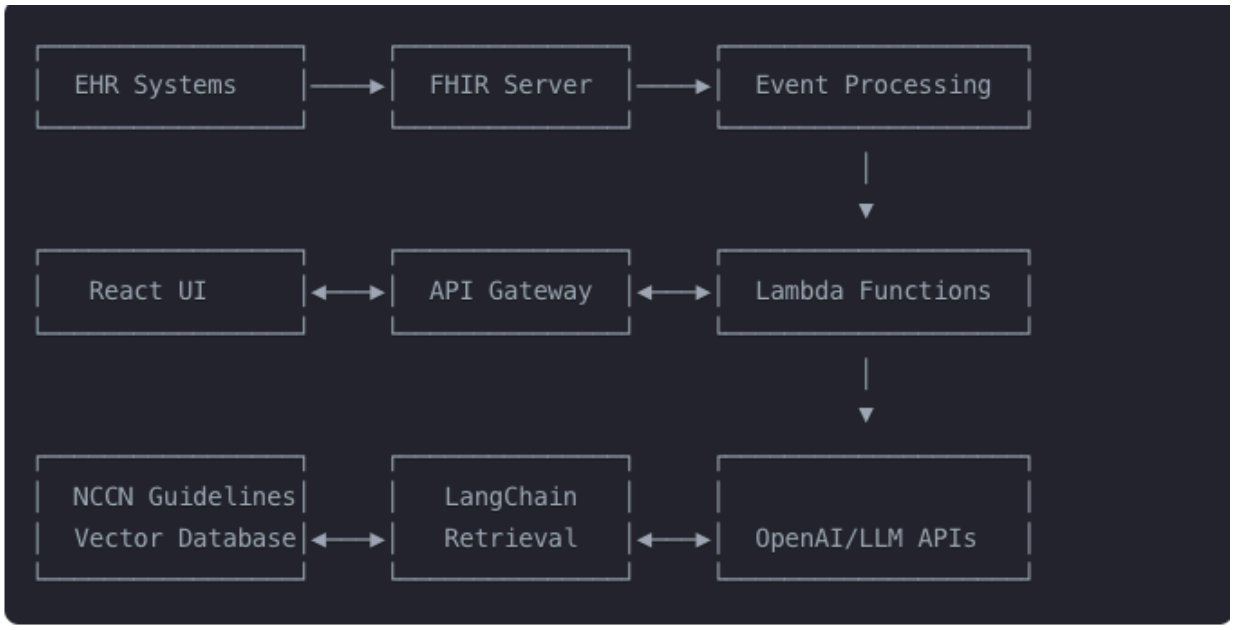
# POC for Oncology Decision Support System: FHIR-based LLM RAG Architecture

https://github.com/tmlrnc/LLM/blob/main/spiral/spiral_rag_LLM_FULL.py

## Summary

This POC integrates NCCN oncology guidelines with FHIR standards and LLM technology to create an intelligent clinical decision support system. The system leverages RAG (Retrieval Augmented Generation) with a React frontend, OpenAI LLM backend, and Pinecone vector database with advanced re-ranking techniques.

## Architecture Overview




## Key Components

## 1. Data Ingestion

- **EHR/EMR Integration**: Connect to hospital systems via FHIR API endpoints
- **NCCN Guidelines**: Process structured content from licensed guidelines
- **Medical Literature**: Index relevant oncology literature and trials

## 2. Data Processing

- **FHIR Resource Normalization**: Transform to standardized mCODE profiles
- **Clinical Data Extraction**: Process cancer type, stage, genomic markers
- **Knowledge Graph Construction**: Build patient-specific treatment pathways

## 3. LLM RAG Pipeline

- **Document Chunking & Embedding**: Segment guidelines into semantic chunks (400-500 tokens)
- **Patient Context Assembly**: Generate comprehensive patient context from FHIR resources
- **Multi-Stage Retrieval**:
    - Coarse Retrieval: Initial filtering by cancer type and stage
    - Re-ranking: Apply oncology-specialized cross-attention model
    - Citation Analysis: Extract and verify evidence sources
- **Response Generation**: Construct clinical recommendations with evidence levels

