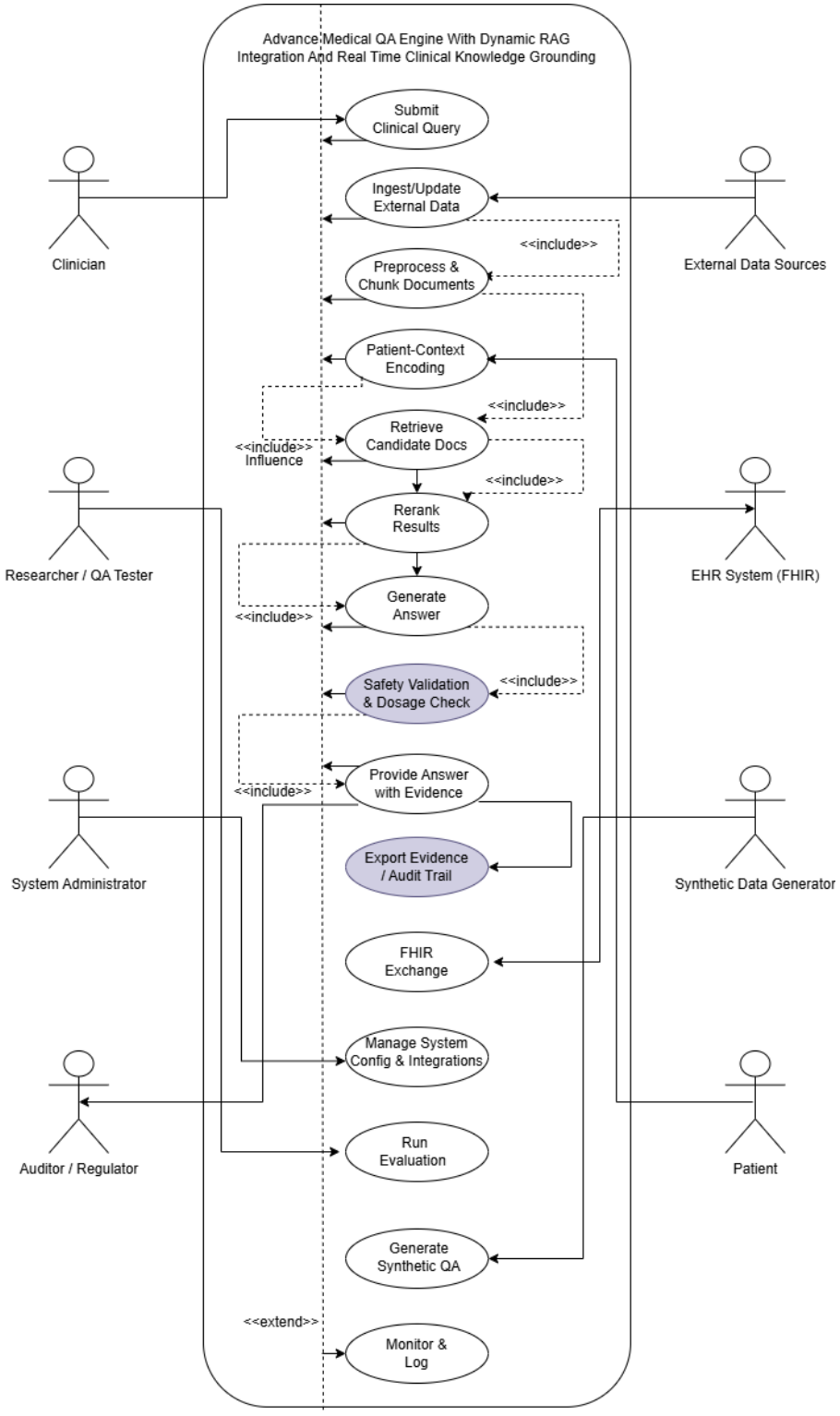

Fig 1. Use Case Diagram

- **Logical View (Class Diagram)**

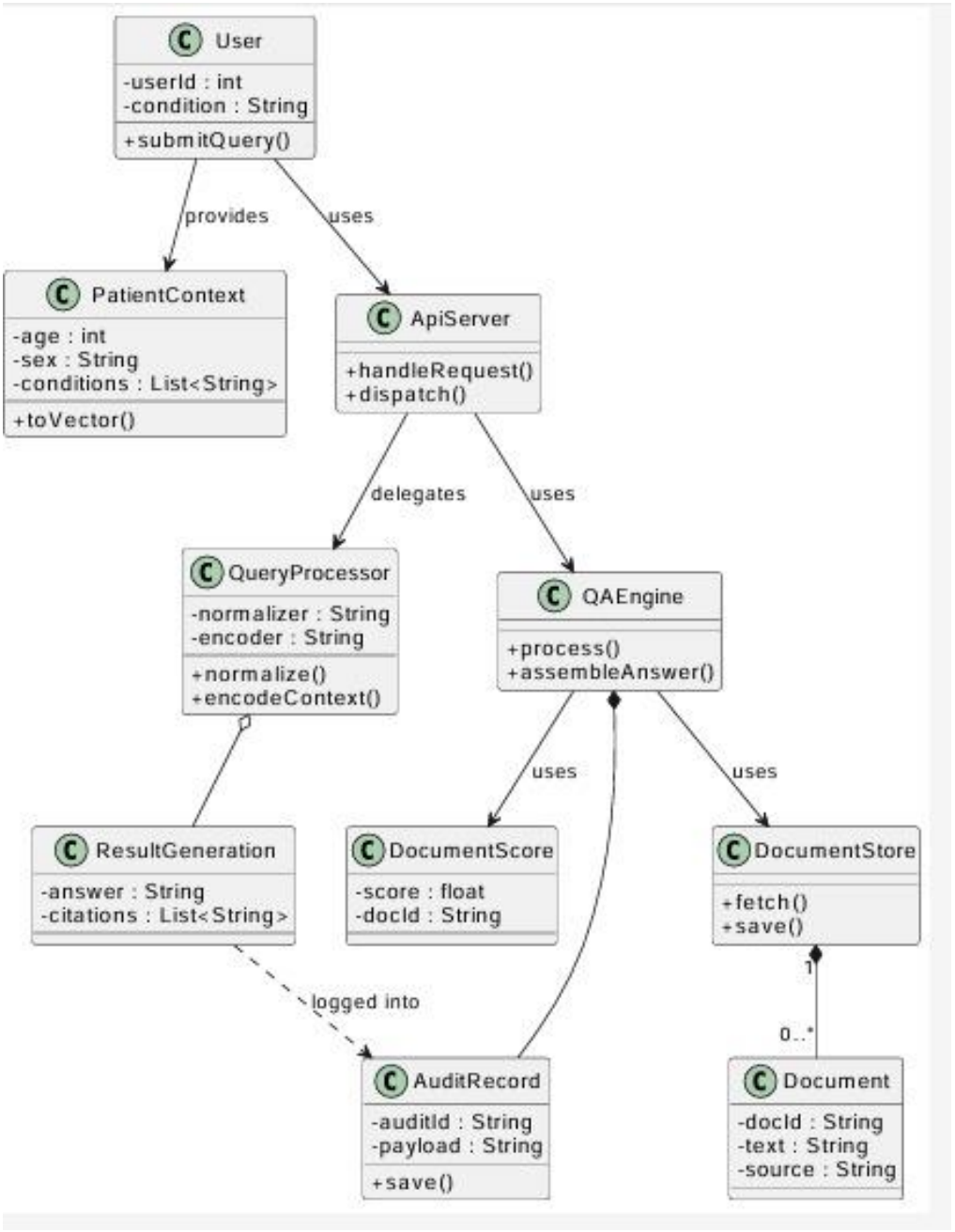

Fig 2. Class Diagram

- **Process/Behavioral View (Sequence Diagram)**

