Full-Stack Developer Takehome

Task

Develop a simple web application in **Python, Go, or NodeJS** that interacts with GitHub's REST API's and Teradici's public repository for Cloud Access Manager: https://github.com/teradici/deploy.

The following resources may be useful:

- GitHub commits API: https://developer.github.com/v3/repos/commits/
- General GitHub API documentation: https://developer.github.com/v3/

You are required to write the application, Dockerfile, and docker-compose.yaml file that allows the application to build and run without any dependencies other than docker installed.

APIs

The application must provide the following APIs – each route must return valid JSON responses:

/users

Returns the users for everyone that has committed code to https://github.com/teradici/deploy during the period from June 1, 2019 - May 31,2020. The response must be an array of objects as shown below:

```
[
    {
                name: string,
        email: string,
        }
]
```

/most-frequent

Count the number of commits which occurred from June 1, 2019 - May 31, 2020, associated with the author. The response must contain the top 5 committers and be returned in an array of objects with the username and the count of commits. An example response structure can be seen below:

```
[
        {
                   name: string, # author's name
                   commits: integer # number commits
        }
]
```

Caching

The application should utilize caching using redis. Responses from github should be cached for 2 minutes. Please add a redis instance to the docker-compose file as a separate service

Bonus

After the application is running and the tests are in place, as a bonus point, make start and end dates configurable as ISO8601 date parameters provided in request url. Example:

```
/most-frequent?start=2019-09-07&end=2020-09-02
```

With the format of the **start** and **end** being YYYY-MM-DD

Running the application

The application should be built and deployable with docker-compose. The port number for the application should be provided as an environment variable with a default of 8080 in docker-compose.

Building

`docker-compose build`

Running the application

The following command should start the application.

`docker-compose up`

When the application is started, it should bind to the port defined in docker-compose. The APIs should be accessible via API clients by:

- http://localhost:8080/users
- http://localhost:8080/most-frequent

Unit Tests

Add at least one unit test for each route that mocks the response provided by GitHub and covers:

- one succesful response (GitHub responds with 200)
- one failed response where GitHub responds with a 500

The unit tests should run from docker-compose. For example, if the application was written in nodejs the following command should execute the tests:

```
docker-compose run <service-name> npm test
```

What we are looking for

- Clean, maintainable and working code
- Comments if/where appropriate
- Ability to write unit tests and mock external services
- Ability to interact with third party APIs
- Ability to use and deploy in containers
- Readme file describing how to run the application and unit tests

Submitting the Application

The application can be submitted in any of the following:

- A compressed archive (zip, tar.gz) uploaded to a shareable location (eg. google drive)
- A compressed archive (zip, tar.gz) over email
- Checked in to a Github repo

The submission should contain only the source code (and tests) of the application, Readme, Dockerfile, and docker-compose.yaml file.