**Detailed Report: QA System with RAG Approach**

**Objective**

The aim of this project is to build a question-answering (QA) system using a Retrieval-Augmented Generation (RAG) approach. The system leverages preprocessed data (sports) from the 20 Newsgroups dataset and GPT models to answer user queries based on relevant document chunks. This system also integrates a user interface (UI) using Chainlit for interaction.

**Architecture Overview**

The solution is divided into several components:

1. **Data Loading and Preprocessing:** This handles the loading and cleaning of text data from the 20 Newsgroups dataset.
2. **Embedding Generation:** We use a SentenceTransformer model to generate embeddings for the question and document chunks, enabling efficient retrieval based on semantic similarity.
3. **Document Retrieval:** We utilize a similarity-based approach to retrieve relevant document chunks using dot product-based similarity.
4. **Answer Generation:** Using OpenAI’s GPT model, answers are generated based on the context from retrieved document chunks or, in cases where no relevant data is found, directly from the model itself.
5. **Chainlit UI Integration:** This provides an interactive user interface for question input and answer display.
6. **Vector Store for Future Expansion:** The current solution uses in-memory retrieval for simplicity, but it can be enhanced using a more scalable vector store like FAISS or Pinecone for efficient large-scale document retrieval.

**Solution Breakdown**

1. **Data Loading and Preprocessing**

Load raw data from two newsgroups: ***rec.sport.baseball and rec.sport.hockey.***

Preprocessing removes unwanted metadata such as email headers and extra whitespace using regular expressions.

**2. Embedding Generation**

Used the SentenceTransformer model (all-MiniLM-L6-v2) to generate document and question embeddings.

These embeddings are used to compute the similarity between the user query and document chunks using dot product.  
  
**3. Document Retrieval**

The top n most relevant documents are retrieved based on similarity scores.

We apply a threshold to ensure that only highly relevant documents are considered.  
  
**4. Answer Generation**

If relevant documents are found, they are used as context for generating the answer using OpenAI's GPT model.

If no relevant documents are found, the system falls back to generating an answer directly from GPT.  
  
**5. Chainlit UI Integration**

The system is integrated with Chainlit to provide a user-friendly interface.

Users can input their questions and receive answers, with the option to display the source of the answer (whether it came from the retrieved documents or OpenAI).  
  
**Optimizations and Enhancements**

**Vector Store Integration:** We could replace the in-memory retrieval process with a more scalable solution like FAISS or Pinecone for efficient handling of larger datasets. This allows faster and more efficient document retrieval.

**Model Fine-tuning:** While the current implementation uses OpenAI’s GPT model for generating answers, further improvements can be made by fine-tuning a custom model or using domain-specific models.

**Data Expansion:** The system currently handles sports-related documents from the 20 Newsgroups dataset. In future iterations, we can expand the system to include a broader range of topics (e.g., technology and healthcare) and use Wikipedia API for additional data sources.

**How does Chainlit App looks :**  
  
