***Note: Please uploading file before using.***

***Available at:*** [*http://130.211.241.123/*](http://130.211.241.123/)

**Overview**

This system integrates n8n with a React frontend to build a knowledge base that:

* Uploads documents (via Google Drive),
* Processes them with an LLM to create embeddings,
* Stores embeddings and metadata (including name, file id, feedback scores) in Qdrant,
* Allows querying via a chat interface,
* Captures user feedback to improve responses.

**Key idea of improve RAG:** store feedbackscore=1 in metadata, then if user dislike take id of point then upsert feedbackscore=-1-> search again and filter where feedbackscore is greater than 0

**Architecture**

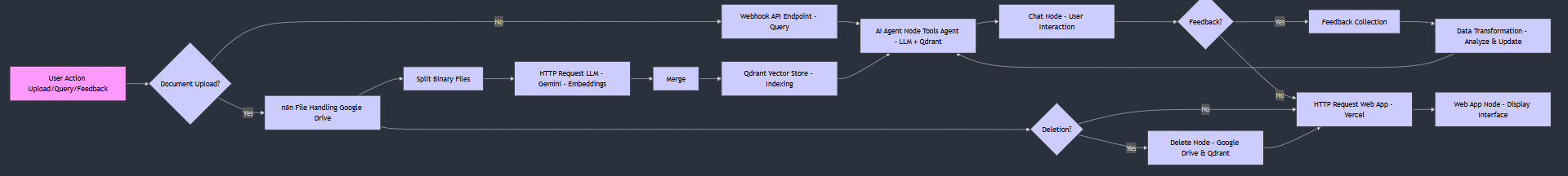
****

Figure 1. Overview of project flow

A computer screen shot of a computer

AI-generated content may be incorrect.

A diagram of a diagram

AI-generated content may be incorrect.

*\*Zoom in*

* **n8n Workflow**
  + **File Upload & Processing**: Files are uploaded to Google Drive, processed via an LLM for embeddings, and stored in Qdrant with metadata **must have** (file ID, name, feedbackscore – default = 1).
  + **File Deletion**: Removes files from both Google Drive and Qdrant.
  + **Chat & Feedback**: Provides endpoints (/chat, /dislike) for querying and capturing user feedback. The output of Agent always JSON format with *id, fileid, and output.* (id is id of Qdrant point in context, fileid is file id of document in context, and output is the response)
  + **Data Tranformation:** If user dislike, reduce feedbackscore of document by 1 that use in Qdrant then Search with filter *feedbackscore* is greater than 0
* **React Frontend**
  + **FileManager.jsx**: Manages file selection, upload, listing, and deletion.
  + **Chat.jsx**: Offers a chat interface to send queries and receive responses, including like/dislike feedback.

**Workflow Details**

1. **Handle User Query / User Feedback (Remember History Chat)**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Purpose:**

* Provide a conversational interface where users ask questions, and the system responds using a Retrieval-Augmented Generation (RAG) approach.
* Collect user feedback (like/dislike) and update Qdrant to refine answers.

**Key Steps:**

1. **Chat Node:** Captures the user’s question.
2. **Dislike node:** Capture the user feedback.
3. **HTTP Request / Condition Node:** Check for user dislike is response with RAG or normal response (RAG response will contain *id* of poin and *fileid*).
4. **AI Agent / Tools Agent Node:** Manages the conversation context, memory, and calls the LLM (Google Gemini-flash-2.0) for generating answers in JSON format with *id, fileid, and output*.
5. **Normal RAG Node:** Perform retrieval from Qdrant based on the user’s query to find relevant chunks of information.
6. **Embeddings / Memory Nodes:** Keep track of conversation history and embed new content or user feedback.
7. **Handle Dislike / Feedback Nodes:** When a user dislikes an answer, the **HTTP Request(2)** updates the feedbackScore = -1 of the previous document that use in context, improving query responses.
8. **Re-search Node:** Search with filter *feedbackscore* is greater than 0.

**2. Delete File on Qdrant and Google Drive**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Purpose:**

* Remove both the stored file from Google Drive and the corresponding record in Qdrant when a user decides to delete a document.

