**Project Report: RAG Q&A Chatbot Server**

**Introduction**

This project implements a **Retrieval-Augmented Generation (RAG) Q&A Chatbot Server** to provide accurate answers based on a knowledge base. It integrates **FastAPI**, **Streamlit**, **LangChain**, **FAISS**, and **Groq’s LLaMA 3.3-70B model** to create a scalable and efficient chatbot. The project also supports analytics generation and document-based question answering.

**Objectives**

* Develop a chatbot that retrieves and generates responses from structured documents.
* Use vector embeddings for efficient search and retrieval.
* Provide API endpoints for question-answering and analytics.
* Ensure scalability and flexibility in handling various data sources.

**Steps Taken**

**1. Data Preprocessing**

**Why?** To ensure the input data is clean, structured, and optimized for retrieval.

**Actions:**

* Loaded documents from a specified directory.
* Removed null values from children, country and agents by Imputation.
* Removed all duplicated rows.
* Propper formatted the columns for statistical analysis by converting float to int and str to datetime.
* Saved the clean file with cleaned\_hotel\_booking.csv
* Processed text using **RecursiveCharacterTextSplitter** to create meaningful chunks with chunk\_size 1000 and overlapping of 200.
* Embedded text using **OpenAI embeddings**. Because it is fast and gives good accuracy.
* Used pyPDFDirectoryLoader for loading documents to train the model.
* Indexed the embeddings using **FAISS** for efficient search.

**2. Data Analysis**

Why? To extract meaningful insights from the data and ensure an optimal chatbot response mechanism.

Actions:

* Performed exploratory data analysis (EDA) to understand data distribution.
* Generated visualizations for data trends and patterns.
* Analyzed word frequency and text patterns in documents to optimize retrieval.
* Evaluated retrieval accuracy using test queries.
* Assessed vector space performance to ensure proper document indexing.

**3. Backend Development**

**Why?** To create a structured API for interaction with the chatbot.

**Actions:**

* Implemented a FastAPI-based backend because it is fast.
* Created API endpoints for question answering (/ask) and analytics (/analytics).
* Used **Pydantic** models to validate request inputs.
* Set up **environment variables** for API key management.

**4. Chatbot Development**

**Why?** To enable intelligent, context-based responses.

**Actions:**

* Configured **Groq’s LLaMA 3.3-70B** for natural language processing. It is free and an open source model.
* Created a **retrieval chain** with FAISS-based document search.
* Designed a **ChatPromptTemplate** to guide LLM responses.
* Integrated response generation with FastAPI routes.

**5. Deployment and API Testing**

**Why?** To ensure seamless interaction and usability.

**Actions:**

* Used **Uvicorn** to serve the FastAPI backend.
* Tested API calls using **Postman** and curl commands.
* Validated chatbot responses against sample queries.

**Challenges & Solutions**

|  |  |
| --- | --- |
| **Challenge** | **Solution** |
| Designing a proper approach  Cleaning the data  Data Analysis  Managing large text documents efficiently  Analysis of customer geography  In a map  Creating API | Breaking the complete project in modules then each module in steps.  With such large data you can not be sure there are not some random values or outliers. Implement cleaning column by column.  Calculated all the statistics. Used online resources for formulas.  Implemented **text chunking** and **FAISS-based indexing**  Used plotly and saved the map in .html file. That could be open in browser terminal.  Searched online, read documents and saw some tutorials. |
| Ensuring accurate responses from RAG | Used **OpenAI embeddings** and **optimized retrieval chains** |
| API latency due to large model size | Leveraged **vector search** to minimize unnecessary model calls |
| Secure API key management | Used **.env files** and **environment variables** |

**Conclusion**

The project successfully built a **scalable RAG-based chatbot** that retrieves information effectively using **FAISS and OpenAI embeddings**. Future improvements include **better UI integration, cloud deployment, and enhanced analytics features** to make the system more robust and user-friendly.