Project Report: HELP MATE AI – Insurance Document Question Answering Assistant

# 1. Objectives

The primary goal of HELP MATE AI is to develop an intelligent question-answering system that can parse and understand insurance policy PDF documents, and provide contextual, human-like responses to user queries. Key objectives include:

* Automating extraction of relevant policy details from large PDFs.
* Leveraging LLMs and embeddings for semantic understanding.
* Building a fast, persistent vector database for efficient retrieval.
* Improving accuracy using cross-encoder reranking.
* Reducing API cost and latency with a caching mechanism.
* Delivering human-readable answers with citations.

# 2. Design

The system is designed as a retrieval-augmented generation (RAG) pipeline, combining:

* PDF Parsing & Preprocessing → Extract structured text and tables.
* Vector Database (ChromaDB) → Store document embeddings.
* Semantic Search → Retrieve top-k relevant chunks.
* CrossEncoder Reranking → Refine retrieved results for precision.
* LLM Response Generator → Generate natural, cited answers.
* Cache Layer → Avoid repeated searches for frequent queries.

Architecture Flow:

1. User enters query.
2. Query checked against cache.
3. If not cached → Semantic retrieval from ChromaDB.
4. Results reranked with CrossEncoder.
5. Top 3 passages passed to LLM.
6. LLM generates contextual, formatted answer with citations.

# 3. Implementation

## a. Environment Setup

* Libraries: pdfplumber, openai, chromadb, tiktoken, pandas, sentence-transformers.
* OpenAI Embeddings: text-embedding-ada-002.
* CrossEncoder: ms-marco-MiniLM-L-6-v2.
* LLM: GPT-3.5-Turbo.

## b. PDF Parsing

* Extract page-wise content (headings, text, tables).
* Store in DataFrame: [Page No., Heading, Text, Metadata].

## c. Vector Store (ChromaDB)

* Collection InsurancePolicyDoc: All embeddings.
* Collection Insurance\_Cache: Query cache.
* Persistent storage ensures scalability.

## d. Semantic Search

* Retrieve top 10 embeddings by cosine similarity.
* Apply threshold filtering (0.2).

## e. Reranking

* Use CrossEncoder to score [query, document] pairs.
* Store reranked scores in DataFrame.
* Select top 3 snippets.

## f. Response Generation

* Input: Query + top 3 snippets.
* Output: Human-readable, LLM-generated response with citations.

## g. Caching

* Store query-result pairs in Insurance\_Cache.
* Flattened metadata (JSON).

## h. Modular Functions

* get\_context() – Cache check + retrieval.
* rerank() – CrossEncoder scoring.
* top\_3\_context() – Extract top 3.
* get\_reply() – Full response generator.

# 4. Challenges

* PDF Complexity: Handling tables and inconsistent formatting in insurance documents.
* Embedding Cost: Large documents increased OpenAI embedding API usage.
* Reranking Latency: CrossEncoder improved accuracy but added extra inference time.
* Cache Design: Flattening JSON metadata for ChromaDB required schema tuning.
* LLM Hallucinations: GPT sometimes generated unsupported answers without citations.

# 5. Lessons Learned

* Hybrid Search (Embeddings + Reranking) significantly improves answer quality over embeddings alone.
* Caching is crucial for performance and cost efficiency.
* Structured Preprocessing of PDFs (headings + tables) makes retrieval more reliable.
* Metadata Preservation (page numbers, sections) ensures trustworthiness via citations.
* Modular Pipeline Design simplifies debugging, upgrades, and future extensibility.

# Conclusion

HELP MATE AI demonstrates an effective retrieval + generation pipeline for interpreting insurance documents. By combining semantic embeddings, ChromaDB storage, CrossEncoder reranking, and GPT-based response generation, the system delivers accurate, context-aware answers with source citations.  
  
This approach can be extended beyond insurance to any domain requiring PDF-based semantic search, such as legal documents, compliance manuals, or research papers.