Smart Document Q&A System

System Architecture Documentation

---

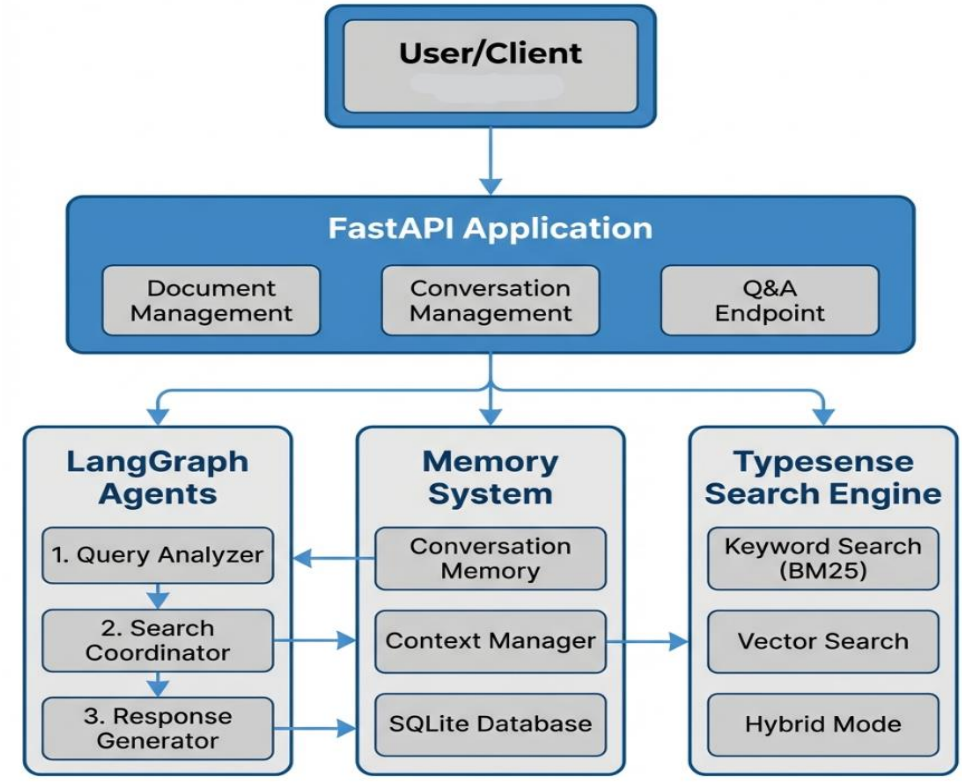
1. Overview

The **Smart Document Q&A System** is a production-oriented, multi-agent platform designed to provide accurate, context-aware question answering over large, unstructured document collections. The system combines **hybrid search (keyword + semantic)** with **agent-based orchestration** and **intelligent memory management** to ensure high-quality, explainable responses with citations.

The architecture is modular, scalable, and suitable for enterprise use cases such as healthcare, legal, insurance, and internal knowledge management.

---

2. High-Level Architecture



---

3. Core Components

3.1 FastAPI Application Layer (`app/main.py`)

Responsibilities - REST API exposure - Request/response validation (Pydantic) - Error handling and status management - CORS configuration - Orchestration of downstream services

Key Endpoints - `POST /api/v1/documents/upload` – Upload and index documents - `POST /api/v1/ask` – Ask questions over indexed documents - `POST /api/v1/conversations/` – Create/manage conversations - `GET /health` – Health and readiness checks

---

3.2 Agent Orchestration Layer (`app/agents/`)

The system uses **LangGraph** to orchestrate specialized agents with explicit state transitions and traceable reasoning.

3.2.1 Query Analyzer Agent

File: `query_analyzer.py`

Purpose Analyzes the user query to determine intent and optimal retrieval strategy.

Key Functions 1. Classifies query intent (factual, conceptual, comparison, procedural) 2. Selects search strategy (keyword, semantic, hybrid) 3. Reformulates the query if required 4. Records decisions for observability

Sample Output

```
{
  "agent": "QueryAnalyzer",
  "strategy": "semantic",
  "reason": "Conceptual question requiring contextual understanding"
}
```

3.2.2 Search Coordinator Agent

File: `search_coordinator.py`

Purpose Executes hybrid search with dynamically tuned parameters.

