Adam The Quiz Maker

1. Project Overview (Adam-The Quiz Maker):

- **What is Adam?**

Adam is an AI-powered quiz bot built using LangFlow with Retrieval-Augmented Generation (RAG). It dynamically generates quizzes on any topic, adapts to users' learning progress, and provides instant feedback.

What Problem Does It Solve?

Many learners struggle with engaging study materials and personalized assessments. Adam solves this by:

-Generating **custom quizzes** tailored to user needs

-**Support multiple question types**: multiple-choice, short-answer, true/false

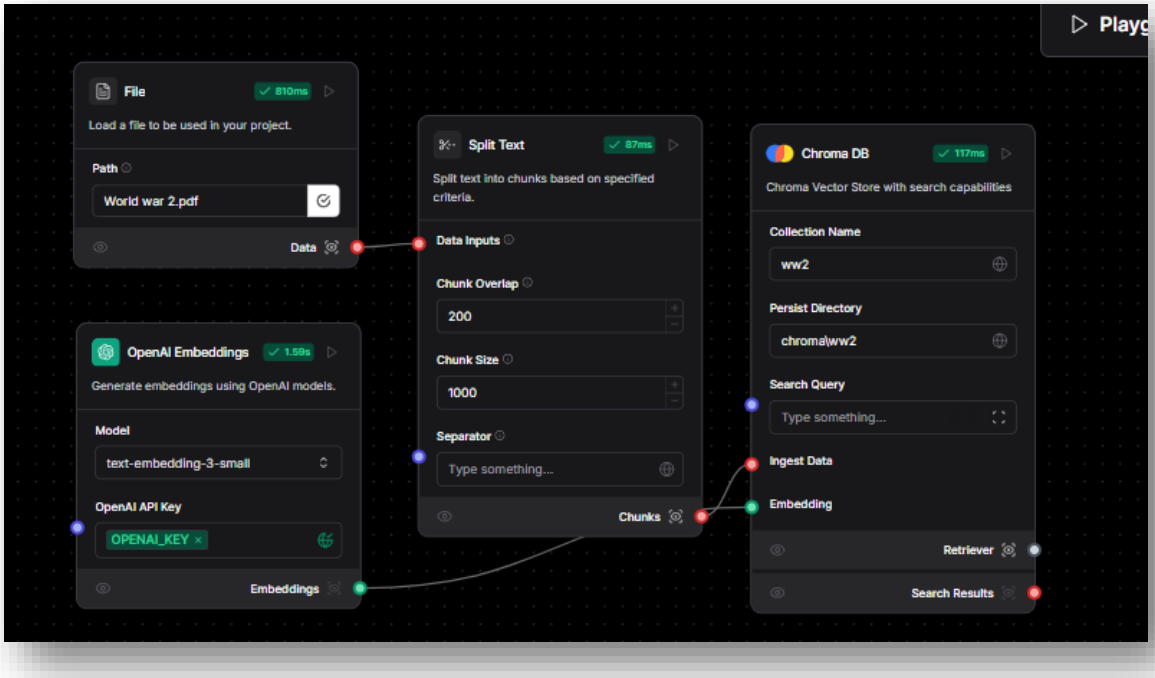
- **Retrieve relevant course materials** using a **vector database (ChromaDB)**

-Providing **real-time feedback** with explanations

- Adjusting **difficulty levels** based on past performance

2. LangFlow Components & Pipeline:

Data Loader Pipeline:



This Data Loader Pipeline is responsible for ingesting documents into Adam's Retrieval-Augmented Generation (RAG) system. It processes raw text from uploaded files, converts them into vector embeddings, and stores them in ChromaDB for efficient retrieval.

Pipeline Components & Workflow:

1. File Loader (Input Source)

- **Purpose:** Loads the document into the pipeline.

- **Node:** File
- **Current File:** Upload your file
- **Functionality:** Reads the document and passes raw text to the next step.

2. Text Splitting (Chunking the Data)

- **Purpose:** Breaks the document into smaller, manageable text chunks for processing.
- **Node:** Split Text
- **Configuration:**
  - **Chunk Size:** 1000 characters
  - **Chunk Overlap:** 200 characters (ensures context continuity)
- **Functionality:** Prevents loss of context in retrieval by maintaining overlapping text sections.

