# Summary of task:

You need to make a User Facing chat style RAG Agentic System in which the user will ask queries to LLM and LLM will provide the answer Using given tools (tool call/function call). The condition is that data must be up to date daily, ie: you will also need to make a data pipeline which will update data daily into your local MySQL or Vector Database ( VectorDB is out of the scope of task but if you do it , it's good).

## Concepts/Required Knowledge to complete the task

1. Asynchronous Programming in python

2. Setting up MySQL Database & Querying the Database

- In Querying the Database Avoid ORMs, Use RAW SQL Queries, it is not needed that you know hard and every query. simple select , data filtering and basic table joining is sufficient

3. Creating Basic APIs in python (FastAPI will be Simplest choice here.) or websockets based interface with LLM.

4. Agentic LLM inference

- For this task , Agent is simply LLM with having predetermined tools/functions [ex: fetch\_result\_from\_google(query: str), function which agent can use to query google by providing us query arguments] at it's disposal to solve the tasks.

- if you are good developer and are familiar with everything except this agentic system, then just understand it as in LLM system prompt we suggest LLM to use functions based on user query and give LLM function schema (function schema is just definition of function basically name of the function ,arguments function takes and what are the type of the argument). LLM returns us which function to use and with what argument in JSON and we parse the json , use python eval() to execute the function and return the result back to LLM, which then LLM summarizes the results and gives the answer to the user.

5. Basic Data Processing

- what is data pipeline?: In simplest definition it is nothing but automation of data processing from data download to clean to pushing to database without near 0% manual input after data pipeline creation.

- you do not need to setup cronjob/schedulers for this task , during demo we will ask you to just run the pipeline and will wait for the data update in DB.

6. Ollama For running local LLM ( use qwen2.5 0.5b or 1b ) if you don't have GPT key. we advise you for testing purpose use local LLM as OpenAI APIs will be costly.

7. Optional: Basic HTML, css, Js for chat and input box.

- Note: it is not needed to have any fancy good looking UI , simplest plain white HTML page with result box and input box will be perfect as long as we can chat with Agent(LLM) through API. You can use AI for this portion. Streamlit , gradio , solara all works.

## Outside of the Scope OR Non required Stuff.

1. Using OpenAI API (Local LLM works)

2. Scraping for data gathering ( Avoid it , use any data which you can get from free API and is daily updating)

- note: Weather Data APIs does not count

3. Vector Database Setup

4. Logging of User and Chat Specific Agent Activity in queryable format. ( sounds simple but complicated and out of demo scope)

5. Fully Functional APIs with Auth and MongoDB history Management

- This will go way beyond the scope of the demo and will become full on project.

- API does not need to be fully functional , only required function is when we hit the api we can chat with the Agent , that's it. History management is not needed as you will need to setup chat\_id and user\_id system which will again go beyond demo. but if you want to do it then just use plain SQLite or even json file with demo sample chat\_id and user\_id.

- You don't even Need to make FastAPI APIs , You can use websockets and it will still count , all that is needed is interface with Agent.

6. Agentic Libraries such as `pyautogen` , `autogen-chat` , `langchain` are not needed you can use barebone OpenAI python client ( supports local ollama all you need to do is change base url from openai api endpoint with your local ollama api endpoint). In fact we would advise you to use Barebone OpenAI python client as agentic libraries are fully bloated and complicated simple stuff unnecessarily, and OpenAI python client by default supports Async. but yes there is no specific guideline here , complexity is subjective you can use Any framework you like/prefer for making Agent As long as it Supports Fully Async non blocking functionality either by built-in or by you manually. Do note that if you go with OpenAI client approach you will need to make tool call parser and executer which is by default available in almost all agentic libraries

## Conditions

### Compulsory Conditions

1. Agent function/tool calls should not be visible to end user

2. Data Must be Daily updating. Static Data Won't count. Do note that daily updating here does not mean if you run today and you must have today's date data. it can be week old. majority of systems data update frequency is between 1 day to week.

