Flush Backend Take-Home Assignment

Flush allows for a two-way payment through its marketplace of bathroom rentals. It utilizes a RESTful API for its backend to process this transaction through Stripe. The app sends data to the API endpoint, including information about which bathroom listing it's for, the price, the user ID, and the homeowner, or the payment recipient.

For this assignment, you will create a simple version of that application within this repository by coding a simple API endpoint that receives a JSON payload with the user information and returns a transaction response (JSON again) – you don't have to worry about the frontend of the application.

The input

First, the would-be frontend of this application allows the user to check out a bathroom, storing information about the User state and retrieving the listing information from the Firestore API. The example result produces a JSON payload, posted to the application's API endpoint, like this example:

```
{
“userId” :”abcdefghijklmnop”,
“listingId”: “qrstuvqxyz”,
“price”: 3.45,
“currency”: “usd”,
“timestamp”: 9045843494,
“recipient”: “bcdefghijklmnopa”,
}
```

Valid input

- `userId, listingId, Recipeint` (must be a string).
- `price` (must be a float greater than 0).
- `timestamp` (an integer indicating seconds since epoch).
- `currency` (`"usd"` or `"eur"`).

The final price algorithm

The application receives the JSON payload through the API endpoint and transforms it into a transaction based on the information provided by the user. The main thing that changes with the transaction is the price; a local fee must be applied to the price such that if the currency is USD, a local fee would be 3%, and if EUR, a local fee would be 4%. The backend should also check if the price is valid; the price must not fall below 3 USD or 2.8 EUR. If between cents, you should *round up*, as in a ceil operation.

The output

After the calculation is complete, take the final timestamp in seconds since epoch. Don’t worry if it is the same as the input. Considering the data provided above (and a fake timestamp), the application should return the following JSON payload:

```
{
    "Final price": 3.59,
    "Timestamp”: 9045843495
}
```

Criteria

You can choose any technology stack to implement this assignment. Using our stack is not a requirement in the selection process - I will consider exclusively that you build a solid system with an emphasis on code quality, simplicity, readability, maintainability, and reliability.

Be aware that Flush will mainly take into consideration the following evaluation criteria:

- How clean and organized your code is, along with its *commenting*
- How you validity-checked the input arguments
- If you implemented the pricing rule correctly

Other important notes:

- Develop a extensible score calculation engine
- Add to the README file: (1) instructions to run the code; (2) what were the main technical decisions you made; (3) relevant comments about your project
- You must use English in your code and also in your docs

This assignment should be doable in less than one day. I expect you to learn fast, communicate with me, and make decisions regarding its implementation & scope to achieve the expected results on time.

It is not necessary to build the screens a user would interact with, however, as the API is intended to power a user-facing application, I expect the implementation to be as close as possible to what would be necessary in real-life. Consider another developer would get your project/repository to evolve and implement new features from exactly where you stopped.

Submission

Your submission should be private- please create a new private repository and share it with eszabo12.

You will hear back within a week of submitting!