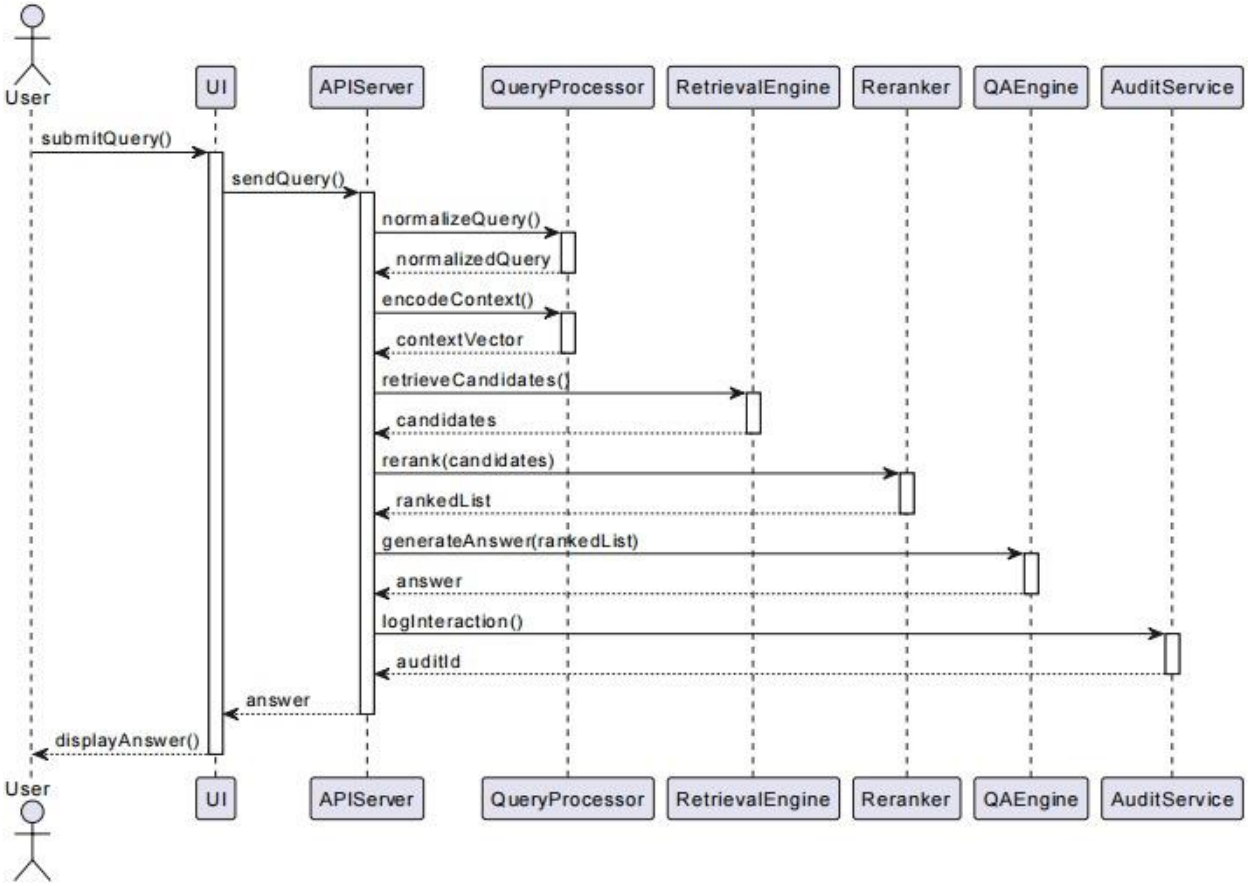

Fig 3. Sequence Diagram

- **Physical View (Physical Deployment Diagram)**

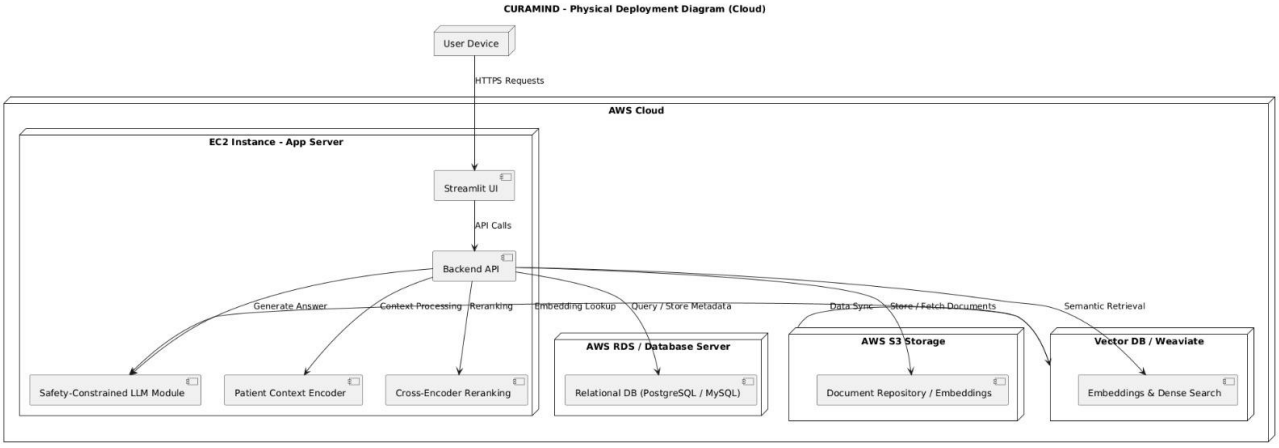

Fig 4. Physical Deployment Diagram