## 4. Real-time Updates

- **Event-Driven Architecture**: React to new lab results, imaging, or diagnoses
- **Alert Generation**: Identify guideline deviations and contraindications
- **UI Integration**: Deliver recommendations to clinical workflow

## Oncology RAG Re-ranker

```
def rerank(self, query, patient_context, initial_results):
    # Create cross-encoder pairs
    pairs = [(f"{query} [SEP] {patient_context}", doc.page_content)
            for doc in initial_results]

    # Calculate combined relevance score using:
    # - Semantic relevance (70%)
    # - Evidence level (A1 > A2 > B > C)
    # - Recency
    # - Genomic marker matching
    # - Trial eligibility
```

```
return reranked_results
```

# Implementation Path

1. Set up FHIR integration with test EHR system
2. Process and chunk NCCN guidelines into vector database
3. Implement RAG retrieval pipeline with specialized re-ranking
4. Build React UI with decision support views
5. Deploy serverless architecture for scalability
6. Implement real-time event processing

# Technology Stack

- **Frontend**: React, Tailwind CSS
- **Backend**: Python, FastAPI, AWS Lambda
- **Data**: FHIR R4, mCODE profiles
- **AI/ML**: OpenAI API, LangChain, Pinecone, QLoRA fine-tuning
- **Infrastructure**: Serverless, Event-driven

# Expected Outcomes

- Improved adherence to NCCN guidelines
- Reduced clinical decision time
- Enhanced identification of relevant clinical trials
- Better integration of genomic data into treatment decisions

## *Oncology Decision Support System: FHIR-based LLM RAG Architecture*

## *Data Flow: Step-by-Step Process*

1. Data Ingestion

1. **EHR/EMR Data Acquisition**
    - Connect to hospital EHR systems via FHIR API endpoints
    - Subscribe to patient data change events for real-time updates
    - Retrieve patient records, conditions, observations, and medication data
2. **NCCN Guidelines Acquisition**
    - Obtain licensed NCCN guidelines content via API or manual exports
    - Extract structured data from guidelines (cancer types, stages, recommendations)
    - Parse PDF guidelines using specialized OCR if needed for historical content
3. **Medical Literature Integration**

- Index relevant oncology literature and clinical trials data
- Maintain updated drug information and treatment protocols
- Integrate regulatory approval information for medications

2. Data Processing & Transformation
    1. **FHIR Resource Normalization**
        - Transform incoming FHIR resources to standardized mCODE profile
        - Validate resources against FHIR schemas
        - Harmonize terminologies (SNOMED, LOINC, RxNorm)
    2. **Clinical Data Extraction**
        - Extract cancer type, stage, TNM classification
        - Process genomic markers and mutation information
        - Identify treatment history and response data
        - Calculate patient parameters (age, BMI, performance status)
    3. **Knowledge Graph Construction**
        - Build patient-specific knowledge graph showing relationships between:
            - Disease characteristics
            - Treatment history
            - Observed outcomes
            - Biomarkers
        - Link to applicable NCCN pathway nodes

3. LLM RAG Processing Pipeline
    1. **Document Chunking & Embedding**
        - Split NCCN guidelines into semantic chunks (400-500 tokens each)
        - Create specialized embeddings using oncology-tuned models
        - Store in vector database with rich metadata including:
            - Cancer type
            - Stage
            - Line of therapy
            - Evidence level
            - Publication date
    2. **Patient Context Assembly**
        - Generate comprehensive patient context from FHIR resources
        - Construct prompt template with structured medical history
        - Include relevant demographics, lab values, and clinical factors
    3. **Multi-Stage Retrieval Process**
        - **Stage 1: Coarse Retrieval**
            - Filter documents by cancer type, stage, and basic criteria
            - Perform initial embedding similarity search (k=25)
        - **Stage 2: Re-ranking**
            - Apply cross-attention re-ranker specialized for oncology
            - Score documents based on clinical relevance to patient context
            - Filter by recency and evidence level

