Dynamic Agentic Systems

Overview

This system enables:

- Querying across documents (e.g. legal, financial) and structured databases (e.g. stock prices).
- Dynamic selection between different AI personas.
- Accurate responses with citations (page numbers + screenshots).
- Suggested follow-up queries to sustain conversational flow.
- Scalable architecture for adding more KBs and LLMs in the frontend.

Core Functionalities

1. Multi-Knowledge Base Integration

- **Document Knowledge Base**: Legal and financial documents (PDFs).
- **Database Knowledge Base**: Year-long stock market data (CSV or SQL DB).

2. AI Personas

- Domain-specific LLM agents:
    - Financial Analyst (for math, stocks)
    - Legal Advisor (for compliance, contracts)
    - General Assistant (mixed queries)
- Each persona is backed by a selected LLM (e.g., OpenAI, Claude, DeepSeek).

3. Query Types

- Mathematical: Stock price trends, moving averages, thresholds.
- Factual: Specific document-based questions.
- Conversational: Multi-step, suggestion-driven dialogs.

4. Accuracy-Driven RAG

- Vector DB (Pinecone) stores:
    - Chunk content
    - Metadata (page number, image of chunk, title, section)
- On match, response includes:

    - Answer
    - Exact **page number**
    - **Screenshot** (from PDF render)

5. Speed & Precision

- Mathematical operations offloaded to dedicated Python nodes (not LLMs).
- Doc-based queries served from indexed vector stores.
- LangGraph routes query to the right pipeline.

LangGraph Architecture

```
None
graph TD
    UI[User Interface]
    RouterNode[Router Node]
    DocNode[Document RAG Node]
    DBNode[Database Node]
    MathNode[Math Execution Node]
    PersonaSelector[Persona Selector Node]
    SuggestionNode[Suggested Queries Generator]
    AnswerFormatter[Answer + Metadata Formatter]

    UI --> PersonaSelector
    PersonaSelector --> RouterNode
    RouterNode --> DocNode
    RouterNode --> DBNode
    RouterNode --> MathNode
    RouterNode --> SuggestionNode

    DocNode --> AnswerFormatter
    DBNode --> MathNode
    MathNode --> AnswerFormatter
    SuggestionNode --> UI
    AnswerFormatter --> UI
```

Node Responsibilities:

- **Router Node**: Routes queries to the right node(s) based on intent classification.
- **Doc Node**: Uses Pinecone + OCR to retrieve documents with page # and image.
- **DB Node**: Runs SQL queries for historical data like prices.
- **Math Node**: Handles computation-heavy queries like MA calculations.
- **Persona Selector**: Routes the request to the selected LLM/persona backend.
- **Suggestion Node**: Generates recommended queries to keep the flow.
- **Answer Formatter**: Adds source metadata and screenshot into final answer.

Frontend Interface Requirements

Key Features:

- Upload or attach:
    - PDFs, CSVs, SQL/NoSQL DB connections
- Select LLM provider per persona
- Live test queries + trace answer pipeline
- Visual flow of query processing (debugging)
- Add new LLMs via API keys

UI Layout

- **Left Panel**: KB Sources and Persona Management
- **Center**: Chat + Answer + Suggested Queries
- **Right Panel**: Metadata and Source (PDF page preview)

Dynamic Expansion Handling

Adding New DB

- Adds new DBNode instance
- Automatically attaches to RouterNode via intent mapping

Adding New Document

- Gets chunked and indexed into Pinecone
- DocNode automatically reruns vector search on entire corpus

Adding New LLM/Persona

- PersonaSelector node is updated
- UI reflects new persona toggle

Sample Query Flow

Query: "Tell me the Moving Average of MSFT from March to May 2024"

- Router → DBNode → MathNode → Formatter → UI
- Suggested Query: "When did MSFT cross its 200-day MA in 2024?"

Query: "What clause handles data breach retention?"

- Router → DocNode → Formatter → UI
- Formatter returns page number, screenshot
- Suggested Query: "Are there penalties for breach of NDA clauses?"

Backend Stack

- LangGraph (agent orchestration)
- Pinecone (vector store)
- PostgreSQL / MongoDB (DB data)
- OCR (PDF screenshot extraction)
- FastAPI / Node.js (API backend)

Frontend Stack

- Next.js (React)
- Tailwind / ShadCN
- WebSocket for live tracing