Top Cryptocurrency Price List

For this task, you will prototype a price list service for top crypto assets.

The service should expose an HTTP endpoint, which displays an up-to-date list of top assets and their current prices in USD when fetched.

- The endpoint should support `limit` parameter which indicates how many top cryptocurrency types should be returned.
- The endpoint should support `datetime` parameter which indicates the timestamp of the returned information. An optional parameter, the default is NOW. If the request is without a defined timestamp, the information should be the most up-to-date from the external services (not from the historical database).
- The output should be either `JSON` or `CSV` compatible, defined by the parameter `format`.

Example call should look somehow like this:

```
$ curl http://localhost:6667?limit=200

Rank,   Symbol,         Price USD,
1,      BTC,    6634.41,
2,      ETH,    370.237,
3,      XRP,    0.471636,
...     ...     ...
200,    DCN,    0.000269788,
```

The ranking and price information should always be up-to-date. For example, let's say that BTC ranking changes from #3 to #6, the list should reflect that change. The historical information should be preserved and accessible with the help of the `datetime` parameter.

Send us only a **link to your GitHub or GitLab repo.** In the repo **include a README.md** file.
We will discuss your solution during our technical call. You will be asked to do a live demo during the technical call

Tech stack:
Python (mandatory)
FastAPI (preferred) or Flask
Docker, preferably docker-compose

Data Sources

To make the challenge a bit more interesting, we ask you to:

- Use coinmarketcap API to get the current USD prices.
- Use cryptocompare API to get the current ranking information for the top assets. Toplist by 24H Volume Full Data.

We know that you can get all the necessary data from either one of those but part of this challenge is to see how you deal with the problem of merging information from multiple data sources as well as fetching data in an efficient way from multiple sources. Please assume each data source returns a significant amount of data and that the response time is also non-trivial.

Architecture

Your solution should consist of at least 3 separate services that run independently (service-oriented architecture):

- Pricing Service - keeps up-to-date pricing information.
- Ranking Service - keeps up-to-date ranking information.
- HTTP-API Service - exposes an HTTP endpoint that returns the list of top cryptocurrency types prices.

Feel free to add more services if you see the need.

You're free to pick any pattern for inter-service communication. We ask you to explain the rationale behind your choice in the README, some of the most well-known patterns are:

- Publish / Subscribe over a messaging bus such as RabbitMQ, NATS, MQTT
- HTTP API
- Remote Procedure Calls
- Shared Database

Hints

- You're free to use any libraries. Think about the best fit for solving the task. Remember that the next step would be a pair programming session with one of our engineers, so you must be comfortable with your decisions.
- Make sure performance and memory consumption concerns are taken into account in your solution. Even though the example data size is trivial, assume it exceeds the main memory and the fetching time is significant for each data source.
- Make sure your approach is not a naive sequential call for two APIs.
- How about any reasonable caching to protect against notorious refreshing?
- How about scalability?
- We highly recommend you use docker and docker-compose for orchestrating your solution.
- Include tests.
- Don't forget documentation, both code and user. Help us to run your code and understand it without hassle.