## Project Documentation: Intelligent Insurance Q&A System

### 1. Project Goals and Objectives

The primary goal of this project was to design, implement, and evaluate a multi-layered intelligent search system capable of answering complex user queries about a detailed insurance policy document. The system is built on the Retrieval-Augmented Generation (RAG) architecture, which leverages the strengths of neural information retrieval and large language models (LLMs).

The key objectives were:

* **Automated Information Extraction:** To develop a robust pipeline that can automatically process a semi-structured PDF document, accurately extracting all text and tabular data while preserving the original context.
* **High-Relevance Semantic Search:** To implement a search layer that goes beyond simple keyword matching, using semantic embeddings to understand the user's intent and retrieve the most contextually relevant sections of the policy document.
* **Performance and Accuracy Enhancement:** To improve search accuracy through a re-ranking mechanism and boost system performance for common queries by implementing a cache.
* **Natural Language Generation:** To utilize a Large Language Model (LLM) to synthesize the retrieved information into a clear, accurate, and easy-to-understand answer for the end-user.
* **Trust and Transparency:** To ensure the system's responses are trustworthy by providing citations that link back to the specific page and policy document from which the information was sourced.

### 2. Data Sources

The system was built and tested using a single, representative data source:

* **Document:** Principal-Sample-Life-Insurance-Policy.pdf
* **Description:** This is a 64-page sample Group Life Insurance policy document from Principal Life Insurance Company.
* **Characteristics:** The document is a typical example of complex legal and financial literature. It contains a mix of unstructured paragraphs, a structured table of contents, legal definitions, benefit schedules, and embedded tables. The presence of intentionally blank pages and varied formatting presents a realistic challenge for data extraction.

### 3. Design Choices and System Architecture

The system is designed with a three-layer architecture: an Embedding Layer for data processing, a Search Layer for information retrieval, and a Generation Layer for synthesizing the final answer.

#### Layer 1: The Embedding Layer

This layer is responsible for converting the raw PDF into a searchable vector database.

* **Text Extraction Tool: pdfplumber**
  + **Rationale:** pdfplumber was chosen for its advanced capabilities in parsing PDF files. Unlike simpler libraries, it provides access to the exact coordinates of every character and object on a page. This was crucial for implementing a custom extraction logic that could differentiate between standard text paragraphs and structured tables, extracting both while maintaining their original order on the page.
* **Chunking Strategy: Page-Level Chunking**
  + **Rationale:** Each page of the document was treated as a single chunk. This strategy was selected for its simplicity and effectiveness in preserving the context of self-contained sections, which is common in policy documents. To clean the data, a filter was applied to remove any pages with fewer than 10 words, effectively eliminating blank pages and noise.
* **Embedding Model: all-MiniLM-L6-v2**
  + **Rationale:** This model from the SentenceTransformers library was chosen as a high-performing, open-source solution for generating semantic embeddings. It is optimized for semantic similarity tasks, making it ideal for understanding the relationship between the user's query and the document chunks. This choice provides a strong balance of performance and cost-effectiveness, as it can be run locally without relying on paid APIs.
* **Vector Database: ChromaDB**
  + **Rationale:** ChromaDB was selected as the vector store for its ease of use and efficient in-memory and persistent storage options, making it perfect for rapid development and testing. It allows for the storage of documents, their vector embeddings, and associated metadata (e.g., policy name and page number), which is essential for providing citations in the final output.

#### Layer 2: The Search Layer

This layer retrieves the most relevant information from the vector database in response to a user's query.

* **Retrieval and Re-ranking Strategy:**
  1. **Initial Retrieval (Semantic Search):** The user's query is embedded, and ChromaDB is used to retrieve the top 10 most semantically similar chunks (pages) based on cosine similarity. This acts as a fast and efficient first-pass filter.
  2. **Re-ranking (Cross-Encoder):** The top 10 retrieved chunks are then re-ranked using a Cross-Encoder model (cross-encoder/ms-marco-MiniLM-L-6-v2). The Cross-Encoder takes the query and each chunk as a pair, providing a more accurate relevance score.
  + **Rationale:** This two-stage process combines the speed of semantic search with the accuracy of a Cross-Encoder. The initial retrieval quickly narrows the search space, and the re-ranker then precisely orders this smaller set of candidates, ensuring the most relevant information is passed to the final generation layer.
* **Caching Mechanism:**
  + **Rationale:** To improve response time and reduce redundant computations for frequently asked questions, a caching layer was implemented using a separate ChromaDB collection. When a query is received, the cache is checked first. If a semantically similar query (within a distance threshold of 0.2) is found, its pre-computed results are returned instantly. If not, a full search and re-ranking are performed, and the new query and its results are stored in the cache for future use.

#### Layer 3: The Generation Layer

This layer takes the curated information from the search layer and formulates a human-readable answer.

* **Language Model: distilbert/distilgpt2**
  + **Rationale:** The implementation uses a transformers pipeline with distilgpt2, demonstrating the viability of an end-to-end system using open-source models. While smaller than state-of-the-art proprietary models, this choice confirms the architectural soundness. For a production environment, this model could be easily swapped with a more powerful API-based model like GPT-4 or an open-source model like Llama 3 to enhance the quality and fluency of the generated text.
* **Prompt Engineering:**
  + **Rationale:** A carefully designed prompt is used to control the LLM's output. The prompt grounds the model by instructing it to act as an expert insurance assistant and to base its answer *only* on the provided context (the top 3 re-ranked documents). It includes explicit instructions to cite sources using the metadata and to format the response clearly, which is critical for preventing the model from generating fabricated information (hallucinations) and ensuring the answer is both accurate and trustworthy.

### 4. Challenges Faced and Future Improvements

* **Challenge: Table Extraction and Interpretation**
  + **Problem:** Extracting tabular data and preserving its structure as readable text was a key challenge.
  + **Solution:** Using pdfplumber to identify table boundaries and extract them separately before re-integrating them into the page's text flow was an effective solution.
* **Challenge: Ensuring Retrieval Accuracy**
  + **Problem:** Initial semantic search sometimes returned pages that were thematically related but did not contain the specific answer to the query.
  + **Solution:** The implementation of a Cross-Encoder for re-ranking proved crucial. It added a layer of deep relevance checking that significantly improved the quality of the documents sent to the LLM.
* **Future Improvement: Advanced Chunking Strategies**
  + While page-level chunking worked well, a more advanced strategy, such as recursive character splitting with semantic overlap, could be explored. This would create smaller, more focused chunks, potentially improving retrieval for highly specific queries whose answers are contained within a single paragraph.
* **Future Improvement: Upgrading the Generation Model**
  + The most significant improvement to the system's output quality would come from replacing the distilgpt2 model with a more powerful, state-of-the-art LLM. This would enhance the fluency, coherence, and summarization capabilities of the final answer.