**Key Steps:**

1. **Trigger (DelFile):** This node listens for a “delete” action (e.g., user clicks “delete” on a web interface).
2. **Google Drive Delete Node:** Deletes the specified file from Google Drive by referencing the file’s ID.
3. **HTTP Request (Qdrant) Node:** Sends a request to Qdrant to remove the corresponding point (document metadata/embedding) from the vector store.

**3. Get Data -> Store in Google Drive and Embed Document -> Store in Qdrant**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Purpose:**

* Handle file ingestion by uploading to Google Drive, creating embeddings with a model (Google Gemini), and storing the resulting metadata and vectors in Qdrant.

**Key Steps:**

1. **AddFile Node:** Receives the uploaded document.
2. **Split Binary Files Node:** Splits large documents into manageable chunks for processing.
3. **Google Drive Upload Node:** Uploads each chunk or the full file to Google Drive for storage.
4. **Edit Fields / Merge Nodes:** Organize metadata (file ID, file name, feedbackscore, etc.) that will be used for indexing.
5. **Embeddings Google Gemini Node:** Calls the Gemini model (or another embedding service) to transform text chunks into vector embeddings.
6. **Qdrant Vector Store Node:** Sends the embeddings and metadata (document ID, filename) to Qdrant for indexing.

**4. List All Files in Google Drive**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Purpose:**

* Retrieve a list of files stored in Google Drive so the system can display them to users or manage them further.

**Key Steps:**

1. **ListFile Node (Trigger):** Initiates a “list files” operation.
2. **Google Drive Node:** Searches the specified folder or drive for all files.
3. **Output:** Returns a structured list of files (e.g., file name, ID, type) which can be displayed to users or used in subsequent workflows (like delete).

**Importing, Modifying, and Reusing the RAG n8n Workflow**

**1. Importing the Workflow**

1. **Export/Obtain the JSON File:**
   * Ensure you have the RAG.json file (the workflow JSON) saved locally.
2. **Open n8n:**
   * Log in to your n8n instance (either self-hosted or n8n.cloud).
3. **Import the Workflow:**
   * In the n8n UI, click on the "Import" button.
   * Select the RAG.json file.
   * Verify that the workflow loads correctly, showing all nodes and connections.
4. **Review the Workflow:**
   * Check that nodes (e.g., Google Drive, HTTP Request, Webhook, Function, Langchain nodes) appear as expected.
   * Ensure that all connections are intact.

**2. Modifying the Workflow**

1. **Update Credentials:**
   * For nodes that require credentials (e.g., Google Drive, Qdrant, Google Gemini API), open each node’s settings and select or update the appropriate credential.
   * If needed, create new credentials in n8n and assign them to the corresponding nodes.
2. **Adjust Parameters:**
   * Review each node’s parameters (like folder IDs in Google Drive nodes, API endpoints, file IDs, and model names).
   * Modify these values to match your environment (e.g., change folder IDs, collection names, or webhook paths).
   * For example, if you want to change the folder where files are listed, update the folderId parameter in the “Google Drive1” node.
3. **Customize Business Logic:**
   * Open Function or Code nodes to review the custom JavaScript code. Modify logic as needed to suit your specific processing requirements.
   * Adjust conditions in “If” nodes or HTTP request body templates if your target APIs or feedback mechanisms change.
4. **Test the Workflow:**
   * Execute each part of the workflow (e.g., trigger the webhook, simulate file upload, etc.) to ensure that all modifications work as expected.
   * Use the n8n execution log for troubleshooting any issues.

**3. Reusing the Workflow**

1. **Modular Design:**
   * The workflow is structured into modules (document upload, processing, query, feedback). You can reuse individual modules by copying nodes into a new workflow.
2. **Export for Reuse:**
   * Once modifications are complete and the workflow is working, export the workflow as a JSON file for future reuse or versioning.
3. **Documentation & Comments:**
   * Use n8n’s “Sticky Note” nodes or comments within nodes to document changes and customizations.
   * Keep a record of parameter changes (e.g., API endpoints or folder IDs) in a separate documentation file.
4. **Scaling & Integration:**
   * If you need to integrate additional data sources or models, add new nodes in a modular way.
   * Reuse common nodes (e.g., feedback handling) by cloning and adjusting them in new workflows.