Andrew Gunn

Fitbit API Dashboard Widget & Data Input App

Project: Vue Fitbit

Proposal: Create a web application interface for a user's fitbit device(s) that have a dashboard-like view of metrics like: heart rate, steps, calories, zone minutes, floors, sleep data, food, weight and also provide an input to that dataset via API.

Repository: https://github.com/amg262/vue-fitbit

Google Doc: https://docs.google.com/document/vue-fitbit

How I'll do this

The **Fitbit API** will drive all the data and is the main engine driving my application. This requires a Fitbit Developer account and Fitbit Developer Application, both of which I have already set up, tested out, and connected to with success in pulling down data from my device.

The target audience is **myself, initially.** I can only reliably count on my data at this time, however I have applied for use of the anonymous daily data fitbit allows in some cases. It will take 5-7 days for my case to be determined.

At a certain point, I may integrate an ability for a social login or text field for a fitbit user ID but I cannot tell at this time if that is possible or feasible. I think proceeding without worrying about that until a later time is best.

Vue.js is the component and main coding framework I will work within, and main of my

There are 2 main parts to this application, the tile-widget dashboard view of the different health metrics which will utilize a D3.js based library, which I have narrowed down into:

Vue-Chart.js looks like what I may go with.

1. C3.js (favorite)
2. NV3M.js
3. Morris.js
4. Plot.ly
5. Highcharts

C3.js is what I am starting with as I believe the combination of lightweight, dozen or so charts to choose from, interactive capability of charts, JSON form of inputting data into the chart, and it seems the least complicated compared to **D3.js** which is so raw and complicated. Other dependencies include Stack.gl which may be needed.

I will utilize many different types of graphs, charts, maps etc for different kinds of health data metrics and will make them interactive, meaning hovering over different plot points will have a tooltip popup showing details for that specific point. Clickability is also true for these as well.

Bootstrap v5 is my front-end framework no question. I did however consider Boostrap briefly. I decided against Material UI because of my familiarity and comfort with BS, and the main reason I choose to not use Bootstrap-vue is the lack of intellisense and autocomplete in Webstorm. It's a sad cop out, I know.

Bootstrap v4 is the backup plan if any of my other dependencies aren't compatible with 5 yet, which is very possible.

This is designed first as a **dashboard backend** style, similar in look to the admin login page of a WordPress site. The main content of that will include cards and tiles. Within those cards and tiles will be the charts. It will be columnar in structure, responsive, but also not look like a bare bones BS implementation. I plan on choosing a Coolors color scheme that looks good and unique and using that as the basis of my colors.

The page will have a **sidebar admin menu** pinned open on the left side of the page with different links to the front end pages and different #element-id hotlinks to different chart elements on the page.

There will also be the user input **front-end** style page that will have a different look and feel completely to the backend. This will have a tree view, collapsible, panel (or etc) splitting of the long list of different health data categories to view, input, or delete data from. Each of these will be a different component as they're all fundamentally different. I will list components below.

SASS version of bootstrap is my plan starting out but I've been running into NPM issues with the SASS loader. Npm install has been failing at times when I download a project from somewhere and try to run it. I'm concerned that requiring sass will cause issues for/if when you need to download and run the project on your machine there will be errors on npm install or serve.

If this problem seems like more trouble than it's worth, **the CSS** version of Bootstrap will suffice.

Components (Frontend): Each work with (xxxx) will mean a different category (directory) of components which will all have the corresponding component to the right of -
Each word separated/by/slashes will 100% mean a new component.
FE suffix - Frontend component to input data.

Andrew Gunn

BE means backend metric of graphic

Activity FE - Create (Delete) Activity Goal/Log/Favorite Activity
Activity BE - Get Activity Goals/Activity log/daily activity summary/etc….
Activity (Body) (Heart) (Nutrition) Time Series BE - Get activity (body) (heart) (nutrition) time series by date/date range
Body FE - Create (Delete) body fat goal/body fat log/weight goal/weight log
Body BE - Get body goals/fat log/weight log
Devices - Create (Delete) (Get) (Update) alarm
Nutrition - Theres a list of 3 dozen regarding Create Food, Get Food, Get meal, things like that
Sleep - Going to be doing a lot with this one and may focus most of my attention towards Create (Delete) (Get) sleep goal/log/by date/date range, Get sleep log list (this will have a huge chart going through 1.5 years of sleep data and analyzing/graphic data
User - Get (Update) Bades/Profile

Other components include: admin menu, widget, chart type, user menu, header, footer, user ID text field, form.

Note: This will be my list to start with, but there are easily dozens of components, I will initially focus on a list of 12 important ones and add more when I have a framework or workflow to easily replicate more.

I **know DRY is crucial**, I will need more planning on not doing this because some of what I stated above implies I will be repeating myself a lot, I assure you I won't I assure you I won't.

Extra Credit

AJAX is going to be utilized to some extent to provide an auto update of the different widgets.

Firebase may be used for storing the API key, login credentials and retrieving those from DB. I also might store each widget's data when I retrieve them.

Security will be addressed to some extent, it's too early to tell now but I will get some aspects covered.

Routing will be used to some extent, as I have several pages I will be switching back and forth to so there is a practical use case.