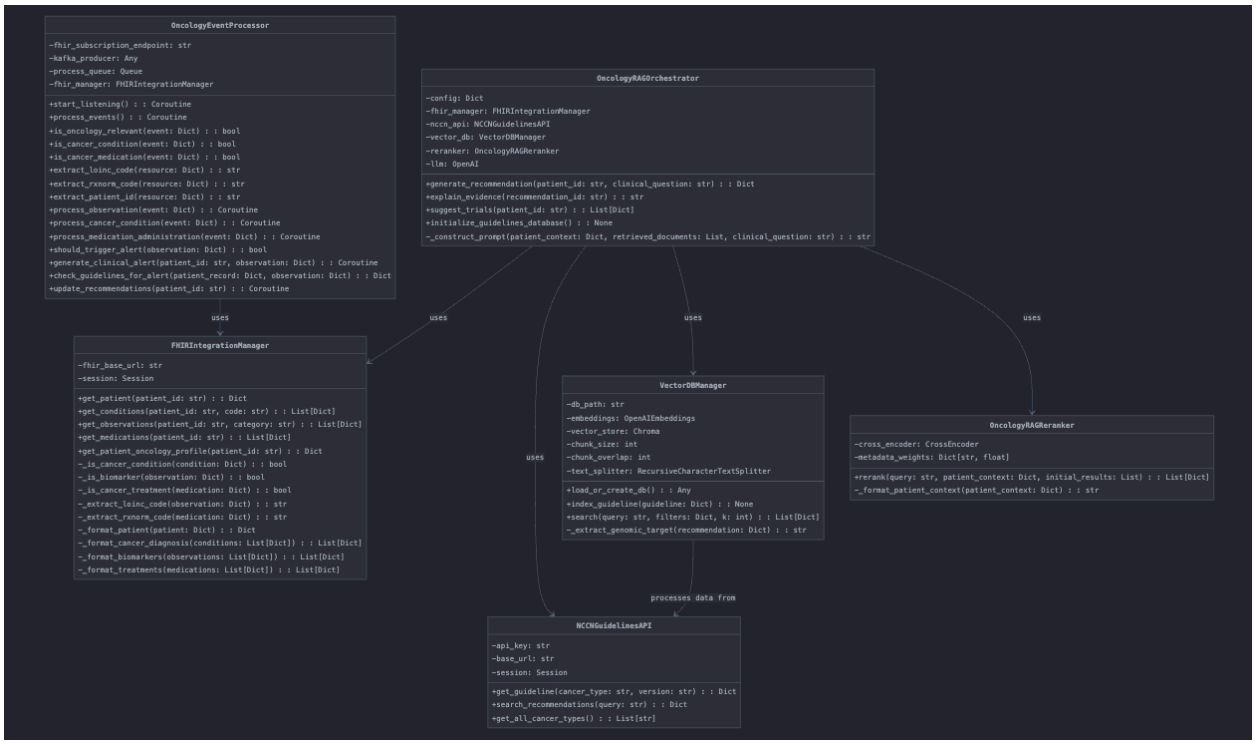
        - Select top 7-10 documents
    - **Stage 3: Citation Analysis**
        - Extract and verify citations from chosen documents
        - Link to primary evidence sources
        - Append citation metadata
4. **LLM Integration & Response Generation**
    - Construct integrated RAG prompt with:
        - Patient context
        - Retrieved guideline chunks
        - Specific clinical question
    - Generate response using domain-adapted LLM
    - Post-process to ensure clinical accuracy
    - Format response with citations and evidence levels

4. Real-time Processing & Delivery

1. **Event-Driven Updates**
    - Listen for FHIR subscription events
    - React to new lab results, imaging reports, or diagnoses
    - Trigger reassessment of recommendations when context changes
2. **Alert Generation**
    - Identify deviations from NCCN guidelines
    - Generate alerts for missing tests or procedures
    - Flag potential contraindications
3. **UI Presentation**
    - Deliver recommendations via RESTful API
    - Integrate into clinical workflow applications
    - Provide interactive decision trees with supporting evidence
4. **NCCNGuidelinesAPI**
    - Handles access to NCCN guidelines
    - Provides methods to fetch guidelines by cancer type and search recommendations
5. **FHIRIntegrationManager**
    - Manages connections to FHIR servers
    - Retrieves and formats patient data including conditions, observations, and medications
    - Contains helper methods to identify cancer-related data
6. **VectorDBManager**
    - Manages the vector database for storing guideline chunks
    - Handles indexing of guidelines and semantic search capabilities
    - Uses embedding models to create vector representations
7. **OncologyRAGReranker**
    - Specialized component for reranking search results
    - Uses cross-encoder models and oncology-specific metadata weights
    - Considers evidence levels, recency, and biomarker matches

