Backend Technical Test

As part of the recruiting process, we ask you to complete this task to show your abilities and knowledge. We want to understand your code standards and practicies.

Description of the task

Implement a RESTful API for tracking IOUs. Four roommates have a habit of borrowing money from each other frequently, and have trouble remembering who owes whom, and how much.

Your task is to implement a simple RESTful API and graphql that receives IOUs as POST requests, and can deliver specified summary information via requests using Django.

Note: We recommend using serializers and generics

API Specification:

User Object

```
{
  "name": "Adam",
  "owes": {
    "Bob": 12.0,
    "Chuck": 4.0,
    "Dan": 9.5
  },
  "owed_by": {
    "Bob": 6.5,
    "Dan": 2.75,
  },
  "balance": "<(total owed by other users) - (total owed to other users)>"
}
```

Methods

DESCRIPTION	HTTP METHOD	URL	PAYLOAD FORMAT	RESPONSE W/O PAYLOAD	RESPONSE W/ PAYLOAD

DESCRIPTION	HTTP METHOD	URL	PAYLOAD FORMAT	RESPONSE W/O PAYLOAD	RESPONSE W/ PAYLOAD
List of Debts from one person to another	GET	/settleup	`{"users":["Adam","Bob"]}`	`{"users": <List of all User objects>}`	`{"users": <List of User objects for <users> (sorted by name)}`
Create user	POST	/add	`{"user":<name of new user (unique)>}`	None	`<User object for new user>`
Create IOU	POST	/iou	`{"lender":<name of lender>,"borrower":<name of borrower>,"amount":5.25, "expiration":timestamp}`	None	`{"users": <updated User objects for <lender> and <borrower> (sorted by name)>}`
Graphql	POST	/expired_iou	`graphql node`	None	`return expired ious`

Result

Please make `/settleup` endpoint return the least amount transactions needed. i.e. If Adam sends $10.00 to Chuck, then Chuck sends $30.00 back to Adam, then the settle up endpoint should say 1 transaction is needed from Adam to Chuck for $20.00. It should calculate the least amount of transactions for any number of users.

Exception messages

Sometimes it is necessary to raise an exception. When you do this, you should include a meaningful error message to indicate what the source of the error is. This makes your code more readable and helps significantly with debugging. Not every exercise will require you to raise an exception, but for those that do, the tests will only pass if you include a message.

To raise a message with an exception, just write it as an argument to the exception type. For example, instead of raise Exception, you should write:

```
raise Exception("Meaningful message indicating the source of the error")
```

Running the tests

To run the tests, run `pytest rest_api_test.py`

Alternatively, you can tell Python to run the pytest module: `python -m pytest rest_api_test.py`

Common `pytest` options

Common pytest options:

- `-v` : enable verbose output
- `-x` : stop running tests on first failure
- `--ff` : run failures from previous test before running other test cases For other options, see python -m pytest -h

`Note:` For unit tests we are expecting you to test all paths, not just happy paths.

What we expect

We are going to evaluate all your choices from API design to deployment. The examples design is not the final design, so spend enough time in every step, not only coding. The test may feel ambiguous at some point because we want you to feel obligated to make design decisions. In real life, you will often find this to be the case. We are going to evaluate these dimensions:

- **Code quality**: We expect clean code and good practices
- **Technology**: Use of paradigms, frameworks, and libraries. Remember to use the right tool for the right problem
- **Creativity**: Don't let the previous instructions limit your choices, be free
- **Organization**: Project structure, versioning, coding standards
- **Documentation**: Anyone should be able to run the app and understand the code (this doesn't mean you need to put comments everywhere).

If you want to stand out by going the extra mile, you could do some of the following:

- Add tests for your code
- Containerize the app
- Deploy the API to cloud environment (GCP (What we use) or AWS)
- Provide API documentation (ideally, auto-generated from code, something like Swagger)
- Propose an architecture design and explain how it should scale in the future

Delivering your solution

Please provide us with a link to your repository and the URL to the running app *if* you deployed it.