**Project Documentation: Athena AI Chat Development**

**1. Overview**

This project focuses on developing a chatbot system using LangChain and Pinecone to create an efficient and scalable AI chat solution.

* **LangChain** is a framework designed to simplify the integration of large language models (LLMs) into applications, including use cases like document analysis, summarization, chatbots, and code analysis.
* **Pinecone** is a high-performance, scalable vector database optimized for applications such as:
  + Natural language processing (NLP)
  + Computer vision
  + Machine learning (ML)

**2. Data**

**Data Source**

* **Datasets**: Two PDF files sourced from the internet.
* **Location**: Stored in the data folder.

**Data Processing**

* **Embeddings**: Generated using Pinecone with the following parameters:
  + chunk\_size=1000
  + chunk\_overlap=200

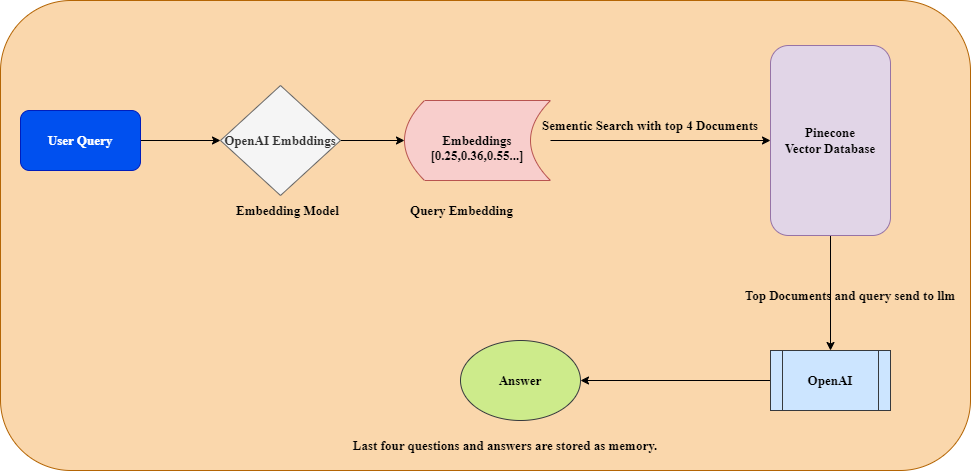
**Memory:**

Last four questions and answers are stored as memory.

**Solution Diagram:**

**Architecture**:

The Architecture is shown in the figure:



* **User Input**: Receives and processes user queries.
* **Query Embedding**: Converts the query into an embedding representation.
* **Semantic Search**: Leverages Pinecone embeddings to search and retrieve relevant information.
* **Response Generation**: Generates responses based on the retrieved results and returns them to the user.

**Multiple Issues:**

1. **Missing Medical Information**:
   * Some medical information is missing in the embeddings. In such cases, we can use external agents to retrieve real-time data from the internet when the model returns an "I don’t know" response.
2. **Document Splitting**:
   * There is a tradeoff when splitting documents for retrieval:
     1. **Small Chunks**: Embeddings for smaller chunks are more accurate as they retain specific meaning.
     2. **Context Retention**: Larger chunks retain better context but may lose precision in embeddings.
   * The **ParentDocumentRetriever** balances these conflicting needs by splitting documents into small chunks while retaining as much context as possible.
3. **Multi-Query Approach**:
   * **Step 1**: User submits a query, which may be incomplete.
   * **Step 2**: The multi-query approach rewrites the query multiple times.
   * **Step 3**: These rewritten queries are sent to OpenAI to improve search accuracy.
   * **Step 4**: Pinecone retrieves the relevant document chunks based on these rewritten queries.

**3. Solution Path**

**Base Model**

* **File**: chatbot\_implementation.ipynb
* **Description**: This notebook provides a basic implementation of the chatbot using LangChain and Pinecone to handle user queries and perform semantic searches.

**Multi-Query Approach**

* **File**: multi\_query\_chatbot\_implementation.ipynb
* **Description**: This advanced implementation improves search accuracy for incomplete queries by generating and rewriting multiple versions of the query before sending them to OpenAI.

**Model Finetune**

* **Status**: Not required for OpenAI models using Rag but can be used when using open-source models.

**4. Implementation**

**Code Organization**

* **Embedding Upsertion**:
  + **File**: data\_upsert\_pinecone.ipynb
  + **Purpose**: This notebook handles the upsertion (uploading and updating) of embeddings into Pinecone.
* **Base Model**:
  + **File**: chatbot\_implementation.ipynb
  + **Purpose**: Provides a simple chatbot framework using Pinecone for semantic search.
* **Multi-Query Model**:
  + **File**: multi\_query\_chatbot\_implementation.ipynb
  + **Purpose**: Improves search accuracy through multiple query rewrites and semantic matching.

**5. Dependencies**

* **Packages**: Listed in the requirements.txt file, which includes all necessary Python libraries for running the project.