GlobalWebIndex Engineering Challenge

Introduction

This challenge is designed to give you the opportunity to demonstrate your abilities as a software engineer and specifically your knowledge of the Go language.

On the surface the challenge is trivial to solve, however you should choose to add features or capabilities which you feel demonstrate your skills and knowledge the best. For example, you could choose to optimise for performance and concurrency, you could choose to add a robust security layer or ensure your application is highly available. Or all of these.

Of course, usually we would choose to solve any given requirement with the simplest possible solution, however that is not the spirit of this challenge.

Challenge

In GWI we want our users to have a list of assets, things that favourite or “star” so that they have them in their frontpage dashboard. An asset can be one the following

1) Chart (that has a small title, axes titles and data)
2) Insight (a small piece of text that provides some insight into a topic: e.g. *40% of millenials spend more than 3hours on social media daily*)
3) Audience (which is a series of characteristics, for that exercise lets focus on gender (Male, Female), birth country, age groups, hours spent on social media, number of purchases last month)
    a) e.g. Males from 24-35 that spent more than 3hours on social media daily.

Build a web server which has some endpoint to receive a user id and return a list of all the user’s assets. Also we want endpoints that would add an asset to favourite, remove it, or edit its description. Assets obviously can share some common attributes (like their description) but they also have completely different structure and data. It’s up to you to decide the structure and we are not looking for something overly complex here (especially for the cases of audiences). There is no need to have/deploy/create an actual database although we would like to discuss about storage options and data representations.

Note that users have no limit on how many assets they want on their favourites so your service will need to provide a reasonable response time.

A working server application with functional API is required, along with a clear readme.md. Useful and passing tests would be also be viewed favourably

Submission

Please create a public or private Github repo as you prefer to hold your submission. When ready please add *thunt-* and *gnikolaropoulos* as collaborators to the project. It is appreciated, though not required, if a Dockerfile is included.

Good luck, potential colleague!