ZML RAG Re-ranking System Data Flow

The ZML RAG Re-ranking System implements a multi-agent architecture for enhanced information retrieval and response generation. At its core, the system employs multiple specialized agents working in concert: Query Processing Agents analyze and expand queries using domain-specific knowledge, Retrieval Agents perform parallel semantic and keyword searches, and Ranking Agents evaluate results based on relevance, diversity, and authority. The re-ranking process leverages a multi-stage pipeline where responses are scored and reordered using weighted criteria and contextual relevance. This is integrated with a vector store for efficient knowledge retrieval and a memory system for maintaining conversation context. A key innovation is the implementation of parallel processing paths where multiple agents can simultaneously evaluate different aspects of the response, with their outputs being combined through a rank fusion algorithm that considers both individual scores and inter-document relationships. The system's effectiveness is enhanced by its use of BioBERT for domain-specific embeddings and GPT-4 for natural language understanding, ultimately delivering contextually appropriate, diverse, and authoritative responses with proper citations.

https://github.com/tmlrnc/LLM/blob/main/ZML/zml_rag_4.py

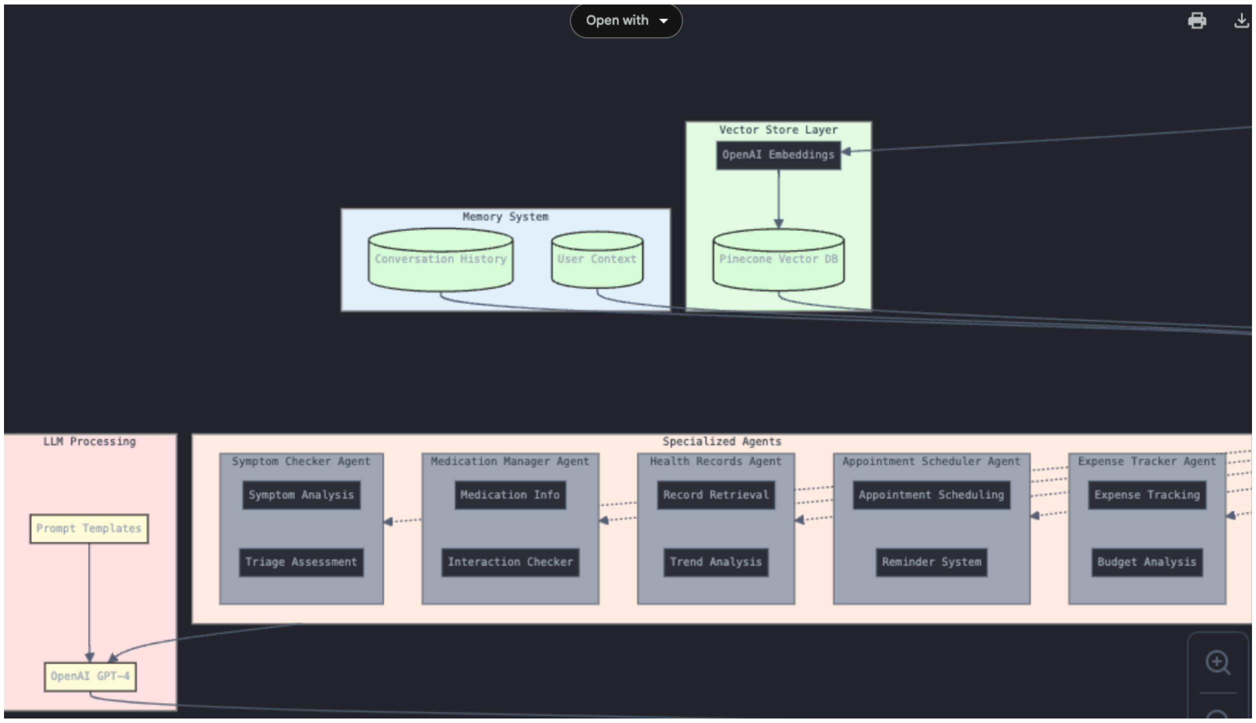
1. **Data Flow Components**:
    - User Interface Layer: Handles query input and response output
    - Core System: Main processing components
    - Vector Store Layer: Knowledge base and embeddings
    - Agent Layer: Specialized processing agents
    - Memory System: Conversation and context storage
    - LLM Layer: Language model processing
2. **Key Processes**:
    - Query Processing
    - Agent Selection
    - Context Management
    - Response Ranking
    - Memory Management
3. **Data Flows**:
    - User query → Query Processor
    - Query → Vector Store
    - Context → Agents
    - Agent outputs → Response Ranker
    - Final response → User
4. **Control Flows**:
    - Agent Selector → Specialized Agents
    - Context Manager → Agents
    - Memory System → Context Manager