8. **OncologyEventProcessor**
    - Processes real-time events from FHIR subscriptions
    - Identifies oncology-relevant updates and triggers alerts
    - Queues events for asynchronous processing
9. **OncologyRAGOrchestrator**
    - Central orchestrator that coordinates all components
    - Generates treatment recommendations by combining patient context with relevant guidelines
    - Provides explanation of evidence and clinical trial suggestions

The relationships between these classes show how they work together to provide oncology decision support, with the Orchestrator serving as the central coordinator that uses the other components.




# USER GUIDE RUN INSTRUCTIONS - Oncology Decision Support System -

https://github.com/tmlrnc/LLM/blob/main/spiral/spiral_rag_LLM_FULL.py

A FHIR-based LLM RAG architecture for clinical decision support in oncology.

# Overview

This system leverages FHIR healthcare data, NCCN guidelines, and large language models to provide evidence-based treatment recommendations for oncology patients. It features real-time processing of clinical events, vectorized storage of guidelines, and specialized reranking of results based on patient context.

# Architecture

The system consists of the following components:

1. **FHIR Integration Manager**: Handles all interactions with FHIR servers and EMR systems.
2. **NCCN Guidelines API**: Accesses structured guideline data from the National Comprehensive Cancer Network.
3. **Vector Database Manager**: Stores and retrieves guideline information using embeddings.
4. **Oncology RAG Reranker**: Reranks retrieval results based on patient context and oncology-specific factors.
5. **Oncology Event Processor**: Processes real-time clinical data events from FHIR subscriptions.
6. **Oncology RAG Orchestrator**: Main component that orchestrates the entire system.
7. **FastAPI Interface**: Exposes the system capabilities through a REST API.

# Requirements

- Python 3.8+
- FastAPI
- OpenAI API key
- NCCN Guidelines API access
- FHIR server access
- Vectorstore (Chroma)
- Sentence Transformers
- Kafka (for event processing)

# Installation

4. Set up configuration in `config.json` or environment variables:

```
{
    "fhir_server_url": "https://your-fhir-server/fhir",
    "nccn_api_url": "https://api.nccn.org/guidelines/v1/",
    "nccn_api_key": "your-nccn-api-key",
    "openai_api_key": "your-openai-api-key",
    "reranker_model_path": "cross-encoder/ms-marco-MiniLM-L-6-v2",
    "vector_db_path": "./vector_db",
    "fhir_subscription_endpoint":
"wss://your-fhir-server/fhir/subscription",
    "kafka_broker": "your-kafka-broker:9092"
}
```

# Usage

### Running the API Server

```
python spiral_RAG_LLM_FULL.py
```

The API will be available at http://localhost:8000.

### Getting Treatment Recommendations

```
curl -X POST "http://localhost:8000/recommendations" \
     -H "Content-Type: application/json" \
     -d '{"patient_id": "patient1234", "query": "First line treatment
for metastatic lung adenocarcinoma with EGFR mutation"}'
```

# API Endpoints

- **POST /recommendations**: Get treatment recommendations for a patient
- **GET /health**: Health check endpoint

# Implementation Details

### Vector Database

The system uses Chroma DB to store embeddings of NCCN guidelines. Each guideline is split into chunks and embedded using OpenAI's text embeddings. Metadata including cancer type, evidence level, and publication date is stored alongside the embeddings.

## Reranking Process

The reranking process combines:

1. Semantic relevance (from cross-encoder model)
2. Evidence level weighting (A1 > A2 > B > C)
3. Recency of guidelines
4. Clinical trial recommendation status
5. Genomic marker matching

## Event Processing

The system can subscribe to FHIR subscription endpoints to receive real-time updates for:

- New lab results or biomarkers
- Cancer diagnosis updates
- Treatment administrations