**Project Citadel: Functional Analysis, Architecture & Data Flow**

**Executive Summary**

Project Citadel is a sophisticated RAG (Retrieval-Augmented Generation) system built on a modular Python architecture that combines intelligent web crawling, hierarchical content processing, vector storage, and AI-powered question answering. The system demonstrates enterprise-ready capabilities for transforming unstructured web documentation into a queryable knowledge base.

**1. Functional Analysis**

**1.1 Core System Capabilities**

**Smart Content Ingestion:**

* **Multi-format Support**: Handles regular websites, Markdown/txt files, and XML sitemaps
* **Adaptive Crawling**: Automatically detects content type and applies appropriate crawling strategy
* **Recursive Discovery**: Follows internal links to specified depth for comprehensive site coverage
* **Parallel Processing**: Concurrent crawling with memory-adaptive resource management

**Intelligent Content Processing:**

* **Hierarchical Chunking**: Splits content by document structure (H1 → H2 → H3 → character count)
* **Metadata Extraction**: Captures headers, word counts, character counts, and source URLs
* **Content Optimization**: Ensures all chunks are under specified size limits for optimal retrieval

**Knowledge Base Management:**

* **Vector Storage**: ChromaDB integration with configurable embedding models
* **Semantic Search**: Context-aware document retrieval using vector similarity
* **Batch Operations**: Efficient document insertion and retrieval with configurable batch sizes

**AI-Powered Querying:**

* **RAG Agent**: Pydantic AI agent with tool-calling capabilities
* **Context-Aware Responses**: Combines retrieved documentation with LLM reasoning
* **Flexible Model Support**: Configurable OpenAI model selection

**1.2 User Workflows**

**Documentation Ingestion Workflow:**

1. User provides URL (website, txt file, or sitemap)
2. System detects content type automatically
3. Appropriate crawler executes with parallel processing
4. Content is hierarchically chunked and vectorized
5. Metadata is extracted and stored alongside content
6. Confirmation of successful ingestion with statistics

**Query & Response Workflow:**

1. User submits natural language question
2. RAG agent analyzes question and formulates search query
3. Vector database retrieval finds relevant documentation chunks
4. Retrieved context is formatted and provided to LLM
5. LLM generates response combining context with reasoning
6. Response is returned to user with source attribution

**2. Architecture Analysis**

**2.1 System Components**

graphql

┌─────────────────────────────────────────────────────────────┐

│ Project Citadel Architecture │

├─────────────────────────────────────────────────────────────┤

│ │

│ ┌─────────────────┐ ┌─────────────────┐ │

│ │ Web Crawler │ │ RAG Agent │ │

│ │ Engine │ │ System │ │

│ └─────────────────┘ └─────────────────┘ │

│ │ │ │

│ ▼ ▼ │

│ ┌─────────────────┐ ┌─────────────────┐ │

│ │ Content │ │ Query │ │

│ │ Processor │ │ Interface │ │

│ └─────────────────┘ └─────────────────┘ │

│ │ │ │

│ ▼ ▼ │

│ ┌─────────────────────────────────────────┐ │

│ │ ChromaDB Vector Store │ │

│ │ (Embeddings + Metadata) │ │

│ └─────────────────────────────────────────┘ │

│ │

└─────────────────────────────────────────────────────────────┘

**2.2 Technology Stack**

**Core Framework:**

* **Python 3.11+**: Runtime environment
* **AsyncIO**: Asynchronous processing for scalability
* **Crawl4AI**: Advanced web crawling with Playwright backend
* **Pydantic AI**: Agent framework with tool integration

**Data Processing:**

* **ChromaDB**: Vector database for embeddings and metadata
* **SentenceTransformers**: Embedding model integration
* **Markdown Processing**: Hierarchical content parsing

**AI/ML Integration:**

* **OpenAI API**: LLM inference and embeddings
* **Vector Similarity**: Cosine distance for semantic search
* **RAG Pipeline**: Retrieval-augmented generation workflow

**Infrastructure:**

* **Memory Management**: Adaptive resource allocation
* **Batch Processing**: Configurable batch sizes for performance
* **Session Management**: Browser session reuse and pooling

