SoFi Software Engineer - Infrastructure challenge!

Overview

The purpose of this coding challenge is to help us better understand your capabilities as an engineer and to give you an exercise which is a bit more reflective of a real world scenario. By allowing you to use the tools that you would use everyday our hope is that this would allow us to see that quality of work that we could expect from you should you join the SoFi team.

The Challenge

This challenge will require you to develop an HTTP endpoint that is capable of receiving (and accepting) several types of HTTP verbs with JSON payloads. The expectation is that any code produced should represent your expectation of high quality code. It's expected that this should be completed within 24 hours of starting the challenge. However, this is typically completed in 4-5 hours.

The specific functional requirements for the challenge can be found below:

Functional Requirements

- The service must be capable of accepting a JSON message as an HTTP POST. The structure of this message, shown in Listing 1, represents a transaction that has occurred within the system.
- The service must provide an operation that will return to the caller the three most frequently visited merchants for a caller specified user id. This should be made available to the caller as an HTTP GET message and the response should be a JSON object. *NOTE: If an insufficient number of transactions have been processed for the user (<5 transactions) then an error should be returned to the caller.*
- The service should store the transactional data in-memory using a set of data structures chosen by you. These should be optimized for minimal memory consumption and quick lookup time. The service should not leverage any 3rd party tool for storing or querying of the transactional data.
- The service must defend itself and it should not accept malformed requests nor should it accept HTTP verbs that are not explicitly intended to be supported.
- You should deliver the code, build scripts, along with a script that can be used to run (and validate) your service to your recruiter. This should not only start your service but also reproduce the expected answers shown below. NOTE: This can be delivered to your recruiter as a zip file, git repo, whichever is easiest for you.
- [Extra Credit] The service should provide some basic metrics around its performance in order for operators to know that the service is performing as expected.

Listing 1 - Transaction JSON

```
{
  "user-id" : 1,
  "merchant" : "Bartell Drugs",
  "price" : "5.78",
  "purchase-date" : "2019-01-01T22:45:40",
  "tx-id" : 100
}
```

Implementation Constraints

You may use any 3rd party library you like in the completion of this exercise unless it's restriction has been previously called out. The use of a 3rd party library for storing/querying transactions is one example of this.

Some notable examples of useful libraries to use here include GSON or Jackson for JSON parsing. There are also numerous toolkits and libraries that are available for creating self-hosting HTTP services.

Sample dataset w/results

In order to make it easier to validate your code, a sample dataset has been provided The "coding_challenge_data.csv" csv file contains test data that will correlate with the expected results outlined below. Use this to validate your findings.

Here are some example validated outputs using the dataset provided:

User ID	Result
1	Bartell Drugs, Flying Pie Pizza, Albertsons
3	Error - Too few transactions
2	Error - Too few transactions