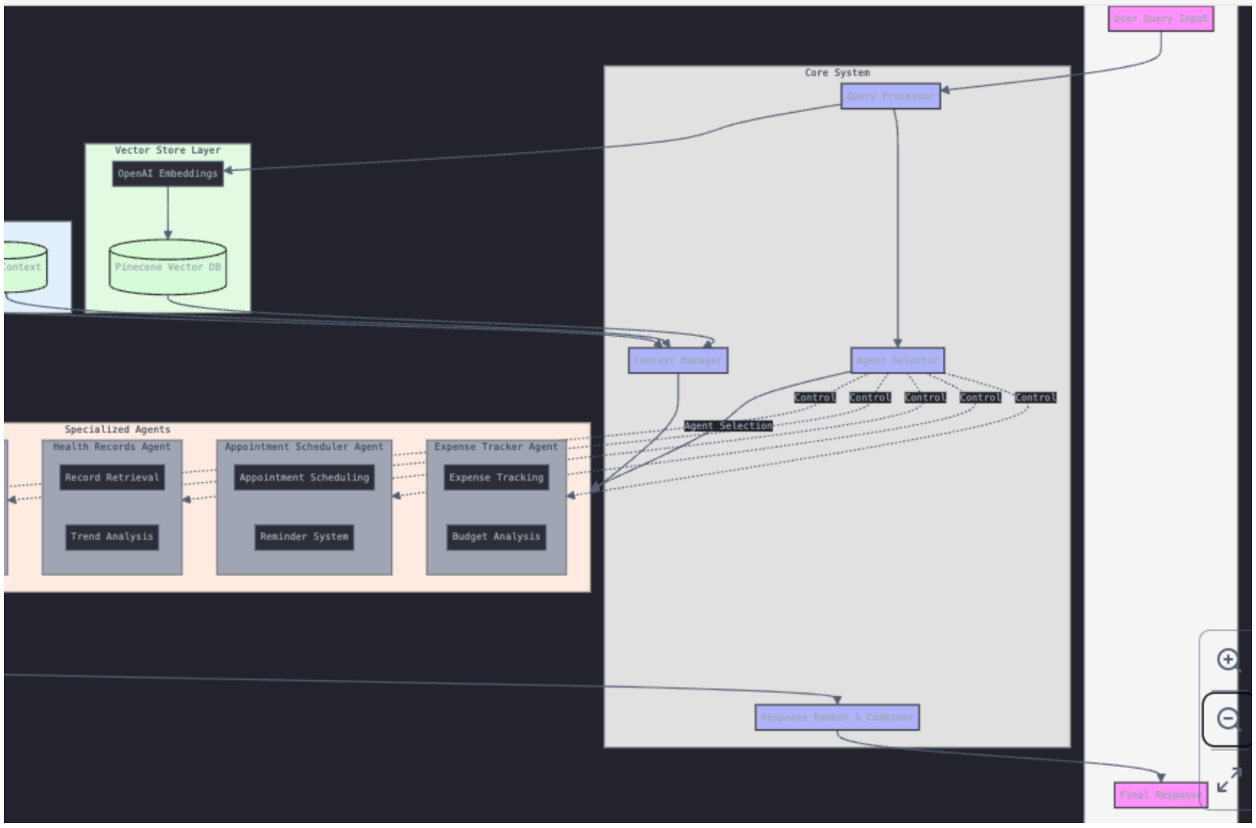
The system follows these steps:

