Title : Multi-Agent LLM-Based Customer Service System for TeleCorp USA

Implementation Plan:

I had broken down the problem into 3 parts:

1. Marketing Data Agent
2. Technical Support Agent
3. Intent Recognition Framework and Agent Orchestration

Marketing Data Agent

Steps:

1. From the scheme of the table , I tried to build a data dictionary (updated the columns with description) using chatgpt, so we could have more context on the table.
2. I then used the schema along with the description and created a table local SQLite DB and loaded the data from the CSV file.
3. I created a prompt template in which I had feed the entire schema with the description and column names, and in that template, I took the user question as an input and I tried to produce a SQL out of it using gemini-1.5-flash.
4. Then I cleaned the output because sometimes when you’re trying to generate SQL code from the LLM, it has some additional inputs with it.
5. The SQL query was fired on the SQLite Local DB, and the tabular response from it was again sent to LLM along with the user questions, so that it can provide natural language response to the user for their questions.
6. As I am using gemini model, to generate an SQL from a given user query, if the user query is not proper, then the model asks a followup question to understand better. I fixed this with try catch in the sense if the user query is not proper, then it will not be able to produce the tabular response from the database, so in that case it will provide the initial response of the LLM analyzing the question to the user, and then user can improve the query

Technical Support Agent:

Steps:

1. We had got a document in which all the errors were mentioned, initially what I did is I converted the doc file to PDF and then I created a folder in which I uploaded that file.
2. The objective was that to build this system in such a way that it can handle a multiple files in a given folder.
3. Then we went through each file of the folder, extract information, then chunk the information, along with the chunk we keep the information about the page number as well the file name as metadata.
4. Once we have the chunks we embed it using text-embedding-001 model and store it in chroma db.
5. We have developed a retrieval system, in which whenever the user asks any query, first we do the similarity search of the user query in our vector db to find relatable chunks and the sources. The chunks, we send it our LLM, via prompt template, in which we have system prompts, we also send user query, and keep track of the history of the conversation. Finally the we share both the response from the LLM, along with the source to the user.

Intent-Recognition Framework and Agent Orchestration:

Steps:

1. We had built the agents individually, now it is time to orchestrate them, and also build a intent-recognition framework which would help to understand whether the user-query is for marketing agent or technical agent.
2. We specified the agents, that we wanted to orchestrate, then we build Graphstate, which is a dictionary of question , response and source, and keep tracks or has the state of the conversation happening using the multi-agent system.
3. Then we enabled the chat system using groq, provided the prompt to differentiate the questions for each agent, this is the router function,
4. Then we defined all the 4 functions that are the 2 agents, router function and added a generic agent which will respond when the query is neither for marketing or technical.
5. Then we added all the agents as nodes, defined the the conditional edges and finally ended the node.
6. Then compiled the entire workflow, and invoked by sending in a question.