Retrieval-Augmented Generation (RAG) System for PDF Text and Tables: Project Explanation

1. INTRODUCTION:

In today's data-driven world, the ability to extract meaningful information from complex documents is vital. Many documents, especially PDFs, contain a mixture of unstructured text and structured tabular data. Traditional text-based search and retrieval methods often struggle to effectively combine insights from both these data types.

This project presents a Retrieval-Augmented Generation (RAG) system tailored to process PDFs containing both text and tables, enabling users to ask natural language queries and receive precise, context-aware answers. The system achieves this by combining semantic retrieval using embeddings stored in a vector database and answer generation using a Large Language Model (LLM).

2. PROBLEM STATEMENT:

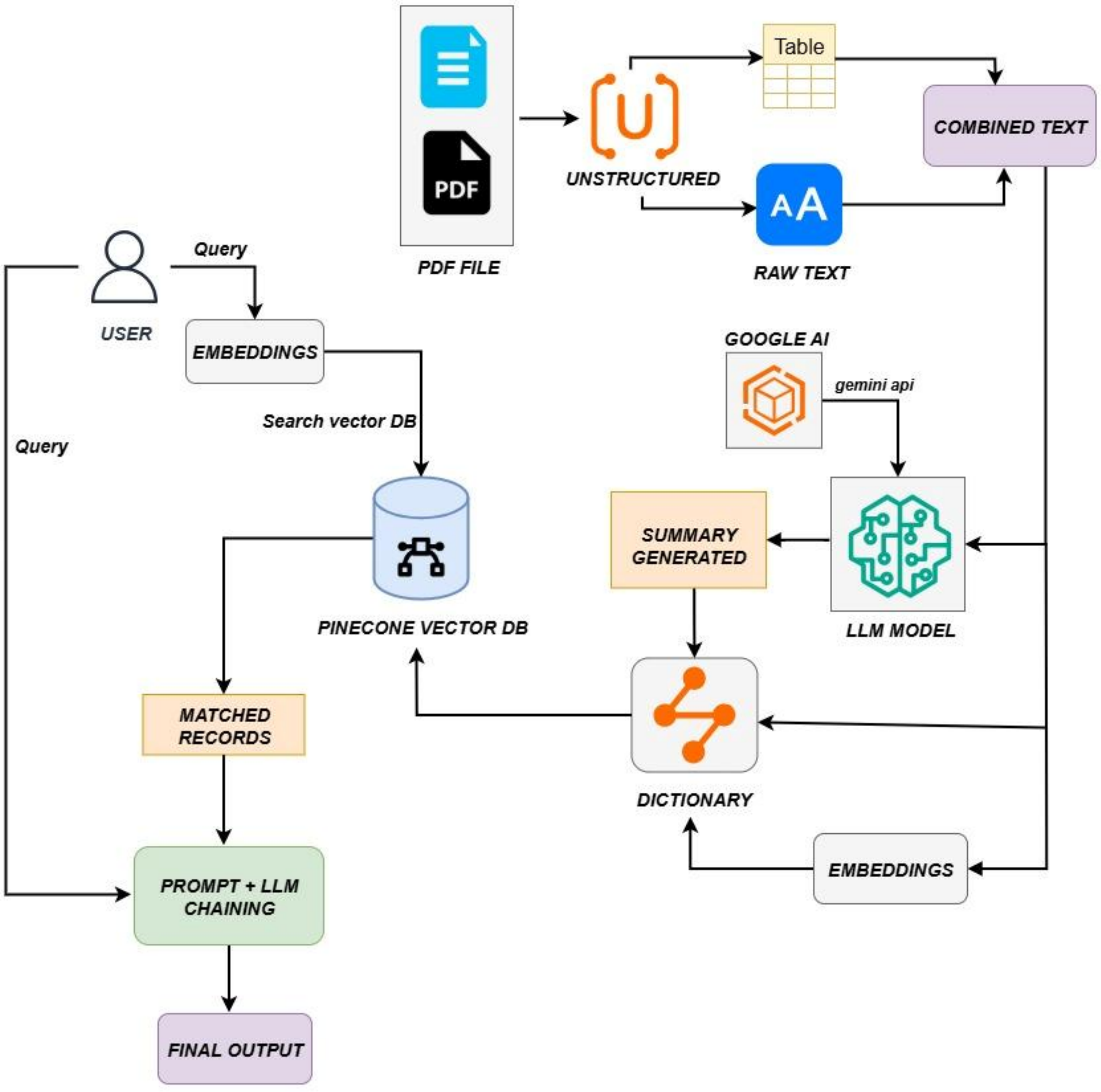
- PDFs often contain a blend of unstructured text and tabular data.
- Existing retrieval systems usually focus on either text or tables, rarely combining them efficiently.
- Users need a system that can understand and search across both content types simultaneously.
- The goal is to develop a pipeline that extracts, processes, and semantically indexes mixed content, enabling fast retrieval and accurate LLM-generated responses.

3. SYSTEM OVERVIEW & ARCHITECTURE:

The core of the system is a RAG pipeline designed to:

1. **Extract** text and tables from PDFs, combining them into unified, meaningful chunks.

2. **Embed** these chunks into numerical vectors that capture semantic information.
3. Store these embeddings, along with related metadata and summaries, in a **vector database** (Pinecone).
4. At query time, **embed the user query** and perform a similarity search to retrieve the most relevant chunks.
5. Use a **Large Language Model** (Google AI Gemini) to generate a natural language answer, leveraging the retrieved content for accuracy.



4. Methodology

i. **PDF Content Extraction**
The PDF is processed using the **unstructured** library to extract both unstructured text and tables. Tables are converted into readable text, and raw text is extracted continuously. Both are combined into unified text chunks that maintain the original document's context.

ii. **Embedding Generation**
Each combined text chunk is converted into a vector embedding. These embeddings capture the semantic meaning of the content in a high-dimensional space. Additionally, a summary is generated for each chunk to provide contextual information and improve retrieval relevance.

iii. **Dictionary Creation**
For every chunk, a dictionary is created containing:

- The raw combined text
- The embedding vector
- The summary
- Metadata such as source file name, page number, and chunk ID

This dictionary fully represents the semantic and contextual information of the chunk.

4. **Storage in Vector Database**
All dictionaries are stored in the Pinecone vector database. Pinecone indexes the embeddings for fast semantic similarity search and retrieves the most relevant dictionaries when queried.

5. **Query Handling and LLM Answer Generation**
User queries are embedded using the same model as the document chunks. Pinecone retrieves top matching chunks by embedding similarity. These chunks, along with their summaries, are passed to the Google AI Gemini LLM, which combines retrieved information with its pretrained knowledge to generate clear, accurate answers.