Tech Tasks for the role of Portal Developer at SABS.

Here are two tech tasks for you to complete. One is a complex data manipulation task and the other is a React App. Don't worry if you don't complete these to working perfection; just provide plenty of comment showing the intention of each step in the code.

If the requirements are unclear (this is not uncommon day to day) or you have any questions while you are progressing, please do not hesitate to ask me (andrew.hardy@sabstt.com), feel free to share code if it helps. If you are invited for interview we may talk through your solutions in more detail.

The aim of the tasks is twofold.

- To see how you discuss and clarify requirements
- To see how you think through a complex data transformation problem.

Thanks,

Andrew

andrew.hardy@sabstt.com

NodeJS Data Manipulation Programming Task.

We'd like you to write a nodejs program that imports data from a given csv file transforms the data and re-writes out the data to an output csv

More than completing to working perfection, the point of the task is to assess how you get to understand and think through a complex data manipulation / transformation problem, so add comment above the code at each distinct step. It's not really a task about reading and writing csv files, so we will give you help with that.

I can explain the task in more detail if needed, so feel completely free to ask any questions at all before you start or while you are doing the task. Once complete email me just the code that performs the task.

The Task.

You should have two example csv files and input file, task-data.csv, and an output file. This should hopefully make the instructions clearer but feel free to ask. Alternatively feel free to create your own input csv file sufficient to the task in principle.

The value in the input file in the column in the CSV with the heading "ColInQuestion", is a semi-colon separated list, representing one or more items picked from a list.

- load the CSV
- Identify the column in question and use the same text as its list values to construct a unique set of CSV column headers additional to the current ones.
- Add these to the header array (item 0 in csvdata.data below)
- After this set 2 (not picked) or 1 (picked) as column values for these new columns for each row (remaining items in the csvdata.data array) according the items present in the value list in the column in question.
- Write out the new data to an output CSV

Snippets of code you 'may' find useful.

// https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference

// NodeJS Array features like spread, split, splice, Array constructor (new Array(n)), etc may possibly be useful

const fs = require("fs");

const Papa = require("papaparse");

fs.readFile("C:/Users/admin/dev/my-api/data.csv", (error, fileData) => {

...

...

// the csv file musn't end with a <CR>. i.e. empty row

csvdata = Papa.parse(fileDataString);

// csv data will be of this form

// {

// data [[], [], []] // i.e. an array of arrays - root array is the rows, top row is the headers, the rest the values - each inner array the column values

// error //

// meta-data //

// }

...

...

// I found this useful

// const newHeaders = new Set();

...

...

// Convert back to CSV - transformeddata is the same object form as csvdata

var newCSV = Papa.unparse(transformeddata);

fs.writeFileSync("C:/Users/admin/dev/my-api/output.csv", newCSV);

});

<u>React App Task</u>

The aim of this task is to get a simple flavour of where you are at with React. When you are done zip up the repo and email it to me.

The task.

This is a more flexible task. Feel free to ask questions, but otherwise, within the spec you have freer reign. Without changing the base requirements, feel free to add more features demonstrating other aspects of your React knowledge.

Develop a React App with a parent component and two child components. Use function based components and Hooks if you know them. One child component should present a list of items; you can first populate the items from hard coded data. The other child component should in some way reflect the currently selected item in the list and provide some dummy actions to take on that item, show that the action has been taken on that item using some placeholder behaviour (if you feel confident, the action could update the state and re-render the list) and then clear the selection. The Selected item in the list should differ in its presentation in some way from the other items.

If you have worked with Redux, if you like you could try implementing the same thing but assume that any data should be global and implement it in Redux.