- **Process/Behavioral View (Activity Diagram)**

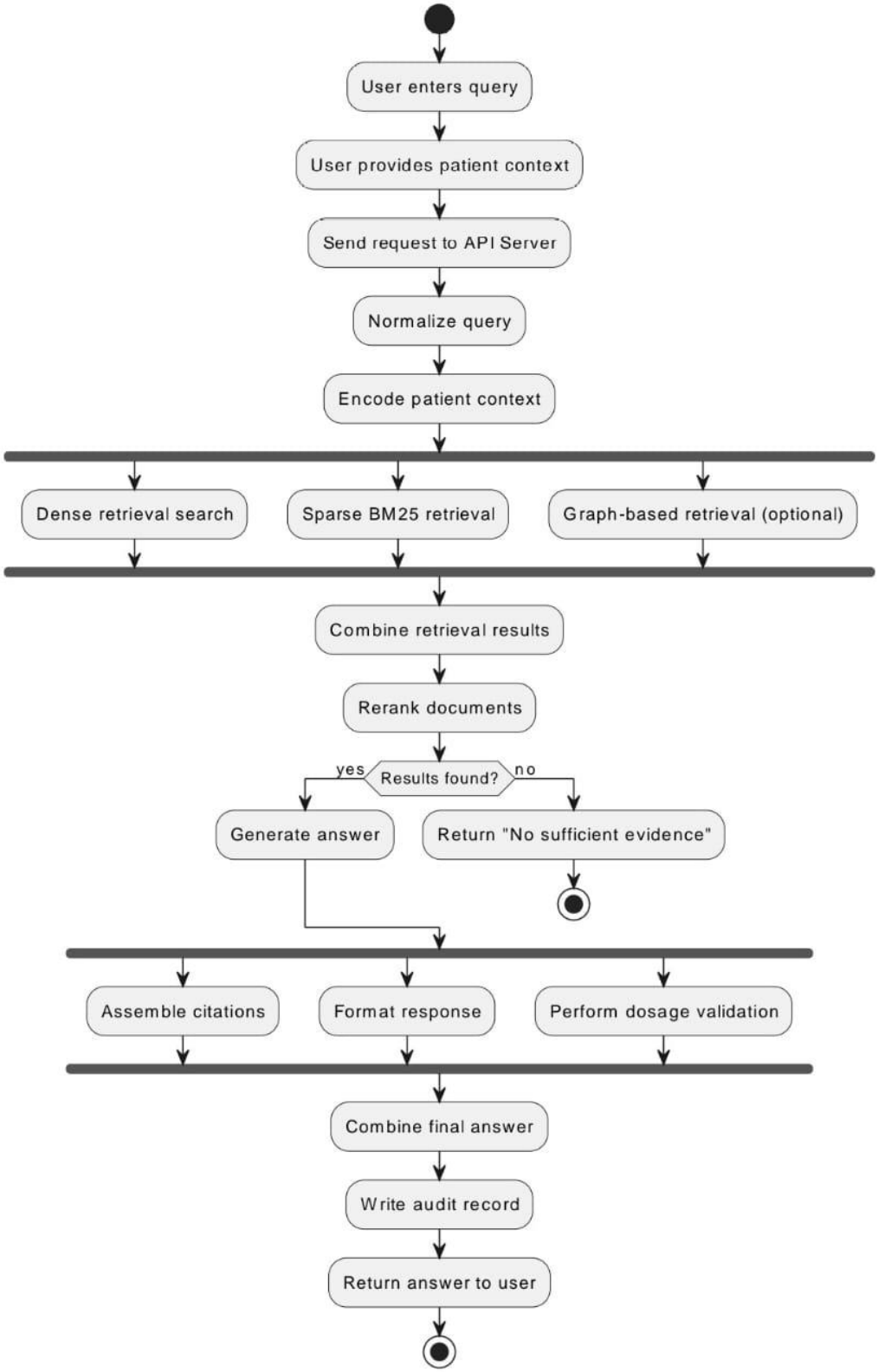

Fig 5. Activity Diagram

- **Development View (Component Diagram)**

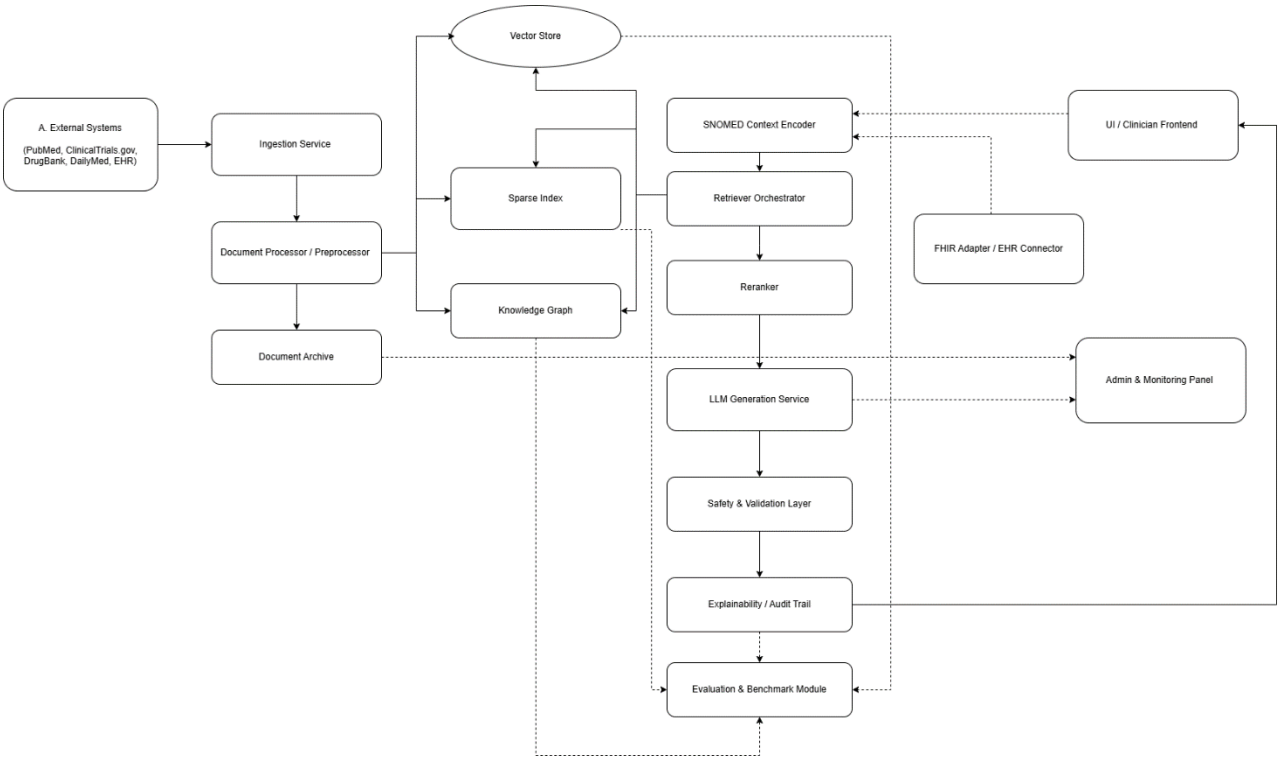

Fig 6. Component Diagram

Appendix C: To-Be-Determined List

Item	Description
Final evaluation dataset size	Precise number of cases and diversity of clinical scenarios.
Clinical validation rubric	Metrics, scoring, and evaluation methodology.
Deployment environment details	Hardware, cloud infrastructure, and system configurations.
Long-term logging retention policy	Duration, storage method, and access controls for audit and research purposes.