Langchain End to End (User txt to MQL)

This is full stack application with backend and frontend integration for User text to MQL.

File structure

Langchain End to End (User txt to MQL).zip

- BackEnd
  - filegpt.json
  - langchain_final.py
  - main.py
  - requirements.txt
  - Data_update.csv
  - path_df.csv
- FrontEnd
  - client(folder)
    - script.js

BackEnd

API link: http://127.0.0.2:8000

filegpt.json: It contain the input data in structured JSON format.

langchain_final.py: It contain all relevant function with langchain(OpenAI and huggingface).

main.py: It contain creation of FastAPI (Frontend API) with endpoint “langchain”.

requirements.txt: It contain all the dependences for this project.

Data_update.csv: This file created from the preprocessing of input document data.

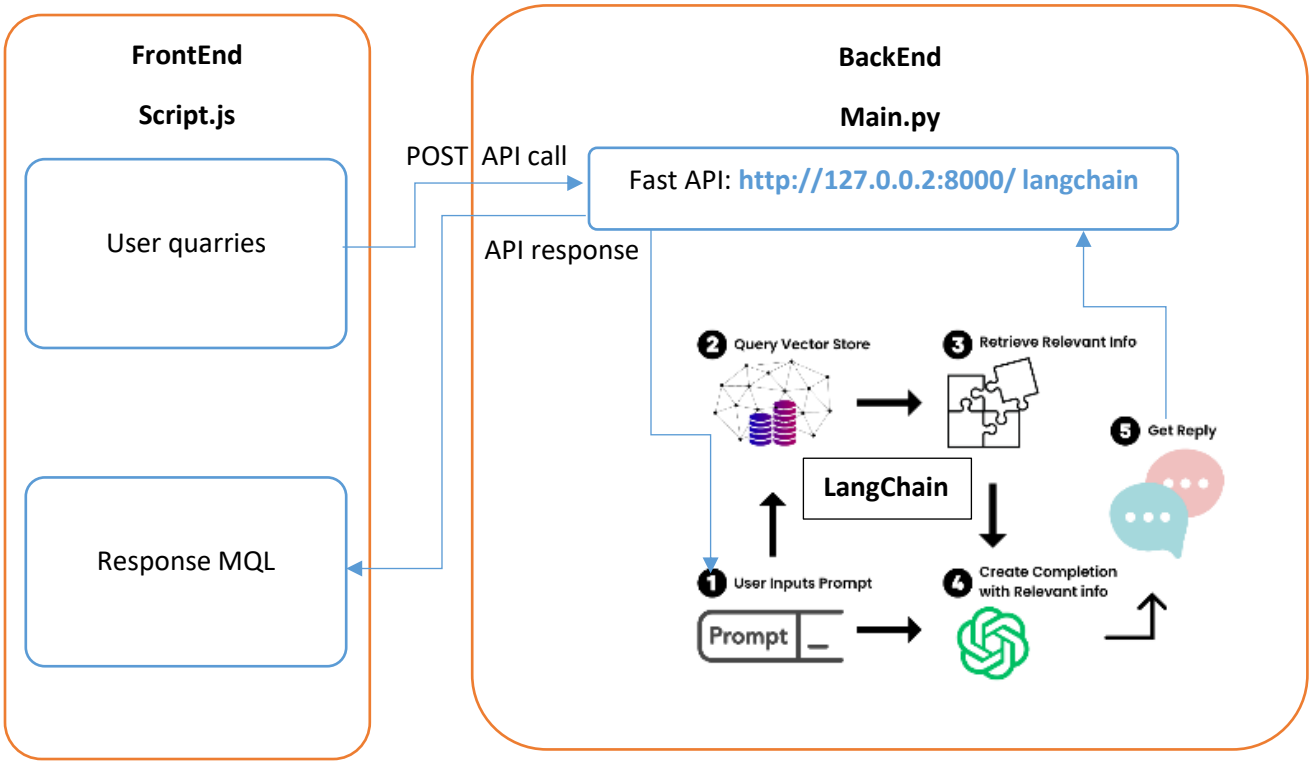
path_df.csv: This file is contain all path of the Data tree.

FrontEnd

API link: http://127.0.0.1:5173/

client(folder) -> script.js : This .js file contain the front code

Code flow



Initialized PromptTemplate with Final_prompt_text and query variables:

```
prompt = PromptTemplate(
    input_variables=["Final_prompt_text", "query"],
    template=template_text,
)
```