**2.3 Modular Design**

**Core Modules:**

* insert\_docs.py: Main ingestion orchestrator
* rag\_agent.py: AI agent with retrieval capabilities
* utils.py: Shared utilities for ChromaDB operations

**Specialized Crawlers:**

* 1-crawl\_single\_page.py: Single page extraction
* 2-crawl\_docs\_sequential.py: Sequential crawling with session reuse
* 3-crawl\_sitemap\_in\_parallel.py: Parallel sitemap processing
* 4-crawl\_llms\_txt.py: Markdown/text file processing
* 5-crawl\_site\_recursively.py: Recursive site crawling

**3. Data Flow Analysis**

**3.1 Ingestion Pipeline**

css

URL Input → Content Detection → Crawler Selection → Parallel Processing

↓

Content Extraction → Markdown Conversion → Hierarchical Chunking

↓

Metadata Extraction → Vector Embedding → ChromaDB Storage

↓

Batch Insertion → Index Update → Ingestion Complete

**Detailed Flow:**

1. **Input Processing**
   * URL validation and type detection
   * Configuration parameter parsing
   * Resource allocation (memory, concurrency)
2. **Content Acquisition**
   * Browser automation via Playwright
   * Parallel session management
   * Memory-adaptive dispatching
   * Error handling and retry logic
3. **Content Transformation**
   * HTML to Markdown conversion
   * Hierarchical structure parsing
   * Smart chunking by headers (H1→H2→H3→character)
   * Metadata extraction (headers, counts, source)
4. **Vector Processing**
   * Embedding generation via SentenceTransformers
   * ChromaDB collection management
   * Batch insertion for performance
   * Index optimization

**3.2 Query Pipeline**

sql

User Question → Search Query Formulation → Vector Similarity Search

↓

Context Retrieval → Result Ranking → Context Formatting

↓

LLM Prompt Construction → Model Inference → Response Generation

↓

Source Attribution → Response Delivery

**Detailed Flow:**

1. **Query Processing**
   * Natural language question analysis
   * Search strategy formulation
   * Parameter configuration (n\_results, filters)
2. **Retrieval Phase**
   * Vector similarity computation
   * Ranked result extraction
   * Metadata preservation
   * Context compilation
3. **Generation Phase**
   * Prompt engineering with retrieved context
   * OpenAI API interaction
   * Response synthesis
   * Quality validation
4. **Response Delivery**
   * Source attribution
   * Confidence scoring
   * Formatted output

**3.3 Memory and Performance Management**

**Resource Optimization:**

* **Memory Monitoring**: Real-time PSUtil tracking with 70% threshold
* **Adaptive Batching**: Dynamic batch size adjustment based on system load
* **Session Pooling**: Browser session reuse to minimize overhead
* **Concurrent Limits**: Configurable parallelism with fallback mechanisms

**Scalability Features:**

* **Depth Control**: Configurable recursion depth for large sites
* **Batch Processing**: ChromaDB operations optimized for bulk insertion
* **Cache Management**: Bypass caching for real-time data
* **Error Recovery**: Graceful handling of failed crawl attempts

**4. Key Technical Innovations**

**4.1 Intelligent Content Detection**

* Automatic URL type detection (sitemap, txt, regular site)
* Context-aware crawler selection
* Hierarchical chunking strategy

**4.2 Performance Optimization**

* Memory-adaptive dispatching
* Parallel processing with resource management
* Batch operations for vector database

**4.3 RAG Integration**

* Tool-calling architecture with Pydantic AI
* Context-aware response generation
* Flexible model configuration

**5. Enterprise Readiness Assessment**

**Strengths:**

* ✅ Modular, extensible architecture
* ✅ Robust error handling and resource management
* ✅ Configurable parameters for enterprise scaling
* ✅ Comprehensive metadata tracking
* ✅ Production-ready vector database integration

**Next Phase Considerations:**

* Authentication and access control
* Multi-tenant collection management
* Enhanced monitoring and logging
* API endpoint development
* Horizontal scaling architecture
* Enterprise security compliance

This architecture provides a solid foundation for scaling to enterprise requirements while maintaining the flexibility and performance demonstrated in the proof-of-concept phase.

Compare with GPT-4.1