# Semantic spotter project

## Option 1: Build a RAG System

“If you prefer a structured project with a clear starting point, you can build a project in the insurance domain, like the project you saw in the ‘Retrieval Augmented Generation’ session. The goal of the project will be to build a robust generative search system capable of effectively and accurately answering questions from various policy documents. You may use LangChain or LlamaIndex to build the generative search application.”

## Problem Statement

A sample life insurance policy was shared with us during the HelpMate AI project. Our goal here is to build a simple RAG application on the policy statement pdf document.

Solution Strategy - Build a POC/MVP which should solve the following requirements.

Users would get responses from the policy document. If they want to refer to the original page from which the bot is responding, the bot should provide a citation as well.

Goal - Solving the above two requirements well in the POC/MVP would ensure that the accuracy of the overall model is good and therefore further improvisations and customizations make sense.

Data Used - Principal Sample Life Insurance documents stored in a single folder

Tools used - LlamaIndex (only for now) has been used due to its powerful query engine, fast data processing using data loaders and directory readers as well as easier and faster implementation using fewer lines of code. Previously we were creating a generation layer as well however LlamaIndex takes care of all the necessary details and does the same job with much less lines of code.

I finally chose to use LlamaIndex instead of Langchain for this project because of the following reason:

| **Features** | **LlamaIndex** | **Langchain** |
| --- | --- | --- |
| Semantic Understanding | Advanced semantic understanding enables precise query results from insurance documents. | Semantic understanding capabilities might be limited, affecting query accuracy. |
| Efficient Retrieval | VectorStoreIndex facilitates efficient retrieval, ensuring quick access to relevant information. | Retrieval methods may be less optimized, leading to slower query performance. |
| User-Friendly Interface | Provides a user-friendly interface, simplifying the query process and enhancing user experience. | Interface might be less intuitive, potentially impacting usability and adoption. |

# System Design

In this section we proceed with the traditional system design for an RAG to actually build this POC/MVP version of the application, followed by a later stage fine tuning the semantic search layer by redefining the chunk size, context window etc. to arrive at a better version of the response to a query.

A diagram of a company

Description automatically generated

<https://colab.research.google.com/drive/1BG_41AG60gYE-F9pFCyHj7dZJHJ3KSk9?usp=sharing>