Insurance : Chatbot For Customer Support

The Insurance Chatbot for Customer Support is designed to assist customers with insurance-related queries efficiently and effectively. Leveraging advanced AI technologies such as Gemini-pro and Lang Chain to understand user messages, provide contextually relevant responses.

Technologies Used :-

- ✓ Streamlit
- ✓ LangChain
- ✓ PyPDF2
- ✓ PyTesseract
- ✓ FAISS

Features:-

- ✓ **Uploading of PDF Documents**:
  Through an intuitive interface, customers can upload multiple PDF documents directly to the chatbot. After these documents are uploaded, the system processes them, extracting metadata and content to enable effective querying.
- ✓ **Chat History Management:**
  The chatbot maintains a session-based chat history, allowing users to revisit previous conversations. Users can also clear the chat history to start a fresh session.
- ✓ **Cache Resources:**
  A cache is utilized to store frequently accessed data, improving the processing time and efficiency of the chatbot.
- ✓ **Integration with OCR (Optical Character Recognition):**
  The chatbot uses PyTesseract for OCR capabilities, allowing it to extract text from scanned documents and images. This feature enables the chatbot to handle a wider variety of document formats and provide accurate responses based on the extracted text.
- ✓ **Metadata Handling and Text Chunking:**
  The RecursiveChar TextSplitter divides the PDF material into digestible portions. Metadata is tagged with each chunk, containing the source.
- ✓ **Generation of Embeddings and Vector Storage:**
  The chatbot converts text pieces into high-dimensional vectors using Google Generative AI Embeddings. These vectors are then stored in an index called FAISS (Facebook AI Similarity Search), facilitating quick and precise similarity searches based on user inquiries.
- ✓ **Contextual Responses to Questions:**
  Based on the context gleaned from the PDF documents, the chatbot provides in-depth responses to user inquiries using Google Generative AI models and a customized prompt

template. If the solution is not found in the provided context, the chatbot notifies the user accordingly.

LangSmith Integration: -

LangSmith Introduction :

LangSmith is a comprehensive platform designed for building, monitoring, debugging, and evaluating production-grade LLM (Large Language Model) applications. It provides detailed insights and tools to optimize the performance and reliability of LLM applications.

- Building Production-Grade Applications: LangSmith offers robust SDKs for Python and TypeScript, facilitating the integration and utilization of LLMs.
- Monitoring and Debugging: Track and debug applications with real-time insights and detailed metrics.
- Performance Evaluation: Evaluate LLM performance using various metrics, such as average latency, total tokens used, and error rates.

Langsmith Integration with Chatbot:

1) **Creating an API Key :**
   a) Navigate to the Settings page on the LangSmith platform.
   b) Click on Create API Key to generate a new key.
   c) Copy and securely store your API key.

2) **Environment Setup :**

.env : Add your Langsmith ,LLM api key in .env file

ExampleGoogle_Api_Key = '<Llm Api Key>'

Langchain_Api_Key = '<Langsmith Api Key>'

Langchain_Project = "Gemini Chatbot"

3) **Project Setup :**
   - First import OS (Operating System)
   - Then call api from .env file
   - And then make trace true

```
import os
load_dotenv()

# Load the Google Generative AI Embeddings
os.getenv("GOOGLE_API_KEY")
genai.configure(api_key=os.getenv("GOOGLE_API_KEY"))

```

```
#Langsmith api load
os.environ["LANGCHAIN_TRACING_V2"] = "true"
os.environ["LANGCHAIN_API_KEY"] = os.getenv("LANGCHAIN_API_KEY")
```

Viewing Traces :

To view your output traces, visit the tracing section on the LangSmith platform.

Link : https://smith.langchain.com/o/a84fdbf6-4cc6-5c03-bca0-957caa12e9dc/projects/p/b952815e-bd60-4b4c-81b4-a8789e5839ad?timeModel=%7B%22duration%22%3A%227d%22%7D

Architecture :