Foreign Exchange Currency - Front End

Preparation

Before starting, you will need:

- Git
- JS dev setup
- Docker

The Exercise

For this exercise, you will be creating a simple app that calculates a converted foreign exchange currency using the Foreign Exchange Rates API https://exchangeratesapi.io/.

Please use the following tech stack:

- Any JS framework you like. Some example include React, Vue, etc -- but you're free to choose your own framework.
- Docker for deployment

The app will consist of:

- A text input specifying the currency input amount of base currency, USD.
  - Initial default value is 10.00.
  - Initial currency may be hardcoded to USD.
- A list with the following values:
  - Target currency -- together with the details
  - Calculated converted amount.
  - The calculated amount should change whenever the input amount changes.
  - Current exchange rate
  - A (-) button to remove the target currency An option to add more currency to the list Upon click, user can input their own currency code via dropdown menu and submit

List of currencies that have to be supported are as follows:

USD CAD IDR GBP CHF SGD INR MYR JPY KRW

You may add more currencies if you'd like to

Example

Using API provided in https://api.exchangeratesapi.io/latest, follow this mockup:

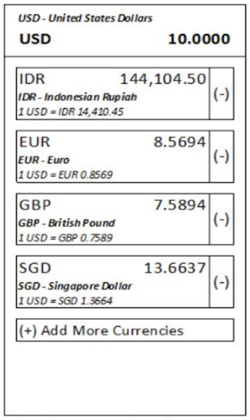

Fig 1: Homepage view of app

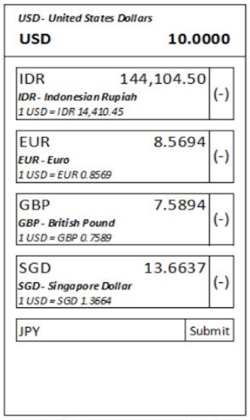

Fig 2: Adding new currency (JPY)

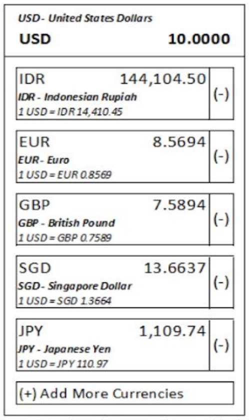

Fig 3: Updated view with JPY added

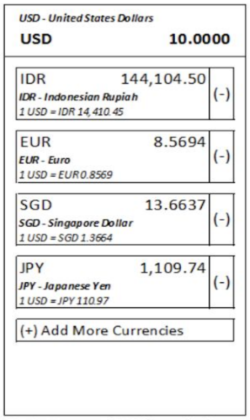

Fig 4: Removed GBP from the list

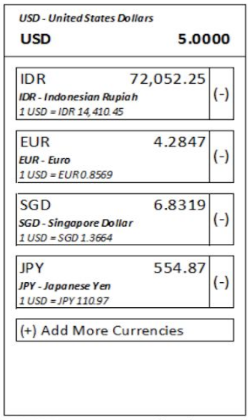

Fig 5: Base currency value changes

You can use any FE UI framework (Bootstrap / Material / Semantic UI) to build upon this mockup.

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.
- Any automated tests (e2e, integration, unit, etc.)

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)
- Explanation on structure of your code, if needed