Prompt for MQL Generation

```
template_text = """
   Suppose 'top_doc' is a MongoDB collection
   As the custom data is large the data tree is created for the custom data. The path of the tree is recorded.
   -> Denote edge of the tree
   MongoDB statement 1 : top_doc[0]["companyName"]
   Tree path for above MongoDB statement 1 : top_doc -> companyName
   MongoDB statement 2 : top_doc[0]['hqlocation']['city']
   Tree path for above MongoDB statement 2 : top_doc -> hqlocation -> city
   Then best path which is related to human query is also selected and given in the Final_prompt_text.

   Now, consider yourself as a bot that returns MQL corresponding to specific human readable query.
   You must consider only the Final_prompt_text structure to returns MQL.
   You must consider only the Final_prompt_text variable names to returns MQL.
   The user may ask you complex queries where you have to create MQL.
   And you only return MQL and nothing else !!!!
   Important Constrain: Strict to the variable names with in Final_prompt_text for MQL

   # Training Phase
   query: "give me hqlocation geo location of GlobalMed Inc.?"
   Final_prompt_text: "
   Iteam 0 :
   top_doc[0]['companyName'] =  GlobalMed Inc.
   top_doc[0]['countries'] =  ['United States', 'United Kingdom', 'Australia', 'Singapore']
   top_doc[0]['hqlocation']['city'] =  Phoenix
   top_doc[0]['hqlocation']['area'] =  dummy
   top_doc[0]['marketingSpendTechStack']['serviceProviderUse'] =  ['Digital Marketing Agency', 'Telehealth Platform Provider']

   Iteam 1 :
   top_doc[1]['companyName'] =  WNS Global Services
   top_doc[1]['countries'] =  []
   top_doc[1]['hqlocation']['city'] =  Mumbai
   top_doc[1]['hqlocation']['area'] =  India
   top_doc[1]['marketingSpendTechStack']['serviceProviderUse'] =  ['IBM,Microsoft']
   "
   MQL statement : db.top_doc.find(!-companyName-: -WNS Global Services- !!, !-hqlocation-: 1!!)

   # Testing Phase
   query = {query}
   Final_prompt_text: {Final_prompt_text}
   MQL statement :
```

How to Run

Step1:

Run man.py it show like below:

Uvicorn running on http://127.0.0.2:8000 (Press CTRL+C to quit)

Now Backend FastAPI run successfully.

Step2:

Enter following cmd in the terminal:

cd client

npm run dev

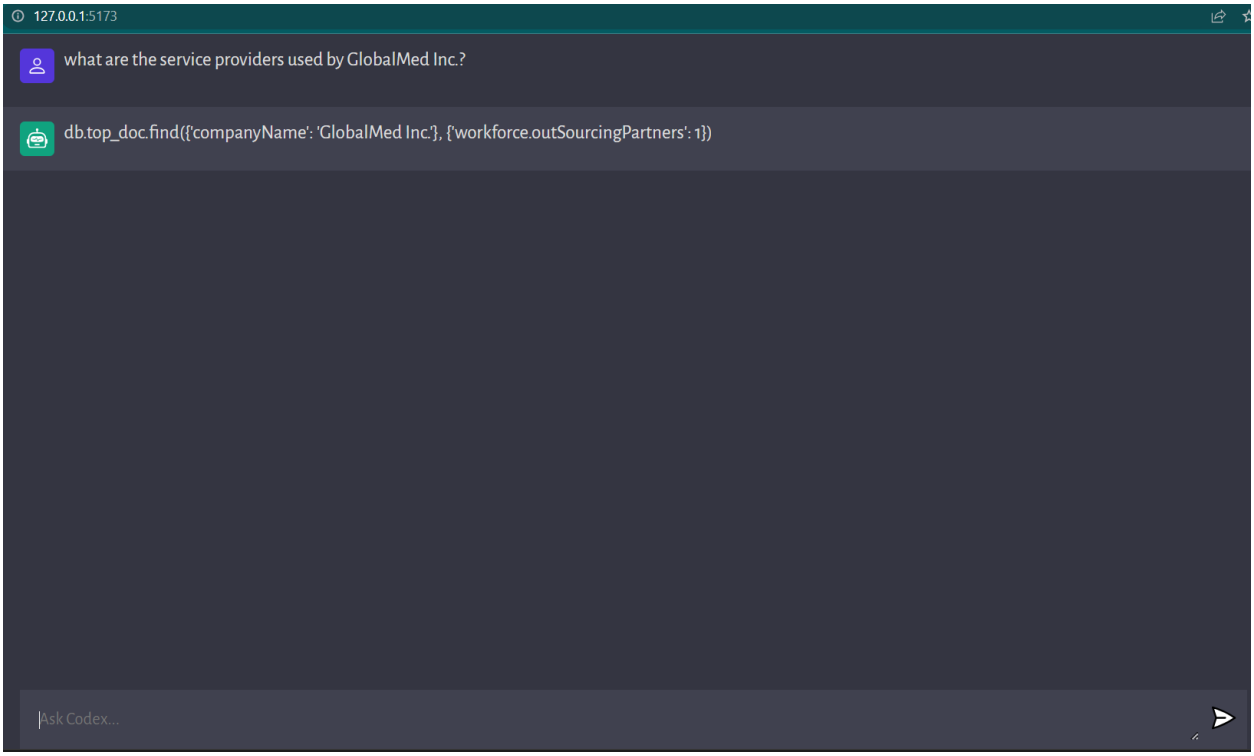
if it show like below:

Local: http://127.0.0.1:5173/

Then FrontEnd run successfully.

Step3:

Now open : http://127.0.0.1:5173/ in browser .



Now you successfully Langchain End to End Project !!

Ask your query then get MQL response.