### Project Overview

As the System Architect, I'll outline a comprehensive project plan for building an on-premise agentic Retrieval-Augmented Generation (RAG) system using Ollama (for local LLMs) and n8n (for workflow automation). This system will support Chinese language processing, handle PDF imports containing sensitive patient information (e.g., personal data and medical records), and ensure everything runs locally for privacy compliance. The "agentic" aspect involves using AI agents in n8n workflows to dynamically select tools, route queries, and handle multi-step reasoning (e.g., deciding whether to retrieve from the vector DB, rephrase queries, or fallback to general knowledge).

Key assumptions:

- Hardware: A machine with at least 16GB RAM, GPU (NVIDIA/AMD preferred for Ollama acceleration), and sufficient storage for PDFs and models.

- Software: All components will be Dockerized for easy on-premise deployment.

- Vector Database: Use Qdrant (open-source, local-friendly) for storing embeddings.

- LLM/Embeddings: Qwen series (e.g., Qwen2.5) for Chinese support in Ollama; multilingual embeddings like nomic-embed-text or multilingual-e5-large.

- Privacy: No cloud dependencies; encrypt PDFs at rest if needed (via n8n nodes or OS-level tools).

The system will have two main workflows in n8n:

1. \*\*Ingestion Workflow\*\*: Import PDFs, extract/chunk text, embed, and store.

2. \*\*Query/Agent Workflow\*\*: Handle user queries with RAG, using agents for complex interactions.

### Project Plan

The plan is divided into phases with timelines (assuming a solo developer; adjust for team size). Total estimated time: 4-6 weeks.

#### Phase 1: Requirements and Setup (Week 1)

- \*\*Objectives\*\*: Gather specs, install tools, and configure the environment.

- \*\*Steps\*\*:

1. Define requirements: Support Chinese text in PDFs; handle 100-500 PDFs initially; query examples (e.g., "Summarize patient X's medical history").

2. Install Docker and Docker Compose.

3. Set up Ollama: Download and run via Docker; pull models like `qwen:2.5` (for Chinese LLM) and `nomic-embed-text` (for embeddings).

4. Set up n8n: Run via Docker; enable community nodes for Ollama, Qdrant, and PDF handling.

5. Set up Qdrant: Dockerized vector DB for storing patient data embeddings.

6. Test basic connectivity: Run a simple n8n workflow to query Ollama.

- \*\*Resources Needed\*\*: Docker docs, Ollama GitHub, n8n docs.

- \*\*Risks\*\*: Hardware limitations (e.g., slow embeddings without GPU); mitigate by starting with smaller models.

- \*\*Milestone\*\*: Local stack running; basic "hello world" query to Ollama via n8n.

#### Phase 2: Design Architecture (Week 1-2)

- \*\*Objectives\*\*: Architect the system for scalability and security.

- \*\*Steps\*\*:

1. Design ingestion pipeline: PDF upload → Text extraction (using n8n's PDF node or Python script via code node) → Chunking (e.g., 500-token chunks) → Embedding with Ollama → Upsert to Qdrant.

2. Design RAG pipeline: Query input → Embed query → Retrieve top-k chunks from Qdrant → Augment prompt → Generate response with Ollama.

3. Add agentic layer: Use n8n's AI Agent nodes to create a router agent that decides actions (e.g., "If query is medical summary, retrieve from DB; else, use general LLM").

4. Handle Chinese: Ensure models support UTF-8; test with sample Chinese PDFs.

5. Security design: Use n8n credentials for internal APIs; store PDFs in a secure local folder; anonymize data if needed (e.g., via regex in workflows).

- \*\*Diagram (High-Level)\*\*:

```

User Query → n8n Webhook → Agent Router → Embed Query (Ollama) → Retrieve (Qdrant) → Generate (Ollama) → Response

PDF Import → n8n Trigger (File Watch) → Extract/Chunk → Embed → Store in Qdrant

```

- \*\*Milestone\*\*: Architecture document and basic workflow skeletons in n8n.

#### Phase 3: Implementation (Week 2-4)

- \*\*Objectives\*\*: Build and integrate workflows.

- \*\*Steps\*\*:

1. Build Ingestion Workflow:

- Trigger: Manual or file watcher on a local folder.

- Nodes: HTTP/File input → PDF Extract (community node or code) → Text Splitter → Ollama Embed → Qdrant Upsert.

- Handle patient data: Add metadata (e.g., patient ID) to embeddings for filtered retrieval.

2. Build Agentic RAG Workflow:

- Trigger: Webhook or chat interface.

- Nodes: AI Agent → Tool: Embed & Retrieve → Tool: Ollama Generate.

- Agentic features: Add tools for re-ranking results, handling hallucinations (e.g., self-critique loop), or multi-query expansion for better Chinese recall.

3. Integrate Chinese support: Use Qwen for generation; test embeddings on Chinese text.

4. Error handling: Add nodes for retries, logging, and data validation (e.g., check for PII leaks).

- \*\*Testing During Implementation\*\*: Unit test each node (e.g., embed a sample PDF chunk).

- \*\*Milestone\*\*: Fully functional workflows; ingest 10 sample PDFs and query successfully.

#### Phase 4: Testing and Optimization (Week 4-5)

- \*\*Objectives\*\*: Ensure reliability, performance, and privacy.

