Python Developer Exercise

1. Create a project called **Crypto Converter** that provides an API to convert amounts of crypto currencies back and forth using HTTP JSON API. Host it on Github or Gitlab. If you decide to make it a private repo, pls share with the following accounts by email.

Note that this is not a swap service, it should just calculate the numbers, and shouldn't do any real currency operations / transfers.

1. m.stabla@sparkland-trading.com
2. d.bogomolov@sparkland-trading.com
3. s.levitin@sparkland-trading.com
4. p.kamaev@sparkland-trading.com

The project must consist of two parts that have to run as separate processes.

1. **Currency Conversion API**
2. **Quote Consumer**

Common Requirements:

1. Both components are configurable using env variables.
2. It must be possible to just run `docker-compose up` and then use the API.
3. It's fine to have the same docker image for both components.
4. It should be a single entrypoint for running both components . Usage example: `python run.py api` or `python run.py quote-consumer`.
5. It should have an informative documentation about the project purpose and how to run it.

Currency Conversion API requirements:

1. It should provide an endpoint `/convert` that accepts `amount`, `from` and `to` as HTTP GET query parameters, and returns a response that contains the amount in the `to` currency and the conversion rate used.
2. For simplicity, we can assume that we convert prices as is, without any extra fees.
3. Also, for simplicity, cross-currency conversion, e.g. `BTC` → `USD` → `ETH` shouldn't be implemented.
4. The API must return an error if conversion is not possible, e.g. if we don't have quotes for this pair.
5. It must use the latest available quotes received by **Quote Consumer**. If they are older than 1 minute, it must return an error like `quotes_outdated` .
6. It must use `asyncio` for serving requests. It's fine to use any framework like `aiohttp`, `fastapi`, whatever.

Quote Consumer requirements:

1. It must subscribe for quotes from Binance API or any other exchange and save them into some storage that can be used by the API to get the quotes. No API key is required for this.
2. It can subscribe via either websockets or HTTP.
3. It should save quotes into the storage every 30 seconds.
4. It must remove all quotes from the storage that are older than 7 days.

Bonus points (but not required to implement).

1. Make it possible to pass a timestamp as a query parameter, so it converts using quotes for a specific date, if available.