Workflow 1. Receives analyzed query and intent 2. Adjusts keyword vs semantic weights 3. Executes search via Typesense 4. Ranks and filters results 5. Prepares source citations

Dynamic Weighting Logic - Factual: Keyword 0.7 | Semantic 0.3 - Conceptual: Keyword 0.3 | Semantic 0.7 - Balanced: Keyword 0.5 | Semantic 0.5

3.2.3 Response Generator Agent

File: `response_generator.py`

Purpose Generates grounded, well-structured answers with citations.

Workflow 1. Receives ranked search results 2. Optimizes context window 3. Synthesizes information across sources 4. Generates final response with references 5. Handles low-confidence or empty-result scenarios

---

3.3 Hybrid Search Engine (`app/search/hybrid_search.py`)

Technology Stack - Typesense - OpenAI Embeddings (`text-embedding-3-small`, 1536 dimensions)

Search Modes - **Keyword (BM25):** Precise term matching - **Semantic (Vector):** Conceptual similarity - **Hybrid:** Weighted combination of both

Hybrid Scoring Formula

```
Score = α × Semantic_Score + (1 − α) × Keyword_Score
```

Index Schema

```
{
  "id": "chunk_uuid",
  "document_id": "doc_uuid",
  "document_name": "filename",
  "content": "text",
  "chunk_index": 0,
  "page_number": null,
  "embedding": [1536 floats]
}
```

3.4 Memory & Context Management (`app/memory/`)

3.4.1 Conversation Memory

Purpose Persistent storage of conversations, messages, and search history.

Key Capabilities - Conversation lifecycle management - Message-level metadata and token tracking - Search strategy logging - Analytics-ready schema

Database Tables - `conversations` - `conversation_messages` - `search_history`

3.4.2 Context Manager

Purpose Maintains an optimal context window under token constraints.

Strategy 1. Preserve most recent messages 2. Include older messages if space allows 3. Summarize older context using LLMs when required 4. Enforce token budget limits

Benefits - Prevents context loss - Avoids token overflow - Improves response consistency

---

3.5 Document Processing (`app/document_processor.py`)

Purpose Transforms raw documents into searchable, semantically meaningful chunks.

Chunking Configuration - Chunk size: 500 characters - Overlap: 50 characters - Sentence-aware boundaries

Supported Formats - TXT - Markdown - Extensible to PDF and DOCX

---

4. End-to-End Data Flow

4.1 Document Ingestion

1. User uploads document
2. API validates format
3. Document is chunked
4. Metadata stored in DB
5. Chunks embedded and indexed
6. Success response returned

4.2 Question Answering

1. User submits question
2. Conversation context retrieved
3. Agent workflow executed
4. Hybrid search performed
5. Answer generated with citations
6. Conversation updated and stored

---

5. Key Design Decisions

Agent-Based Architecture

- Clear separation of responsibilities
- Explainable and debuggable decisions

- Easier future extensibility

Technology Choices

- **LangGraph:** Deterministic agent workflows
- **Typesense:** Native hybrid search with low operational overhead
- **SQLite:** Lightweight persistence, easy migration path

---

6. Performance Characteristics

Typical Latency - Embedding: 100–200 ms - Search: 50–100 ms - LLM Response: 1–3 seconds

Scalability Options - Horizontal API scaling - PostgreSQL migration - Typesense clustering - Redis caching layer

---

7. Security & Compliance

Current Measures - Input validation - ORM-based SQL protection - Environment-based secrets

Planned Enhancements - Authentication & authorization - Rate limiting - Encryption at rest and in transit - Audit logging

---

8. Future Roadmap

Short Term - PDF/DOCX support - Embedding cache - Authentication

Mid Term - Feedback-driven re-ranking - Multimodal documents - Advanced summarization

Long Term - Autonomous multi-agent collaboration - Distributed deployment - Fine-tuned domain embeddings

---

9. Conclusion

This system demonstrates a **production-ready, explainable, and scalable** approach to document-based question answering by combining:

- Agent-driven reasoning
- Hybrid search retrieval
- Persistent memory management
- Enterprise-grade extensibility

It is well-suited for real-world deployments requiring accuracy, traceability, and performance.