Write-up – eco-and-rebano-tracker – Ivan Morales

- ***Problem Statement***

Living in Guadalajara is amazing, going into some news sites and scrolling through all of the clickbait to read the concise information at the end of it, is not, at the same time, monitoring local sustainability metrics like the level of Chapala Lake requires digging through unfriendly government websites, this only creates friction for one to stay informed efficiently about two topics that I am passionate about, Chivas and Sustainability.

The solution comes with a bot that gathers the info and displays it on a Streamlit Dashboard sending you a brief email every day to wake up informed.

This project was indeed a challenge for me, I had heard about Agentic AI, but what exactly is an AI Agent, prior to the execution of this project I did a quick research to fully understand what am I working with, according to Google Cloud, "AI agents are software systems that use AI to achieve goals and complete tasks on behalf of users."

With the following in mind, I started working.

- ***How I approached the problem.***

My approach was to build a modular architecture, separating data ingestion, processing and presentation, following the "AI Agent Workflow" concept.

3 main aspects are the ones I decided to present in the dashboard:

1. Air Quality in Guadalajara's Metropolitan Zone (ZMG)
2. Chapala Lake Level, which has been an important topic recently in the region
3. News analysis about both environmental and Chivas topics to complement and enrich the dashboard.

The project architecture relies on five key components:

1. app.py, the front end, I used Streamlit because I was already familiarized with it, the software provides a fast and reliable creation of a well-made user interface.
2. utils.py, caching mechanism, this script creates a local cache (JSON based file) to avoid using tokens repeatedly
3. chivas/data.py
4. environment/data.py, the backend, both scripts are responsible for fetching the required data for the dashboard.
5. daily_briefing.py, works as the orchestrator to send emails daily, working with a CRON function that runs on GitHub actions.

- Where and how you used AI tools in development.

AI tools are incredibly powerful, for this project I used two of them, Cursor.ai for the principal layout of the project structure, to support the development of the project and to refine the code Cursor did I used Gemini, the second also helped me fully understand each of the classes and functions used in the whole process, Gemini also helped a lot condensing code going from 1000 lines to 300 lines of functional code to put an example.

In the heart of the project lives Gemini-2.5-flash and Gemini-2.5-flash-lite these 2 models are in charge of analyzing the news and prompting the summaries.

- ***What challenges you faced and how you solved them.***

2 principal challenges were faced during the creation of this project,

A) Unstable Data Sources
B) Excessive API tokens consumption

About the unstable data sources, the first idea was using mock data, using this kind of data prevents the scripts and the data visualization to fail, this way the UI remains functional if any of the data sources fails.

The problem of the Token consumption was solved with two actions, first one, adding billing information to my Google AI studio account, this provided me with $300 USD worth of tokens, more than enough to cover the project necessities, alongside and with the help of Cursor and Gemini, I implemented a caching system that saves the prompted information for later, this prevents to use tokens repeatedly, it makes sense to do this because the information will be consulted daily, the same remains available allocated in streamlit with the use of @st.cache_data.