Description:

This challenge requires implementing a web application, allowing customers to register and maintain their stock portfolios. It should consist of a server interacting with a client using HTTP. **No GUI or website are required.**

Terminology

- Stock - represents a single stock, Identified by its name (a string).
- Stock state - a combination of a stock and its current value.

Startup

Upon startup, the application should read the stock market history from

a file. This file contains all known stocks and their value history.

For example, if Fyber's stock was created with a value of 1, and

changed 5 times over its history, each time raising by 0.2, the file should

contain the following 5 entries:

Fyber, 1.0, 1.1.2018

Fyber, 1.2, 2.1.2018

Fyber, 1.4, 3.1.2018

Fyber, 1.6, 4.1.2018

Fyber, 1.8, 4.1.2018

Fyber, 2.0, 6.1.2018

From that point on, the application should get updated whenever a stock price has changed. For simplicity's sake, you can assume the same file changes whenever such change happens and have the application read the changes from it.

A note to the reader - the entries' format doesn't have to match the one presented here, and can be anything you see fit for solving the challenge.

Exposed API

The application should expose the following endpoints:

- Enabling a **new** client to register their stock portfolio. The request should contain all of the client's stocks alongside their respective amount. A successful response consists of the client's id, which will be used for future calls to the application.

- Enabling an **existing** client to update their stock portfolio, either by sending a new portfolio (in a similar fashion to registering a new one) or by sending a list of already owned stocks and the respective amount of stocks sold/bought. In either case, the request must contain the previously created client id.

- Enabling an **existing** client to check their current portfolio value. The request must contain the previously created client id, and the response is the portfolio's total value. A portfolio's value is simply the sum of all its stock values, where a stock value is its amount multiplied by its value. The application must comply with the following technical requirements:

- User friendly latency. Every request must be responded in reasonable time. We're not setting an exact number, but the user experience should be sane.

- Resilient to crashes. If the application crashes it should be able to restart with all the registered portfolios and historical data

needed for it to function properly.

- Handle edge cases. It must not crash or throw errors when confronted with invalid or unknown inputs (unknown stocks, negative number of stocks, etc...). Instead, a reasonable and relevant error response should be returned to the client.

- Can be build with standard build tools (maven, sbt)

- Write unit and functional tests.

- Expose an additional API allowing the client to ask for simple buying recommendations. A buying recommendation is an id of a stock recommended by the application. A stock can be chosen by 3 strategies: ○ Performance - An already owned stock, who raised the most in value during the last 7 days. ○ Most stable - An already owned stock, with least value fluctuation during the last 7 days. For example, a stock whose max value was 4.5 and min value was 2.5 is considered more stable than a stock whose max was 10 and min was 5. ○ Best - A **not** already owned stock, whose current value is the highest among all stocks.

- Protect the application from attacks by limiting its capacity to support a 10000 concurrent requests. If 10000 request are already being handled by the application, it should reject other incoming ones.