FOREX_BE.md 3.12 KB

Foreign Currency BE Exercise

Preparation

Before starting, you will need:

- Git
- Python/Golang dev setup
- Docker for deployment
- docker-compose
- 6 hours of your time

The Exercise

For this exercise, you will be creating a set of APIs to be used by FE developers to develop application that store and display foreign exchange rate for currencies on daily basis.

Please use the following tech stack:

- Python/Golang
- Docker for deployment
- MySQL or PostgreSQL

Use cases

User wants to input daily exchange rate data

The UI will look as follows:

Type	Input
Date	`2018-07-01`
From	`USD`
To	`GBP`
Rate	`0.75709`

`input_field` means input field

User has a list of exchange rates to be tracked

The UI will look as follows:

Type	Input
Date	`2018-07-02`

From	To	Rate	7-day avg
GBP	USD	1.314233	1.316904
USD	GBP	0.7609	0.759366
USD	IDR	14347	14289
JPY	IDR	*insufficient data*	

`input_field` means input field

7 day average here means the average of exchange rate for the last 7 days, including date selected. For example, last 7 days in the example will calculate the rate average from 26 Jun 2018 to 2 Jul 2018.

If the daily data is missing (either to display historical rate, and/or last 7 days rate), please leave it as *insufficient data*

User wants to add an exchange rate to the list

The UI will look as follows:

Add Exchange Rate to track:

Type	Input
From	`USD`
To	`GBP`

`input_field` means input field

User wants to remove an exchange rate from the list

The UI will look as follows:

Remove Exchange Rate to track:

Type	Input
From	`SGD`
To	`IDR`

`input_field` means input field

Evaluation Checklist

As this exercise is a very simple one, the functional correctness of this exercise is secondary. It should be a given that you will be able to get the correct outputs from above. Therefore, to make your work really stand out we look at the following things:

- Code quality & readability: Will any random engineer be able to understand the execution just by briefly scanning through the source code?
- Software design: Does the implementation make full use of classes, objects, functions, abstractions, interfaces, etc.
- Engineering best practices: Does it follow proper architectural patterns (like MVC), and SOLID principles?
- Any automated tests (e2e, integration, unit, etc.)

and *NOT*:

- Fancy UI.

Submission

Once you have completed the exercise, please push the git repository to a host of your choice, preferrably GitHub. Your Dockerfile and code should be sufficient for us to recreate and test your API.

Please submit the following items:

- Git repository for your code (including Dockerfile)
- API documentation (that FE dev is gonna make use)
- Database design documents (DB structure and explanation)