**Multi-Agentic RAG System Operational Plan**

**1. Introduction**

This document outlines the operational plan for a multi-agentic Retrieval-Augmented Generation (RAG) system comprising seven agents: DataIngestion, Embedding, Storage (backend); Query, Retriever, Answer, and Feedback (frontend). The system processes course data to generate syllabus-aligned, exam-focused, personalized answers, incorporating user feedback for iterative improvement. The plan details each agent’s role, tools, interactions, and data flow, emphasizing modularity and clarity to ensure efficient collaboration. The backend prepares and stores data, the frontend handles user queries and responses, and center data stores (ChromaDB and SQLite3/MongoDB) facilitate data exchange.

**2. System Components**

***2.1 Agents***

Backend Agents:

* DataIngestion: Validates and parses admin-provided course data.
* Embedding: Converts unstructured data into embeddings for semantic search.
* Storage: Stores structured data for precise retrieval.

Frontend Agents:

* Query: Preprocesses user queries.
* Retriever: Fetches relevant data from data stores.
* Answer: Generates personalized, syllabus-aligned answers.
* Feedback: Processes user feedback to refine future responses.

***2.2 Tools***

* Data Parser (DataIngestion): Parses and validates course data inputs.
* Database Loader (Storage): Inserts structured data into SQLite3/MongoDB.
* Text Chunker, Embedding Model (Embedding): Chunks text and generates embeddings.
* ChromaDB Query Tool (Retriever): Queries ChromaDB and SQLite3/MongoDB.
* Language Model (Answer): Generates answers using Gemini.
* Feedback Parser (Feedback): Extracts preferences from user feedback.

***2.3 Data Stores***

* ChromaDB: Stores vector embeddings of unstructured data (e.g., syllabus, textbooks).
* SQLite3/MongoDB: Stores structured data (e.g., course metadata, CO-PO mappings, questions).

***2.4 Inputs and Outputs***

Inputs:

* + Course Data: Admin-provided data (e.g., syllabus, textbooks, CO-PO mappings).
  + User Query: User questions (e.g., “Explain databases in CS101”).
  + User Feedback: User suggestions (e.g., “Use more examples”).

Output:

* + Personalized Answer: Syllabus-aligned, exam-focused response.

**3. Operational Flow**

The system operates through two workflows—**backend (data preparation) and frontend (query processing)**—connected via center data stores. The Feedback agent enables iterative personalization.

***3.1 Backend Workflow***

The backend agents prepare Course Data for storage, ensuring it’s accessible for frontend retrieval.

***3.1.1 DataIngestion Agent***

* + Function: Validates and parses Course Data inputs.
  + Process:
  + Accepts raw inputs (e.g., CSV files, PDFs, URLs) from admins.
  + Uses the Data Parser tool to:
  + Validate data integrity (e.g., checks for missing fields, malformed entries).
  + Parse into structured data (e.g., metadata, CO-PO mappings) and unstructured data (e.g., syllabus text, textbook content).
  + Routes structured data to StorageAgent and unstructured data to EmbeddingAgent.

Output:

* + Structured data (e.g., JSON with course metadata).
  + Unstructured data (e.g., raw syllabus text).

-Interactions:

* + Sends structured data to StorageAgent.
  + Sends unstructured data to EmbeddingAgent.

***3.1.2 StorageAgent***

Function: Stores structured data in SQLite3/MongoDB.

Process:

* + Receives structured data (e.g., course codes, CO-PO mappings, question banks).
  + Uses the Database Loader tool to:
  + Format data for database insertion (e.g., creates tables or documents).
  + Insert data into SQLite3/MongoDB with appropriate indexing (e.g., by course code).
  + Ensures data is optimized for fast retrieval.

Output: Structured data stored in SQLite3/MongoDB.

-Interactions:

* + Populates SQLite3/MongoDB, accessed by RetrieverAgent.
  + Operates in parallel with EmbeddingAgent.

***3.1.3 EmbeddingAgent***

-Function: Processes unstructured data into embeddings for ChromaDB.

-Process:

* + Receives unstructured data (e.g., syllabus, textbooks, web content).
  + Uses the Text Chunker tool to split text into chunks (e.g., 500-word segments).
  + Uses the Embedding Model tool (e.g., Sentence-BERT) to generate vector embeddings.
  + Stores embeddings in ChromaDB, indexed by course and content type.

Output: Vector embeddings in ChromaDB.

Interactions:

* + Populates ChromaDB, accessed by RetrieverAgent.
  + Works in parallel with StorageAgent.

***3.2 Frontend Workflow***

The frontend agents process user queries to deliver answers and incorporate feedback.