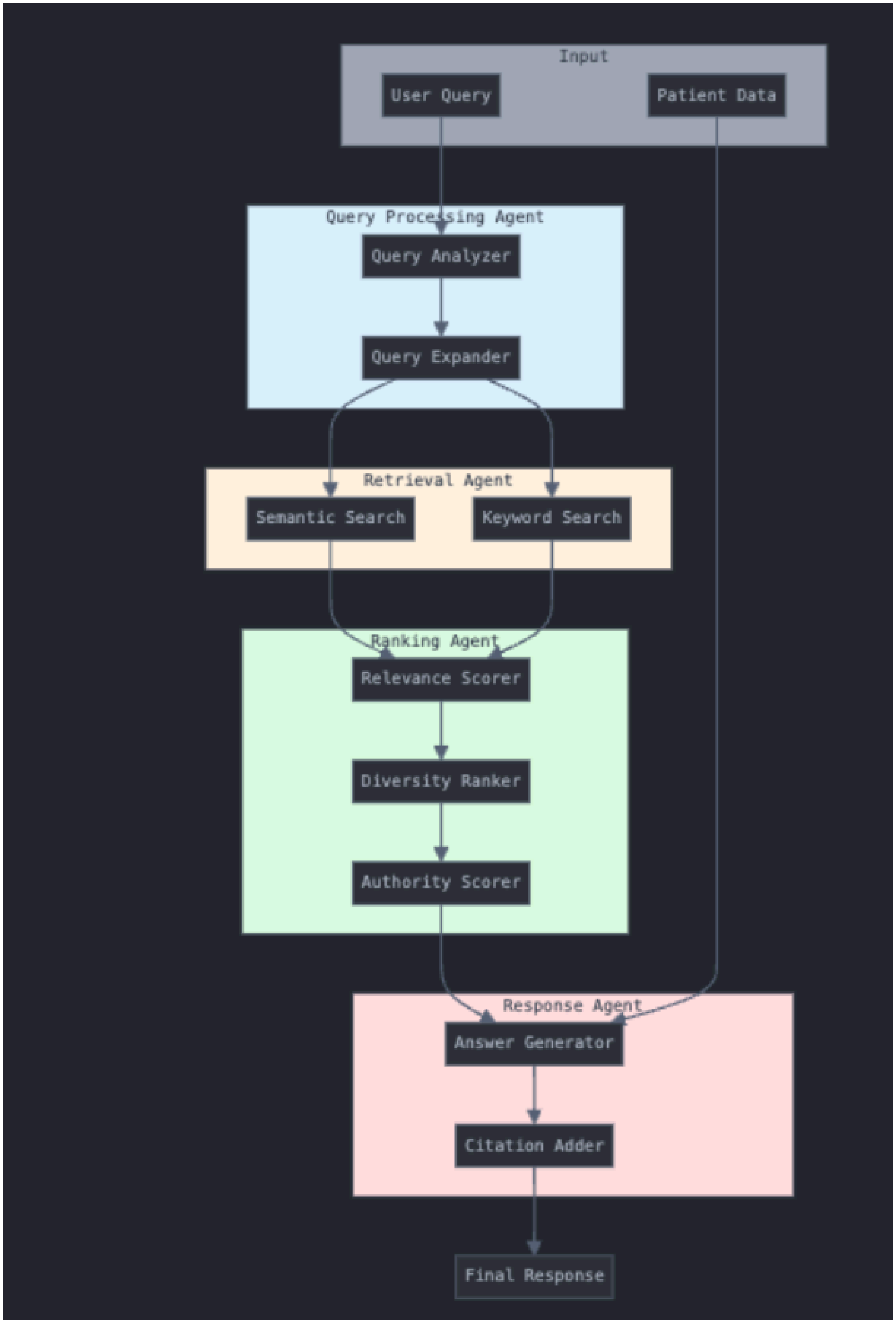
1. User submits query
2. Query Processor analyzes and embeds query
3. Agent Selector chooses relevant agents
4. Agents process query with context
5. Responses are ranked and combined
6. Final response is returned to user
7. Input Stage:
    a. User Query: Initial question from the user
    b. Patient Data: Contextual medical information about the patient
8. Query Processing Agent:
    a. Query Analyzer: Examines query intent and context
    b. Query Expander: Enriches query with relevant medical terms
9. Retrieval Agent:
    a. Semantic Search: Performs embedding-based similarity search
    b. Keyword Search: Traditional term-based search Both methods run in parallel to maximize coverage
10. Ranking Agent:
    a. Relevance Scorer: Evaluates document relevance to query
    b. Diversity Ranker: Ensures varied information sources
    c. Authority Scorer: Weights results based on source credibility
11. Response Agent:
    a. Answer Generator: Synthesizes information with patient context
    b. Citation Adder: Includes reference sources in response

The system uses specialized agents at each stage, with BioBERT for medical text encoding and GPT-4 for natural language understanding and generation. Each stage feeds into the next, with the final output being a contextualized medical response with citations.