3. No Code Interpreter/Execution by LLM. Only tool/function calls.

4. Agent will not query the DATA API directly , Agent can only use your MySQL data and other utils that you may provide.

### Optional but preferred Condition

1. Everything must be async and non blocking (In Agent and User Agent communication interface. for pipeline it is ok to go with sync as this is just a demo).

# Task Breakdown.

Task is divided mainly into 4 parts.

1. Data pipeline

2. Agent Creation with tool calls functionality

3. API/Communication interface with Agent

4. Basic UI.

## 1. Data pipeline

Choice of Data will determine your system and end project. here are few data sources which meets all the conditions of required Data.

of course you can use whatever Data you see fit as long as it meets conditions.

1. [Federal Registry](https://www.federalregister.gov/developers/documentation/api/v1#/Federal%20Register%20Documents/get\_documents\_\_format\_)

- This is USA federal registry data API which contains executive documents and other registry related data. if used this data then user queries for example would be `what are the new executive orders by president donald trump this month and summarize them for me.`

- note: queries won't be related to semantic similarity as we have suggested to not setup vector database. example `what are the executive documents related to artificial intelligence and security in all the past years`

- This data is daily updating and you can query the API to fetch date specific updates after , before data. and it is free.

- you only need to get the 2025 dataset.

2. [FDA DRUG RELATED DATA](https://open.fda.gov/apis/)

- You can use any dataset from them , related to adverse effects of drugs , reported incidents, disease related.

### Best practices in the data pipeline.

This will make development easy

1. Downloader and Processors Must be separate

2. Records must be kept atleast 1 week pipeline records needed to be kept (example: daily downloaded raw data, and processed data)

## 2. Agent Creation with tool calls.

Agent is really fancy word of saying `LLM with specific task and tools at its disposal to solve it`

- You will find many tutorials for this and you can use any framework.

- if you do plan to use barebone OpenAI python client or direct aiohttp request to openai/local\_ollama endpoint , you will need to make a system to render/execute LLMs generated tool calls, which might get little extra but it is not that hard. all you need to do is parse the tool call request and execute the function and return the result.

make sure to only execute functions which you have defined. otherwise `eval()` will be very dangerous to use.

- basic overview is that

User Query -> LLM , LLM thinks and decide if need tool to use and yes then which tool -> tool call response -> Your system executes the tool -> returns result back to LLM -> Again LLM Decide need to response (here it means: answer to user query) or need to execute another tool , say it decides to make response -> returns Response -> User.

- if you plan to use an agentic library then also it is ok.

## 3. API/Communication interface with Agent & 4 Basic UI.

- Pretty simple and straightforward, You just need to connect users with LLM using API or websocket.

- In simple terms user enters query to UI -> UI calls Your API -> it gives user query to LLM -> and process happens same as (2) -> API returns returns result to UI -> UI renders it and shows the response to user.

## Overall

Data Pipeline -> MySQL : will daily fill the data.

User Query -> YOUR INTERFACE API -> LLM -> tool call to use MySQL to get user requested data -> LLM summary -> Response -> API returns the result to UI.

## Other Important Notes:

- Treat it as demo , Not full project , You can cut corners for example:

- in data pipeline you only get past 2 month data instead of full 2025,

- if not able to understand async programming then stick to sync.

- We are aware of the complexity of the task , this is moderately complex but actual production/real world grade task.

- Asynchronous Programming is a big ask here. the original task does enforces fully async system but if you are able to architect the overall system but have a hard time in async programming then you can stick to sync.

- Using LLM for code help and understanding project is allowed , but make sure to understand system design by yourself as this is not a CRUD task.

- At the end all we want to see is can you design the actual system. and do note that the prototype does not need to be perfect, it can be improved laterwards.

## Few Async Libraries which will be useful for your task

1. aiohttp (for making requests)

2. aiofiles ( for writing to files)

3. aiomysql ( for async mysql query , use this as lot of documentation is available for this one)