Reflection

Group Members: Hima Sineni, Aryana Bellamkonda, Mariana Mendez, Anjana Raman, and Caroline Nguyen

For this project, our goal was to build a chatbot using Flask that could answer questions based on two types of data: one from a live API and another from a local dataset we processed ourselves. We chose to focus on natural disasters because they are a globally relevant topic that impacts millions of lives and economies worldwide. Exploring this data helps raise awareness, support disaster preparedness, and reveal patterns across countries and years. Our chatbot can answer questions like how many disasters occurred in a country during a certain time period, what the deadliest disaster was in a given year, and also report the current weather at any location using live data.

We selected our local dataset from Kaggle's Natural Disasters dataset. This dataset had detailed records of disaster events around the world, including information like the year, disaster type, country, number of deaths, and the location (latitude and longitude). We liked that it had a wide range of data going back many years, so it could be used to explore different types of questions. For our live API, we used the Open-Meteo API to provide real-time weather information. This helped expand our chatbot's capabilities beyond just historical data. We wanted users to not only learn about past disasters but also to be able to check the current weather at any location by giving coordinates.

One of the biggest challenges was working with the ETL pipeline. The raw CSV file from Kaggle had many columns and some missing or messy data. In our etl.py file, we had to extract only the useful columns, rename them for easier use, and clean up missing values. We also created a new column called search_key to help with future filtering or searching if we wanted to expand our chatbot's features. Additionally, we found ways to handle errors smoothly, like if the user input was invalid or if the weather API was down. We made sure to give clear error messages so the chatbot doesn't just crash when something goes wrong.

This project does a great job in conglomerating all the main tasks and skills we learned this year into one purpose of building a chatbox. However, throughout this project, we learned a lot more about the purposes of google cloud platform and why it is important to connect localhost work to GCPs. By using google cloud, we were able to collaboratively work on the project and create a chatbox that is for public domain use. This skill is essential in shaping productive and sufficient technological advances and codes to help create sharable projects. This way, we can enforce authorization through firewall rules, tokens, and other security measures to ensure safety for all users. These skills we have now gained experience with will allow us to collaborate and build new projects in the workforce as well. Along with the use of GCPs, this project gave us further

insight into APIs and how they can be used to communicate between different datasets and platforms. For example, by using the weather API, we were able to connect our chatbox to open source platform and compare data, which allowed us to build an innovative project. And, by doing so, we were able to learn to utilize external sources in handling real-time data (open API). These skills that we learned in class helped us understand their practical and applicable knowledge through this project, establishing a valuable experience.

Given more time, this project could be further enhanced with more Gemini integration in terms of user-friendliness. Since the Gemini outputs raw data when asked questions by the user, it would be more friendly to program Gemini to give “normal” responses to questions in a sentence format. This would be enhance our project and make it more appropriate for industrial and consumer use. Working on this aspect of the project will make this feature more applicable and initiate further consumer use, leading to more evidence for upgrades as more users would be inclined to utilize this feature to learn about the weather. For example, instead of returning: “Temperature 78°F,” it could say: “It is currently 78°F in Charlottesville, a great time for a picnic on the lawn!” By personalizing these responses, it enhances the quality of our project and ensures all users feel welcomed and appreciated. Additionally, Gemini could be integrated to answer “poorly-worded” questions or vague questions from users, given them many options for answers instead of just spitting out raw data that may answer the users’ questions. This allows for the chatbox to be a scalable and appropriate resource for consumers about the weather.

Overall, this project and its outline helped build our skill repertoire in many ways, including integrating Flask, building a chatbox, and utilizing GCP and APIs to understand the weather around us.