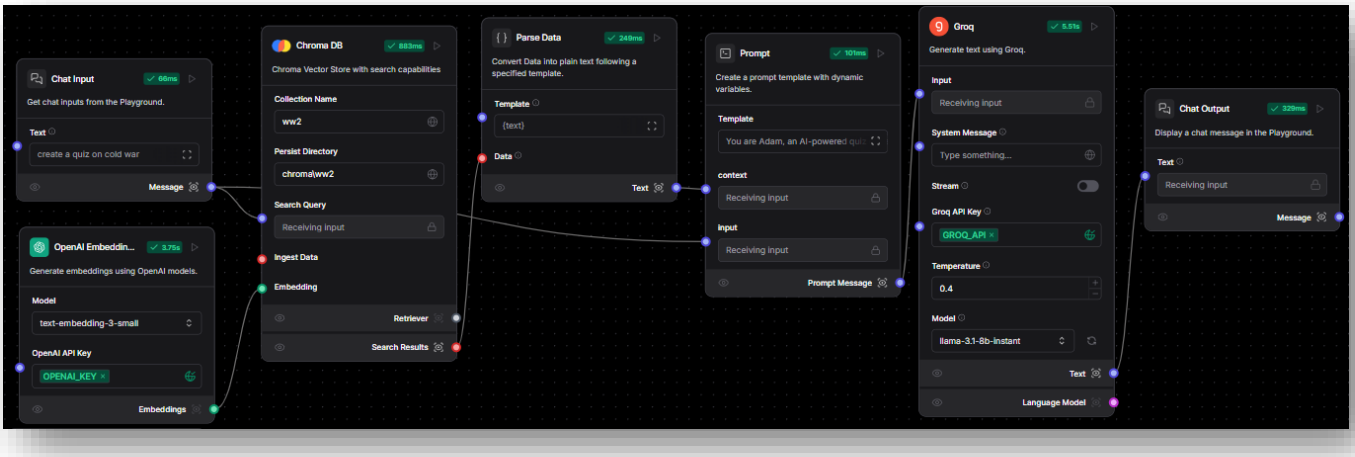
3. Embedding Generation

- **Purpose:** Converts text chunks into vector embeddings for efficient semantic search.
- **Node:** OpenAI Embeddings
- **Model Used:** text-embedding-3-small
- **Api Key:** Add your api key
- **Functionality:** Uses OpenAI's embedding model to transform text into a numerical format that can be stored and searched in the vector database.

4. Vector Database (ChromaDB Storage)

- **Purpose:** Stores embeddings for fast and accurate retrieval when generating quizzes.
- **Node:** Chroma DB
- **Configuration:**
  - **Collection Name:** Create a collection
  - **Persist Directory:** Create a directory (ensures stored embeddings are persistent)
- **Functionality:** Enables **quick retrieval of relevant content** when the user requests a quiz on **World War 2** or related topics.

<u>Retriever Pipeline</u>:



The Retriever Flow in this LangFlow setup is responsible for searching and retrieving relevant content from ChromaDB when a user requests a quiz on a specific topic. It ensures that the quiz is generated using accurate, context-aware information.

1. User Input (Chat Input)

- **Node:** Chat Input
- **Function:** Captures user input
- **Output:** Sends this input as a **search query** to ChromaDB.

2. ChromaDB Retrieval

- **Node:** Chroma DB
- **Function:** Searches the **vector database** for relevant information based on the user's query.
- **Configuration:**
  - **Collection Name:** Same name as data loader-ChromaDB
  - **Persist Directory:** Same directory (Stores embeddings for persistent retrieval.)
  - **Search Query:** Receives **user input** from the Chat Input node and searches for matching vector embeddings.
- **Output:** Returns the **most relevant text chunks** that match the query.

3. Parsing Retrieved Data

- **Node:** Parse Data
- **Function:** Converts the retrieved **text chunks** into a plain-text format for the quiz prompt.
- **Input:** {text} (retrieved content from ChromaDB)
- **Output:** Structured, readable text to be inserted into the prompt.

4. Prompt Template (Quiz Generation)

- **Node:** Prompt
- **Function:** Uses the retrieved **context** and **user input** to structure a quiz.
- Example Prompt:

> You are Adam, an AI-powered quiz generator designed to create engaging and well-structured quizzes based on course material. Your goal is to provide interactive quizzes that help users learn effectively.
>
> Instructions:
>
> - If the user asks about your introduction, role, or capabilities, explain what you do without generating a quiz.
> - If the user explicitly requests a quiz, analyze the input and generate one based on course content
> - Analyze the user's input to decide if course content is needed
> - Generate a mix of multiple-choice and short-answer questions.
> - Each multiple-choice question should have four answer options (a, b, c, d).
> - Ensure questions are accurate, relevant, and well-structured.
> - The correct answer for each multiple-choice question should be included separately at the end.
> - Keep the response conversational, friendly, and engaging so users feel like they're interacting with a tutor.
> - Do not generate a quiz unless explicitly asked.
>
> + User Input: {input}
>
> + Relevant Course Content: {context}

- **Template Variables:**
  - **Context:** Retrieved text from ChromaDB
  - **Input:** User's original request

5. LLM Processing (Quiz Generation)

- **Node:** Groq(FREE)

- **Function:** Uses the **retrieved context** and **prompt template** to generate quiz questions.
- **Model Used:** llama-3.1-8b-instant
- **Temperature:** 0.4 (Ensures balanced creativity and accuracy.)
- **Output:** Returns a structured quiz based on the user's request.

6. Displaying the Quiz

- **Node:** Chat Output
- **Function:** Displays the generated quiz as a chatbot response.