Walmart Mexico Code Challenge

Code Challenge

Introduction:

Believe it or not, Walmart Mexico is a growing startup-like division within Walmart that is looking for fullstack front-end developers like yourself to help it scale to a be a billion dollar revenue division.

This code challenge is meant to mimic the type work we do each day as front-end/fullstack developers in the Walmart Mexico division. Everyday, we ask developers such as yourself to take ownership of the implementation and management of front-end features. This requires understanding of front-end architecture, data models, how-the-feature-works-end-2-end, in addition to writing the feature in html/css via React or other technologies.

Through this challenge, we are looking for signals that indicate that you would be able to handle the unique and interesting engineering challenges we have that take more than just writing of front-end code to fit a UI mock.

Our techstack is React, Redux, Node (bff layer), Java (microservice layer). We don't expect mastery - we prefer fullstack understanding and the ability to quickly learn what you don't know.

Code Challenge

Instructions:

- Fork the code challenge repo to your personal Github account
- Once you are complete with the tasks of this challenge, push the code to your repo and send the link to the recruiter to forward to the hiring team for review.
- Here are the tasks to complete using the challenge repo:

1) Configure the server so it serves the static assets (index.html etc.) from the *public* folder

2) Make sure the app uses the favicon.png file in the *public* folder

3) Modify the GET /users/age endpoint API so that the *itemToLookup* value in the function *getListOfAgesOfUsersWithHandler* can be variably specified by the front-end application calling the endpoint and not hardcoded as ‘carrot’ like it currently is.

Code Challenge

4) In the UI design mocks in the following slides, for the *Age Demographic of Users With* table, the backend needs to return a data structure that has an age group along with and count of the number of users with that item that belong to that age group. The front-end should not do algorithmic processing of backend data. Therefore, modify both the *getListOfAgesOfUsersWith* and the *getListOfAgesOfUsersWithHandler* functions so that they work together to return that data structure to the front-end. Use the *getUsers* and *getUsersHandler* as an example of how to write these types of methods.

5) Improve both *getUsersHandler* and *getListOfAgesOfUsersWithHandler* so that if an error is thrown by the *getUsers* or *getListOfAgesOfUsersWith* functions, the server does not crash.

6) Implement the UI design mocks in the following slides. Use your choice of front-end technology. You DO NOT have to follow the look or layout of the UI components. You MUST have the functionality shown in the mocks. Make the UI look nice :) We used Bootstrap in the mocks but you can use whatever else.

When you are done, the hiring team should be able to get your code up and running by these 3 commands:

- git clone your completed code challenge repo and cd into the directory
- then run -> npm install
- then run -> npm start

Code Challenge

We know that you may have questions and typically, at work, those questions are resolved when receiving mocks or during sprint-planning. But since this is an interview, make your best guess to unblock yourself and keep on moving. Use comments in code if needed to convey your thoughts. In our division, this type of problem comes up every day and you must be able to make good judgements and be able to proceed even with the lack of clarity.

Also… fix any bugs you find along the way. Not sure if we left some in there just like in real life :P

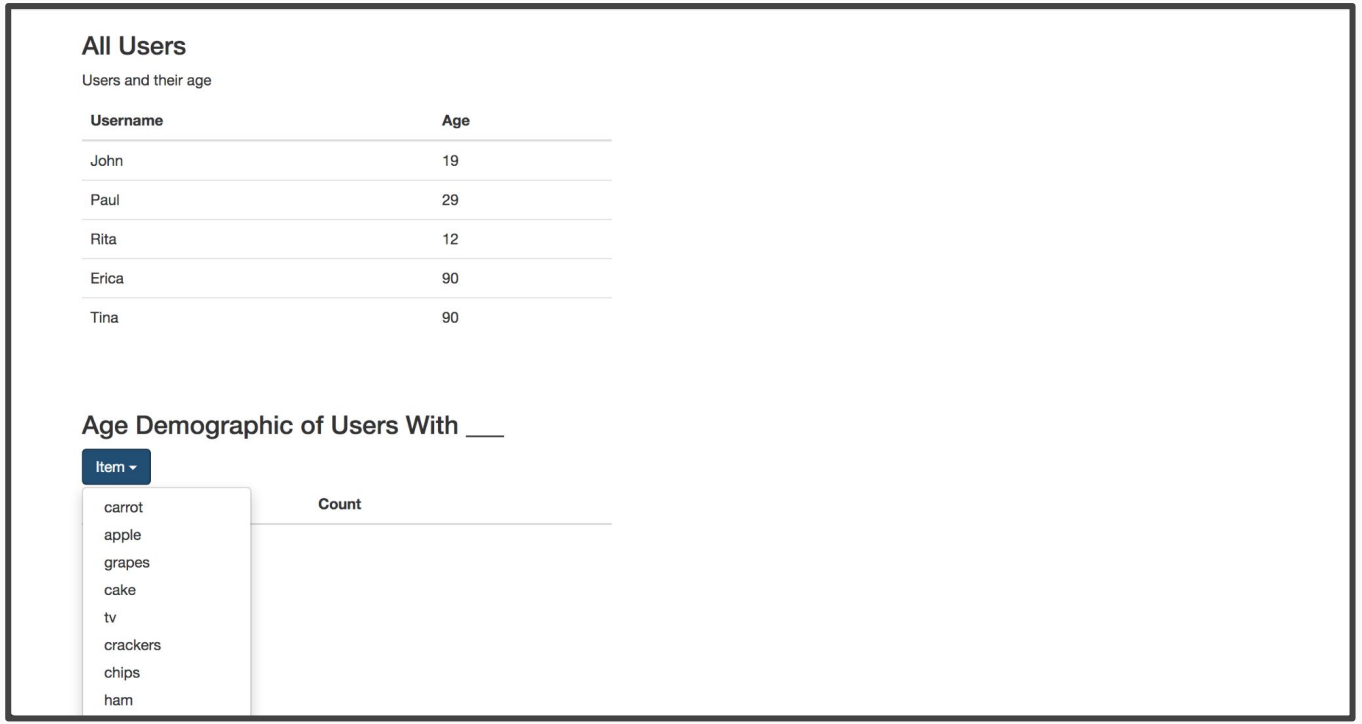
Good luck! We look forward to reading your implementation.

