Hello!

I’m writing to officially hand over the data portal. Here are several important points to note:

1. I’ve set up a Gmail account to manage everything: mobilab.unilu@gmail.com. The password is Francesco1528\*. If you ever forget the password or need more details, Francesco, the sticky note you have, contains additional information like the date of birth, among other things.
2. I’ve used this email account to create a GitHub account (<https://github.com/>) named “mobilab-unilu”, a Docker account (https://hub.docker.com/) for using Docker Desktop, and an ngrok account (<https://ngrok.com/>) to expose the Airflow UI and the Postgres database which can be accessed via DBeaver Community (https://dbeaver.io/download/). The password is the same for all these platforms. It’s recommended to change the passwords if you see fit.
3. On GitHub, I’m currently the administrator of the repository found here: <https://github.com/jdpinedaj/luxmobi>. However, I’ve added the mobilab profile as a collaborator. My being an administrator means that if you wish to make any changes and modify the repository via a “Pull Request”, I must approve it before merging the changes. If this becomes the case, it’s crucial to understand git/GitHub and branch management to avoid causing chaos in the repository. Nevertheless, you can edit the repository locally without any issues.

Now, let’s delve into the general aspects of the repository.

The repository consists of various specific files:

* **README.md:** This is the project’s README. Here, I explain the different sources of the data portal, the workflows I’ve created, and a general method to set up Airflow in a Docker container.
* **Requirements.txt:** This file lists the required libraries to run the data portal using Python.
* **Pyproject.toml:** This is part of the poetry environment manager I used to install Python libraries. It helps avoid conflicts and chaos with library versions.
* **Docker-compose.yaml:** This file allows you to mount the Docker container with the project.
* **Dockerfile:** An additional file that installs extra requirements in the Docker container to ensure the entire portal operates correctly.
* **.gitignore:** This is a standard file that determines which files get uploaded or ignored when building the data portal.
* **LICENSE:** This is the project’s license. It’s mainly a standard procedure and not of significant concern.

Regarding the directories, the following are included:

* **Readme-resources:** This is where I stored the image that’s visible in the README.
* **Notebooks:** Currently, this folder is empty, but this is where I tested everything locally using notebooks before transitioning to production code.
* **Docs:** I’ve saved a .docx file here which explains how to expose the Postgres database to the internet using ngrok. It also provides guidance on accessing the database from both an administrator’s perspective and that of a regular user. Please note: an administrator can accidentally delete tables; a regular user cannot. Be cautious when sharing credentials.
* **Dags:** This is where the workflows (or DAGs) are stored, which I’ll explain next.

A DAG (Directed Acyclic Graph) (https://en.wikipedia.org/wiki/Directed\_acyclic\_graph) is the foundational structure upon which workflows are built in Airflow. As explained in the README, there are a total of four constructed workflows.

Each workflow follows the conventional structure of how DAGs are set up in Airflow, comprising:

* **Importing modules:** This is where all the necessary modules for the workflow are imported.
* **Defining default arguments for the DAG:** Here, any default parameters or settings for the DAG are outlined.
* **Instanciate the DAG:** This involves initializing the DAG with its unique ID and other specified arguments.
* **Tasks:** Tasks represent the core activities or steps within the workflow.
* **Setting up dependencies:** This entails establishing the sequence or order in which tasks should run, determining which tasks are dependent on the completion of others.

The “tasks” section is where the magic happens. Here, the specific activities are delineated in the order in which they are to be executed. Many of these tasks involve simply running a Python function, providing the necessary arguments. Others are Postgres (SQL) tasks where parameters are also passed.

The SQL queries and Python functions are in separate folders nested within the “dags” directory. Within the “dags” directory, you’ll find the following sub-folders:

* **Folder custom\_operators:** These are some custom operators that I created for specific workflows. However, in the end, I did not use them.
* **Folder data:** This is where all the CSV files will be stored. Every hour, each data source’s data will be stored in its separate folder.
* **Folder sql:** This contains the SQL queries used to establish schemas and tables in the Postgres database. These are especially utilized in the “workflow\_create\_tables” workflow and in others to transfer data from the CSV files into Postgres.
* **Folder scripts:** This directory contains several .py files encompassing all the Python functions employed across different workflows to extract and transform data. Inside the “scripts” folder, there’s another sub-folder named utils. Within it, there’s an additional file, “logs.py”, which I used for managing loggings in the Python functions. It’s responsible for generating various message types, such as warnings, info, etc. This is considered one good programming practice. For more info on logging, you can see this: <https://docs.python.org/3/library/logging.html>. Moreover, there’s a file named “config.py”, which contains all the parameters used across the different workflows (like links, keys, names, etc.), eliminating the need for “hardcoding”.

It’s essential to point out that the code is entirely modularized. I’ve developed generalized functions that, in turn, call sub-functions. I’ve endeavored to craft them as broadly applicable as possible. The intent is for the same functions to be usable across all data sources, with only the arguments to each function requiring modification as needed.

The fundamental skills required to effectively work with everything here include:

* Python
* SQL (Postgres)
* Docker
* Apache Airflow
* Git / Github
* Additionally, I recommend working with Ubuntu. Personally, I operate with WSL – Windows Subsystem for Linux (<https://learn.microsoft.com/en-us/windows/wsl/install>).

Everything is fully operational, except for the GPT data because the website is dynamic, and it broke my web scraping attempt (haha). I’ll be looking into understanding the data extraction costs necessary for the project and I will let you know!

Please take some time to review the entire data portal. Let me know if you need a more detailed explanation on any specific aspect. I can then create a supplementary document in the “docs” folder or improve the README to provide clearer guidance.

Thank you!