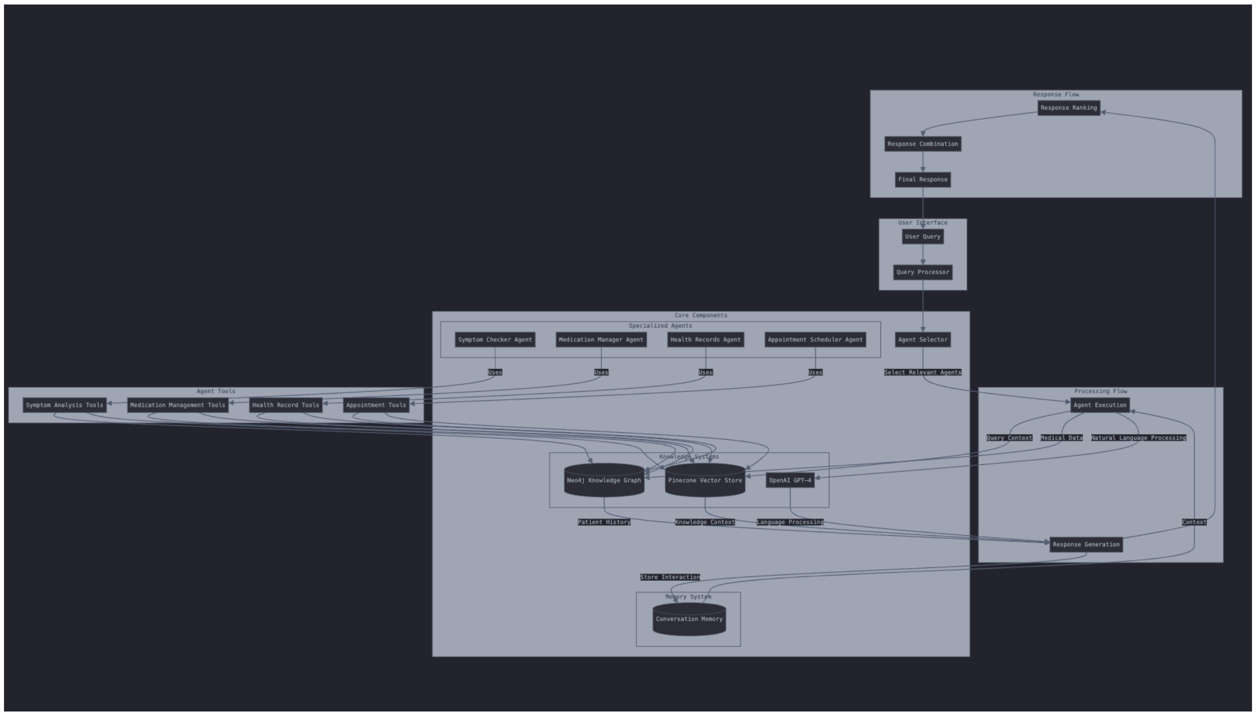
Architecture Diagram

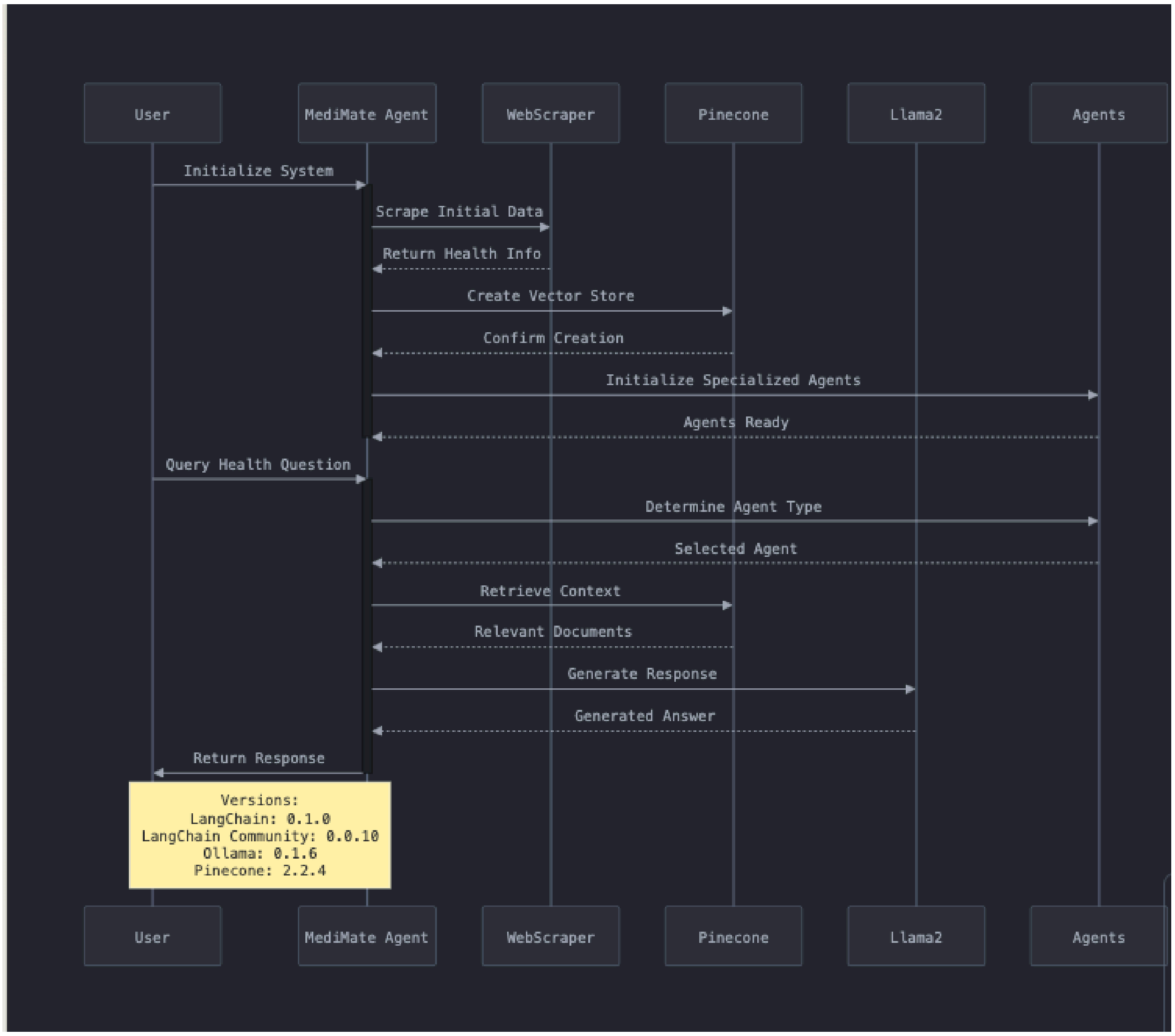

Open with
Vector Store Layer
OpenAI Embeddings
Pinecone Vector DB
Memory System
Conversation History
User Context
LLM Processing
Prompt Templates
OpenAI GPT-4
Specialized Agents
Symptom Checker Agent
Symptom Analysis
Triage Assessment
Medication Manager Agent
Medication Info
Interaction Checker
Health Records Agent
Record Retrieval
Trend Analysis
Appointment Scheduler Agent
Appointment Scheduling
Reminder System
Expense Tracker Agent
Expense Tracking
Budget Analysis