- \*\*Steps\*\*:

1. Functional testing: Ingest real-like PDFs; query in Chinese (e.g., "患者X的病史是什么？").

2. Performance testing: Measure latency; optimize chunk size or model quantization.

3. Security audit: Scan for data exposure; ensure no external calls.

4. Edge cases: Handle large PDFs, corrupted files, non-Chinese mixed text.

5. Scalability: Test with 100+ PDFs; monitor resource usage.

- \*\*Tools for Testing\*\*: n8n's built-in debugger; Ollama metrics.

- \*\*Milestone\*\*: 95% query accuracy on test data; no crashes.

#### Phase 5: Deployment and Maintenance (Week 5-6)

- \*\*Objectives\*\*: Go live and plan ongoing support.

- \*\*Steps\*\*:

1. Deploy: Use Docker Compose for one-command startup; set up systemd for auto-start.

2. User Interface: Expose n8n chat or build a simple frontend (e.g., Streamlit integrated via n8n).

3. Documentation: Workflow exports, user guide for importing PDFs.

4. Maintenance: Schedule backups; monitor logs; update models via Ollama.

5. Compliance: Document how the system handles sensitive data (e.g., GDPR/HIPAA analogs for China).

- \*\*Milestone\*\*: System running on production hardware; handover to ops.

### Checklist

Use this as a trackable list during the project. Mark items as [ ] To Do, [x] Done.

#### Environment Setup

- [ ] Install Docker and Docker Compose.

- [ ] Run Ollama container and pull Qwen2.5 model.

- [ ] Run n8n container and install community nodes (Ollama, Qdrant, PDF).

- [ ] Run Qdrant container and create a collection for patient embeddings.

- [ ] Verify GPU acceleration in Ollama (if available).

- [ ] Set up secure local storage folder for PDFs.

#### Model and Language Support

- [ ] Download multilingual embedding model (e.g., nomic-embed-text).

- [ ] Test Chinese text processing: Embed and generate a sample Chinese sentence.

- [ ] Quantize models for performance (e.g., 4-bit Qwen).

#### Workflow Development

- [ ] Build ingestion workflow: PDF → Extract → Chunk → Embed → Store.

- [ ] Add metadata handling for patient IDs in embeddings.

- [ ] Build RAG query workflow: Query → Embed → Retrieve → Generate.

- [ ] Implement agentic routing: Add AI Agent node with tools for retrieval and fallback.

- [ ] Integrate error handling and logging nodes.

#### Testing and Security

- [ ] Ingest 10+ sample PDFs with mock patient data.

- [ ] Run 50+ test queries, including Chinese and mixed-language.

- [ ] Verify retrieval accuracy (e.g., top-5 recall >80%).

- [ ] Audit for privacy: No data leaves the local machine.

- [ ] Test failure modes: Invalid PDFs, high load.

- [ ] Optimize: Reduce latency under 5 seconds per query.

#### Deployment

- [ ] Create Docker Compose file for all services.

- [ ] Set up auto-backups for Qdrant and PDFs.

- [ ] Document import process (e.g., drop PDFs in folder).

- [ ] Train users on querying the system.

- [ ] Monitor initial usage and fix bugs.

### How to Do It: Step-by-Step Guide

Follow these steps to implement the core system. Reference n8n docs for node details.

1. \*\*Install Dependencies\*\*:

- Download Docker: `sudo apt install docker docker-compose` (on Linux).

- Ollama: `docker run -d -v ollama:/root/.ollama -p 11434:11434 --name ollama ollama/ollama`.

- Pull models: `docker exec -it ollama ollama pull qwen:2.5`.

- n8n: `docker run -d --name n8n -p 5678:5678 -v ~/.n8n:/home/node/.n8n n8nio/n8n`.

- Qdrant: `docker run -d -p 6333:6333 qdrant/qdrant`.

2. \*\*Create Ingestion Workflow in n8n\*\* (Access n8n at localhost:5678):

- Add Trigger: "On File Change" node watching a PDF folder.

- Add "Read PDF" node (community) to extract text.

- Add "Split Text" node: Chunk into 500 tokens.

- Add "Ollama Embed" node: Use nomic-embed-text.

- Add "Qdrant Upsert" node: Store vectors with metadata (e.g., {"patient\_id": "123"}).

- Activate workflow.

3. \*\*Import PDFs\*\*:

- Drop PDFs into the watched folder.

- Workflow auto-processes: Extracts text (handles Chinese via UTF-8), embeds, stores.

- For bulk: Use n8n's HTTP node to batch import.

4. \*\*Create Agentic RAG Workflow\*\*:

- Add Trigger: "Webhook" for queries (e.g., POST /query with JSON {"question": "患者信息"}).

- Add "AI Agent" node: Set LLM to Ollama (Qwen), add tools:

- Tool 1: "Retrieve" – Embed query → Qdrant Search (top-5) → Return chunks.

- Tool 2: "Generate" – Prompt with retrieved chunks.

- Agent prompt: "You are a medical assistant. Use retrieval for patient data queries."

- Add output node: Return response.

- For agentic complexity: Add loops for refinement (e.g., if low confidence, re-query).

5. \*\*Query the System\*\*:

- Send a query via curl: `curl -X POST http://localhost:5678/webhook/... -d '{"question": "Summarize patient X medical records"}'`.

- Response includes RAG-augmented answer in Chinese if needed.

6. \*\*Troubleshoot and Scale\*\*:

- Logs: Check Docker logs for Ollama/n8n.

- Scale: Add more Qdrant shards; use larger models.

- Updates: Pull new Ollama models periodically.

This setup ensures a secure, local system. If issues arise (e.g., model performance), iterate based on tests. For advanced features like voice input, extend n8n workflows later.