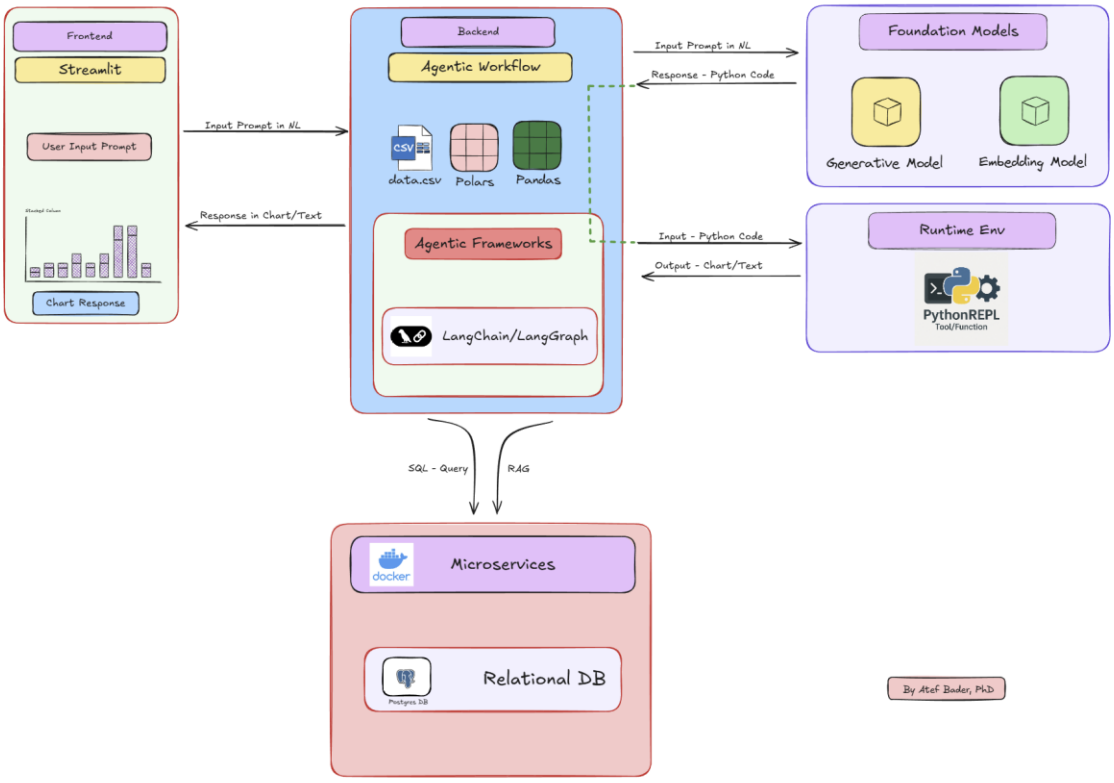
Bonus Assignment #1



Deliverables:

Your assignment submission ZIP file (("Bonus_Assignment_1_YourLastName")) must have the following items in it:

1. Your **ZIP** file must have two directories: **src** and **doc**
2. Your **src** directory must have **requirements.txt** that has package dependencies to install through **uv pip install**
3. Your **Readme** files must have the detailed steps to install, deploy, and run every IPYNB script and required packages
4. Use **Panopto** to create 5-10 minutes **video** for a live-demo of your application
5. All source code must be stored under **src** directory

6. Documentation and video must be stored under **doc** directory

Important Notes:

1. The **grading** for the bonus assignment is **BINARY: credit or no credit**; there is **NO partial credit** for semi/partially completed requirements

2. You must verify the code submitted for completeness, correctness, consistency, coherence. Zero credit for cope/paste code from GenAI that doesn't meet the requirements.

Instructions:

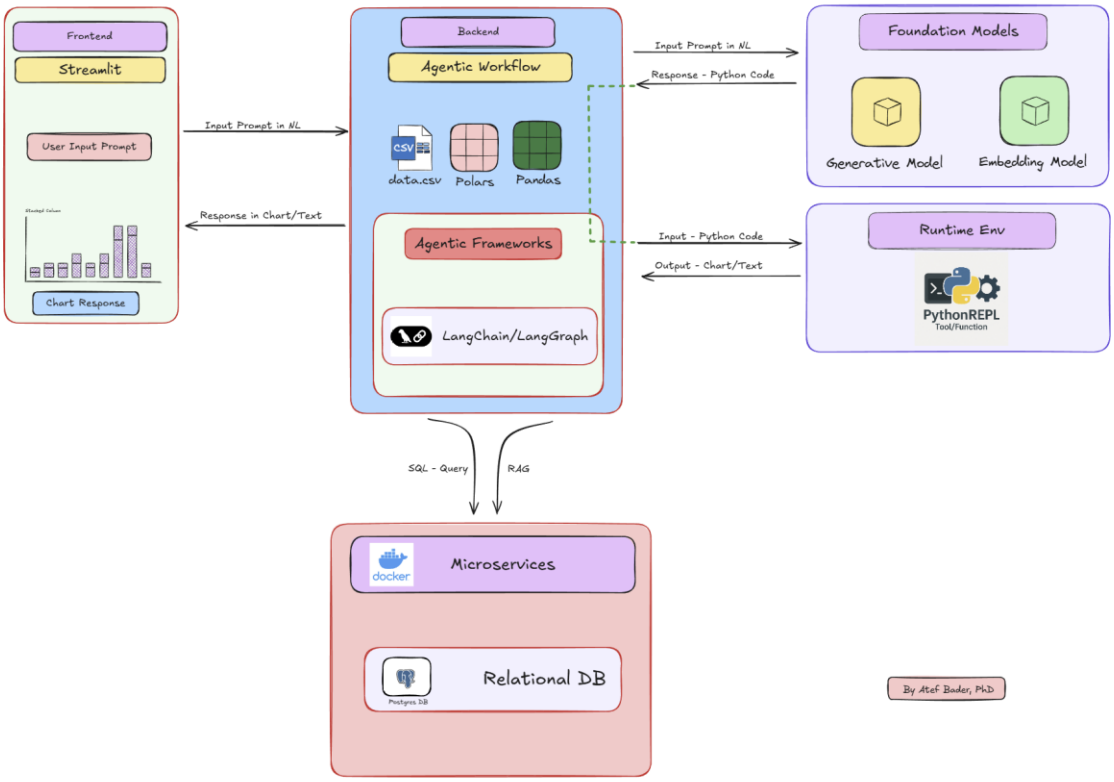
1. Use **Python/GitHub API** to retrieve information of the **past 2 months** for the following repositories (JSON to CSV data):
   1.1. https://github.com/meta-llama/llama3
   1.2. https://github.com/ollama/ollama
   1.3. https://github.com/langchain-ai/langgraph
   1.4. https://github.com/openai/openai-cookbook
   1.5. https://github.com/milvus-io/pymilvus/

2. Store the information (CSV data) for these repos in **Postgres** database tables

3. Implement the requirements listed under the **Requirements** sections.

Requirements:

1. Use Python 11, Polaris, LangChain/LangGraph, Streamlit, gpt-4o-mini to implement the following architecture



2. Your Python program/UI (**Streamlit**) must allow the user to input queries written in natural language, produce output in textual/tabular format and visual/graphical/chart format.

3. Your INPUT PROMPT will be written in Natural language and processed by LangChain/LangGraph to use the **LLM** to generate Python code for the requested input prompt

4. Use **PythonREPTool** to execute the **Python** code generated by the **LLM**

5. Answer the input queries below based on the data stored in **Postgres** database tables

6. Queries entered as input in natural language (enter every query as-is in Streamlit/input) and output in **textual/tabular** format:
    1) Which Repo has the highest number of issues created?

    2) Create a **table** of the total number of issues created for **every repo** for **every day of the week**; that is total number of issues created on Monday, Tuesday, Wednesday …, Sunday for EVERY Repo name.

    3) Which day of the week has the highest number of total issues **created** for ALL repos

    4) Which day of the week has the highest number of total issues **closed** for ALL repos

7. Queries input in natural language and output in **Chart** format:
    1) Plot a line chart of total issues created over time.

    2) What is the percentage distribution (create Pie Chart) of issues **created**.

    3) Create a Bar Chart to plot the **stars** for every Repo

    4) Create a Bar Chart to plot the **forks** for every Repo

    5) Create a Bar Chart to plot the issues **closed** for every week for every Repo

    6) A Stack-Bar Chart to plot the **created** and **closed** issues for every Repo

    7) Use **Facebook/Prophet** package to forecast/chart and plot the **created issues** for every repo

    8) Use **Facebook/Prophet** package to forecast/chart and plot the **closed issues** for every repo

    9) Use **StatsModel package** to forecast/chart and plot the **pulls** for every repo

    10) Use **StatsModel package** to forecast/chart and plot the **commits** for every repo