Trade Republic Challenge

Thank you for your interest in Trade Republic!

We have prepared a short challenge for you. Our goal is to see how you work, how would you solve a real-world problem and the tools you are comfortable with.

To keep the test brief, in your code you do not need to worry about algorithm efficiency, logging/monitoring or persistence of data (you can just use sqllite or mock api calls).

But as you'd be working inside a team collaborating with other engineers to improve our products we do care about software engineering best practices that promotes extensibility, reliability & scalability.

Looking forward to seeing your work :)

The Task

We want you to build a service that will provide Stock information for our Android/iPhone apps. The service should read instrument price data from an external service (detailed below) and should expose the requested endpoints with appropriate instrumentation to deal with the data needs of the business case.

At this moment, we're interested in providing 2 endpoints:

- `/instruments/{id}` - Retrieve the last 5minutes of price history with minutely resolution.
- `/instruments/{id}/similar` - Return the 10 most similar stocks to the given Stock. (check details for the algorithm below)

You can mock any calls to underlying infra - just make sure you document those (example below).

Follow up questions

1. How would you change the system to support 50k instruments each streaming one or more quotes every second?
2. Which changes are needed so your code could support 100M daily active users hitting each endpoint 10 times/day?
3. What would you put in place to make the sure the system and the model are working as expected?
4. How would you evaluate and improve the Similarity model?

If you answer on any of these questions with a specific technology, framework or architecture, please explain which properties of the technology makes it feasible for the question.

Technical specification

External Service

In order to allow you to test your code, we provide a mock API for the external InstrumentStream service.

Here's an example on how to use it:

```
stream = InstrumentStream()
for event in stream.read():
    print(event)
# { "id": 1, "price": 12.34, "timestamp": 12345678 }
```

If your language of choice is not Python, please create a simple mock that handles the event generation.

The Similarity Stocks model

We're implementing a simplified version of the logic that Amazon uses for the section "Customers that viewed this product also viewed:".

We'll compute similarity by using the number of users that saw both stocks in the same day. You can call it an item based collaborative filtering model with implicit feedback.

Assume we've received the requests:

```
2021-01-01 /instruments/A user_id:1
2021-01-01 /instruments/B user_id:1
2021-01-02 /instruments/A user_id:2
2021-01-02 /instruments/C user_id:2
2021-01-02 /instruments/D user_id:2
2021-01-03 /instruments/B user_id:1
2021-01-03 /instruments/C user_id:1
2021-01-04 /instruments/A user_id:3
2021-01-04 /instruments/B user_id:3
```

Respective examples of similar stocks with their ranks:

```
# for stock A
B 2
C 1
D 1

# for stock B
A 2
C 1
```

StorageAPI mock

In order to allow you to test your code, we provide you a mock of a generic Storage API.

Of course we want to know exactly which kind of Storage you would use in real life, so please specify it with comments (or provide your own mocks).

Here's an example on how to use it:

```
# for this case I'd use a SQL database with schema (id,foo,bar)
sql_db = StorageAPI()
sql_db.save('id', ['foo', 'bar'])
sql_db.get('id')
```

If your language of choice is not Python, please provide a mock yourself.