UI Design

When you land on the / path, this is the view you should see

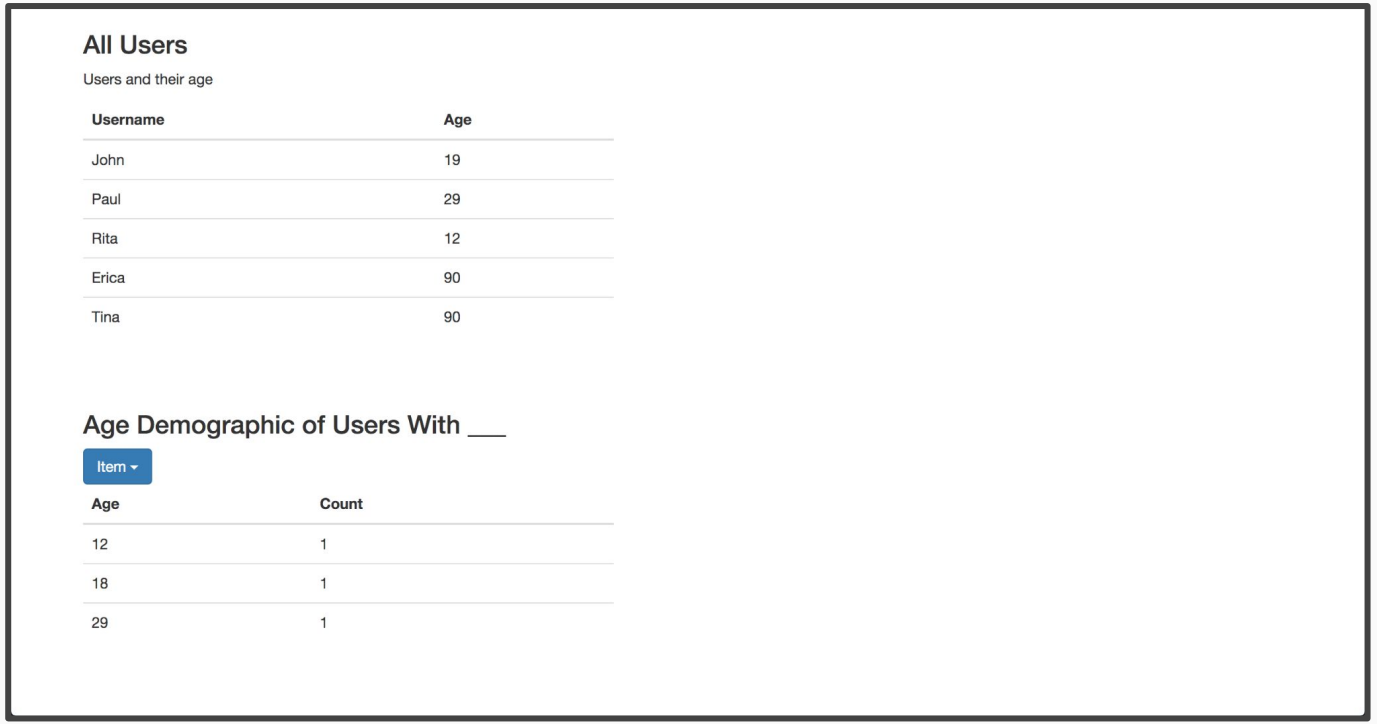
UI Design

When you click on the Item dropdown, you should see all possible items in an unordered list



UI Design

When you select “cake”, you should see the age demographic and the count of each age. The front-end should just use the backend response as-is and render it without additional logic applied to the response.



UI Design

When you select “tv”, you should see the age demographic and the count of each age. The front-end should just use the backend response as-is and render it without additional logic applied to the response.