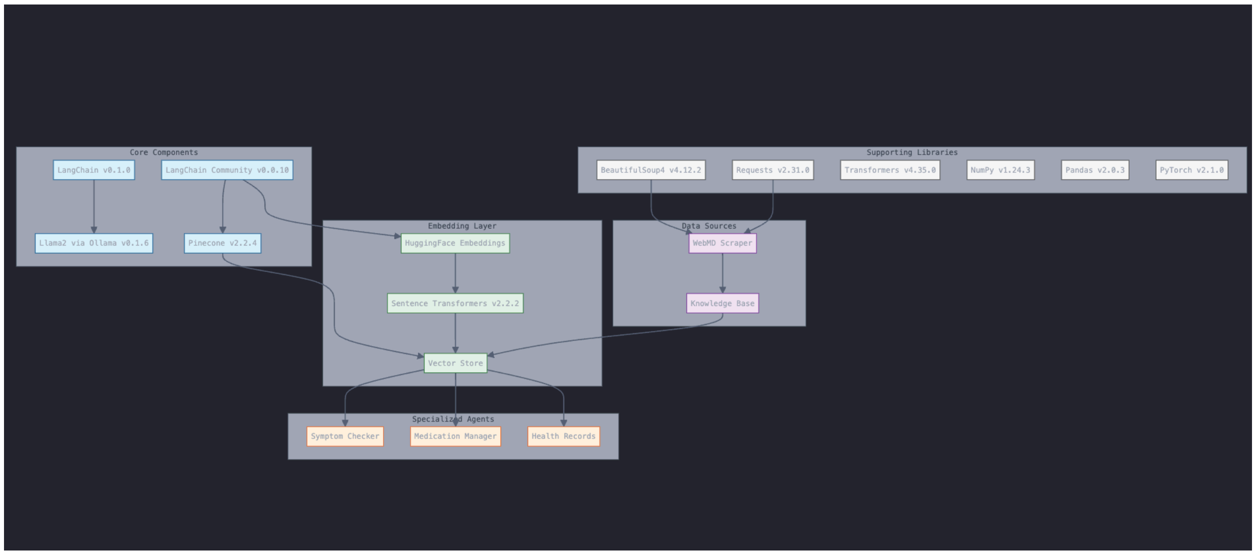

User Query Input
Core System
Query Processor
Vector Store Layer
OpenAI Embeddings
Context
Pinecone Vector DB
Context Manager
Agent Selector
Control
Control
Control
Control
Control
Agent Selection
Specialized Agents
Health Records Agent
Record Retrieval
Trend Analysis
Appointment Scheduler Agent
Appointment Scheduling
Reminder System
Expense Tracker Agent
Expense Tracking
Budget Analysis
Response Ranker & Combiner
Final Response


Input
User Query
Patient Data
Query Processing Agent
Query Analyzer
Query Expander
Retrieval Agent
Semantic Search
Keyword Search
Ranking Agent
Relevance Scorer
Diversity Ranker
Authority Scorer
Response Agent
Answer Generator
Citation Adder
Final Response

1. Knowledge Graph Schema :
    - Shows the relationships between different medical entities
    - Includes patients, conditions, medications, symptoms, etc.
    - Defines key relationships like HAS_CONDITION, TAKES, INTERACTS_WITH
2. Neo4j Integration Code:
    - `MediMateKnowledgeGraph` class for Neo4j operations:
        - Schema setup with constraints
        - CRUD operations for medical data
        - Specialized queries for medical knowledge
        - Integration with existing agent infrastructure
3. Key Features:
    - Medical history tracking
    - Medication interaction checking
    - Symptom-condition relationships
    - Patient record management
    - Treatment and insurance tracking




User
MediMate Agent
WebScraper
Pinecone
Llama2
Agents
Initialize System
Scrape Initial Data
Return Health Info
Create Vector Store
Confirm Creation
Initialize Specialized Agents
Agents Ready
Query Health Question
Determine Agent Type
Selected Agent
Retrieve Context
Relevant Documents
Generate Response
Generated Answer
Return Response
Versions:
LangChain: 0.1.0
LangChain Community: 0.0.10
Ollama: 0.1.6
Pinecone: 2.2.4
User
MediMate Agent
WebScraper
Pinecone
Llama2
Agents


Core Components
LangChain v0.1.0
LangChain Community v0.0.10
Llama2 via Ollama v0.1.6
Pinecone v2.2.4
Supporting Libraries
BeautifulSoup4 v4.12.2
Requests v2.31.0
Transformers v4.35.0
NumPy v1.24.3
Pandas v2.0.3
PyTorch v2.1.0
Embedding Layer
HuggingFace Embeddings
Sentence Transformers v2.2.2
Vector Store
Data Sources
WebMD Scraper
Knowledge Base
Specialized Agents
Symptom Checker
Medication Manager
Health Records