***3.2.1 QueryAgent***

Function: Preprocesses user queries.

Process:

* + Receives User Query (e.g., “Explain topic X from course Y”).
  + Validates query (e.g., checks for course context, clarity).
  + Extracts metadata (e.g., course code, topic keywords).
  + Reformats query for retrieval (e.g., structured query terms).

Output: Processed query (e.g., {course: “CS101”, topic: “databases”}).

-Interactions:

* + Sends processed query to RetrieverAgent.

***3.2.2 RetrieverAgent***

-Function: Retrieves relevant data from data stores.

-Process:

* + Receives processed query.
  + Uses the ChromaDB Query Tool to:
  + Perform semantic search in ChromaDB (e.g., matches query to syllabus chunks).
  + Query SQLite3/MongoDB for structured data (e.g., CO-PO mappings).
  + Ranks results for syllabus alignment and relevance.
  + Combines data into a context package.

Output: Context package (e.g., syllabus chunks, CO-PO mappings).

Interactions:

* + Fetches data from ChromaDB and SQLite3/MongoDB.
  + Sends context to AnswerAgent.

3.2.3 AnswerAgent

Function: Generates syllabus-aligned, exam-focused answers.

Process:

* + Receives context package (unstructured and structured data).
  + Uses the Language Model (Gemini) to:
    - Synthesize a response aligned with CO-PO objectives.
    - Ensure exam focus (e.g., concise, structured).
    - Apply user preferences (e.g., formal tone).

Formats response (e.g., bullet points).

Output: Personalized Answer.

Interactions:

* + Receives context from RetrieverAgent.
  + Applies preferences from FeedbackAgent.
  + Delivers answer, triggering feedback.

***3.2.4 FeedbackAgent***

Function: Processes user feedback for personalization.

Process:

* + Receives User Feedback (e.g., “Make answers concise”).
  + Uses the Feedback Parser tool to extract preferences (e.g., tone: concise).
  + Updates user profile with preferences.
  + Sends preferences to AnswerAgent.

Output: Updated preferences (e.g., {tone: “concise”}).

Interactions:

* + Sends preferences to AnswerAgent.
  + Closes feedback loop.

***3.3 Center Data Stores***

The data stores connect backend and frontend workflows.

***3.3.1 ChromaDB***

Function: Stores embeddings for semantic search.

Process:

* + Receives embeddings from EmbeddingAgent.
  + Stores with metadata (e.g., course code).
  + Provides unstructured data to RetrieverAgent and AnswerAgent.

Interactions:

* + Written by EmbeddingAgent.
  + Read by RetrieverAgent and AnswerAgent.

***3.3.2 SQLite3/MongoDB***

Function: Stores structured data for precise retrieval.

Process:

* + Receives structured data from StorageAgent.
  + Organizes into tables/documents.
  + Provides data to RetrieverAgent and AnswerAgent.

Interactions:

* + Written by StorageAgent.
  + Read by RetrieverAgent and AnswerAgent.

**4. Collaborative Workflow**

1. Backend:

* + DataIngestion parses Course Data, routing to StorageAgent (structured) and EmbeddingAgent (unstructured).
  + StorageAgent and EmbeddingAgent populate SQLite3/MongoDB and ChromaDB, respectively.

2. Frontend:

* + QueryAgent preprocesses User Query.
  + RetrieverAgent fetches data from data stores.
  + AnswerAgent generates Personalized Answer.

3. Feedback Loop:

* + FeedbackAgent processes User Feedback, updating AnswerAgent preferences.

4. Center:

* + Data stores act as a hub, enabling backend-to-frontend data flow.

5. Modularity and Debugging

Modularity:

* + Each agent has a single responsibility.
  + Tools encapsulate specific tasks.
  + Backend/frontend are independent, connected via data stores.
  + Feedback loop affects only AnswerAgent.

Debugging:

* + Issues are traceable (e.g., incorrect retrieval → RetrieverAgent).
  + Data stores allow inspection of stored/retrieved data.
  + Feedback loop supports iterative fixes.

6. Example Scenario

Course Data: Admin uploads CS101 data (syllabus, CO-PO mappings).

* + DataIngestion validates and parses.
  + StorageAgent stores mappings in SQLite3.
  + EmbeddingAgent embeds syllabus in ChromaDB.

**Query: “Explain databases in CS101.”**

* + QueryAgent extracts {course: “CS101”, topic: “databases”}.
  + RetrieverAgent fetches syllabus chunks and CO-PO mappings.
  + AnswerAgent generates a CO-PO-aligned answer.

**Feedback: “Use more examples.”**

* + FeedbackAgent updates preferences.
  + AnswerAgent applies in future responses.