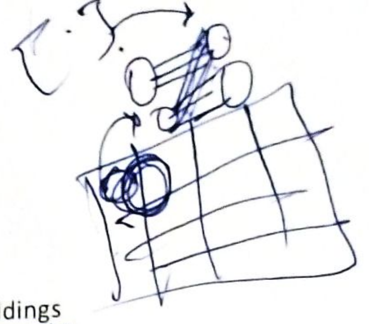

AI/ML PROJECT ABSTRACT

TITLE:

AI-Powered Document Query System Using RAG and Vector Embeddings

Algorithms Used:

The project leverages Retrieval-Augmented Generation (RAG) and vector embeddings using LangChain, OpenAI API, and ChromaDB. RAG is utilized to enable the integration of relevant chunks from the data source into a coherent response, while embeddings are generated for text similarity computation. Techniques like recursive character text splitting and cosine similarity are employed for efficient document segmentation and query matching.

Expected Output:

The system will enable AI-driven interaction with PDF documents. Users can query the data, and the AI will provide accurate, context-specific responses derived from the document content while quoting the original sources. This ensures transparency and minimizes hallucinated responses. The application aims to serve use cases like customer support, document summarization, and knowledge retrieval.

Dataset (Authenticated):

The dataset consists of PDF files or collections of text/markdown files loaded into the system. These documents are segmented, embedded, and stored in a ChromaDB for efficient querying. The demonstrated example in the development process includes AWS Lambda documentation, ensuring the dataset's reliability and relevance.

*PDF, doc. E

TEAM MEMBERS:

2320030312

2320030365

2320030243

(Abstract).

AI-POWERED DOCUMENT QUERY SYSTEM USING RAG AND VECTOR EMBEDDINGS

KEY THINGS ABOUT OUR PROJECT

Algorithms Used:

- Retrieval-Augmented Generation (RAG): Integrates relevant document chunks into coherent responses.
- Vector Embeddings: Enables text similarity computation.
- Techniques Employed:
- Recursive character text splitting for document segmentation.
- Cosine similarity for accurate query matching.
- Tools and Libraries:
- LangChain
- OpenAI API
- ChromaDB

3. How the Project Works
   1. Data Loading:
      - Authenticated PDF or markdown files are uploaded.
   2. Preprocessing:
      - Documents are segmented using recursive character text splitting.
      - Vector embeddings are generated for the segmented chunks.
   3. Database Creation:
      - Segmented and embedded data is stored in ChromaDB for efficient retrieval.

4. Team Members

- 2320030312
- 2320030365
- 2320030243

Logo