RAG-based Generative Search System for Insurance Policy Documents

1. Problem Statement

The insurance industry deals with large volumes of policy documents containing extensive details.
Retrieving specific information from these documents is often time-consuming and requires manual efforts.
Customers, agents, and businesses face challenges in efficiently accessing the required policy information.
To address this, we have developed a Retrieval-Augmented Generation (RAG) system that enables users
to ask natural language questions and receive accurate, context-aware responses generated from policy documents.
This project integrates generative AI models with a document retrieval mechanism to enhance the search process.

2. Approach

The project follows a Retrieval-Augmented Generation (RAG) approach, combining document retrieval and
LLM-powered text generation. The system first retrieves relevant sections from policy documents based
on the user query and then uses a language model to generate accurate responses.

3. Methodology

The project consists of the following major steps:

Step 1: Document Ingestion

Insurance policy documents (PDFs, DOCX, and TXT files) are loaded into the system using document parsers like PyPDF and docx2txt. The documents are then preprocessed for tokenization and cleaning.

Step 2: Indexing and Storage

The processed documents are indexed using LlamaIndex (or LangChain). This enables efficient

retrieval of relevant text based on user queries.

Step 3: Query Processing

Users submit their queries in natural language. The system retrieves the most relevant document sections using a retriever mechanism before passing the information to the language model.

Step 4: Response Generation

A pre-trained Large Language Model (LLM), such as OpenAI's GPT, generates a final answer based on the retrieved content. This ensures accuracy and relevance while avoiding hallucinations.

4. Technologies Used

- **Python**: Used for developing and implementing the system.
- **LlamaIndex/LangChain**: Handles document retrieval and indexing.
- **OpenAI GPT API**: Used for text generation and answering user queries.
- **PyPDF, docx2txt**: Document loaders for parsing policy documents.
- **Jupyter Notebook**: Development and testing environment.

5. Implementation Details

The implementation follows a modular approach, ensuring flexibility and scalability.
The major components include document loaders, retriever mechanisms, LLM integration, and a user-friendly interface for query submission.

6. Conclusion

This project demonstrates the power of Generative AI combined with information retrieval in
solving real-world problems in the insurance domain. By leveraging RAG, policyholders, agents, and businesses can access information quickly and accurately without manually scanning
through documents. This system significantly enhances user experience and operational efficiency.