Project Report: Quiz Bot with Retrieval Augmented Generation

1. Introduction

The objective of this project is to create an intelligent quiz bot capable of generating and validating quiz questions based on documents stored in PDFs. The system leverages advanced technologies such as **Retrieval Augmented Generation (RAG)**, **Faiss vector storage**, and **GPT-based models** to enhance the quiz generation process. The bot uses a **Streamlit-based user interface** for interactive quizzes, enabling users to select from different types of questions, including Multiple Choice Questions (MCQ), True/False questions, and Open-Ended questions.

The main goal is to provide an interactive experience for users, where they can test their knowledge based on dynamically generated questions from a large corpus of documents.

2. System Workflow

2.1 User Interface (UI)

The front-end interface is built using **Streamlit**, a powerful library for creating interactive applications with minimal code. The user interface includes the following features:

- **Sidebar for options**: Users can select the question type (MCQ, True/False, or Open-Ended).
- **Dynamic question generation**: Upon clicking the "Generate Quiz" button, a random question is generated based on the type selected by the user.
- **Answer submission**: After reading the question, users can submit their answer. The system validates the answer and provides feedback.

2.2 Back-End Logic

The back-end consists of several modules that handle the task of generating quiz questions, validating answers, and managing metadata. These components include:

1. **Ingestion and Data Processing**:
    - The system first loads and processes PDF documents from a designated folder. Each PDF is converted into clean text using the **PyMuPDF (fitz)** library.
    - The text is split into chunks to make it manageable for embedding. Each chunk is then transformed into a vector representation using **Sentence Transformers**.
    - The embeddings are stored in **Faiss**, a library optimized for efficient similarity search, which allows us to store and retrieve document chunks based on their vector similarity.
2. **Quiz Question Generation**:
    - The question generation logic is handled by **GPT-based models**. These models create MCQs, True/False, and Open-Ended questions from the document chunks.

    - The **RAG** (Retrieval Augmented Generation) approach is used: the system retrieves relevant text (document chunk) based on user input or random selection and then generates a quiz question from it.
3. **Answer Validation**:
    - Once the user submits an answer, the system validates it using the same GPT-based model. The model is tasked with comparing the user's answer to the correct answer and providing feedback on the correctness of the response.

2.3 Libraries and Technologies Used

Several libraries and technologies were carefully chosen to implement the various aspects of this project. Below is a discussion on why each library was selected.

2.3.1 Faiss (Facebook AI Similarity Search)

Faiss is a library developed by Facebook for efficient similarity search and clustering of dense vectors. Given that we are dealing with embeddings of document chunks, **Faiss** is ideal for storing and querying large numbers of high-dimensional vectors. The library allows for fast retrieval of relevant document chunks based on similarity, enabling the **RAG** approach to function effectively.

- **Why Faiss?**
    - **High Efficiency**: Faiss is optimized for speed and memory usage when working with large datasets, which is critical for this project where many document chunks need to be processed and queried quickly.
    - **Scalability**: Faiss supports both CPU and GPU acceleration, making it suitable for large-scale implementations.

2.3.2 Sentence Transformers

Sentence Transformers provides pre-trained models for converting sentences or paragraphs into fixed-size vector representations (embeddings). In our system, these embeddings are used to represent document chunks and retrieve relevant information based on semantic similarity.

- **Why Sentence Transformers?**
    - **Pre-trained Models**: These models are fine-tuned for various NLP tasks and are capable of generating high-quality sentence embeddings out-of-the-box.
    - **Versatility**: It supports multiple languages and works well with longer documents, which is important as our input data comes from PDF documents that could contain detailed information.

2.3.3 GPT4All

GPT4All is a GPT-based model that serves as the core engine for question generation and answer validation. By feeding the relevant text chunks into the model, it generates quiz

questions in different formats (MCQ, True/False, Open-Ended). The model also validates the user's responses by comparing the provided answers to the expected ones.

- **Why GPT4All?**
    - **Flexibility**: It allows for the generation of different types of questions (MCQ, True/False, and Open-Ended) by simply modifying the prompt.
    - **Contextual Understanding**: GPT models are known for their deep understanding of language and context, making them ideal for generating meaningful and coherent questions and validating answers.

2.3.4 Streamlit

Streamlit is used for building the front-end of the application. It's a simple and fast way to create interactive web applications with minimal setup.

- **Why Streamlit?**
    - **Quick Prototyping**: Streamlit allows for rapid development of the UI, which is ideal for our project where the focus is more on the back-end logic.
    - **Real-time Updates**: The ability to interact with the user and update the interface dynamically based on user input makes Streamlit an excellent choice for our quiz application.

2.3.5 PyMuPDF (fitz)

PyMuPDF is used to extract and process text from PDF files. It is an efficient tool for handling PDF documents and is capable of extracting text while preserving the structure of the document.

- **Why PyMuPDF?**
    - **Efficiency**: PyMuPDF is fast and accurate in extracting text from PDF files.
    - **Versatility**: It supports a variety of formats, including text extraction from complex PDFs with mixed content.

3. Challenges and Solutions

During the development of the project, several challenges were encountered and addressed:

- **Document Chunking**: Handling large documents and splitting them into manageable chunks without losing important context was a challenge. By using a simple word-based chunking strategy, we ensured that each chunk was small enough to process efficiently but still meaningful.
- **Question Generation Consistency**: Ensuring that the questions generated by GPT models were relevant and of high quality required fine-tuning the prompts. By refining the prompts and providing clear instructions, we achieved the desired results.

- **Performance Optimization**: With large amounts of data, performance became a concern. By using Faiss for efficient vector storage and retrieval, we were able to significantly improve the search and retrieval time.

4. Conclusion

This project demonstrates the successful implementation of an intelligent quiz bot using **Retrieval Augmented Generation** and **Faiss vector storage**. The system allows for the generation of quiz questions based on a variety of document sources, providing users with an interactive and dynamic experience. The choice of libraries such as **Faiss**, **Sentence Transformers**, and **GPT4All** has enabled the efficient processing, storage, and generation of high-quality content, ensuring that the system meets the goals of speed, accuracy, and scalability.

The final application offers an engaging platform for users to test their knowledge on a range of topics, and the technology stack used ensures that it is both efficient and scalable for future enhancements.