Coding Assignment

Getting started

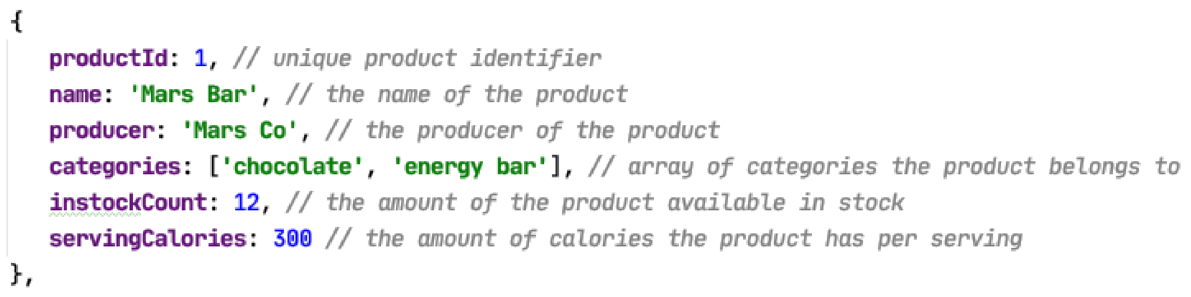
The entire coding question will be coded in Javascript and Node.js. You will be provided with a zipped file that contains the starter kit code you need.

Please follow these steps to ensure you're ready to begin:

1. Ensure you have node version **v14.15.4** installed.
2. Unzip the file you have received for the coding challenge.
3. Within the terminal in the directory run: `npm install.`
4. Then run `node index.js`. If you see the output 'It works!', then you are ready to start working on the challenge.

Your assignment

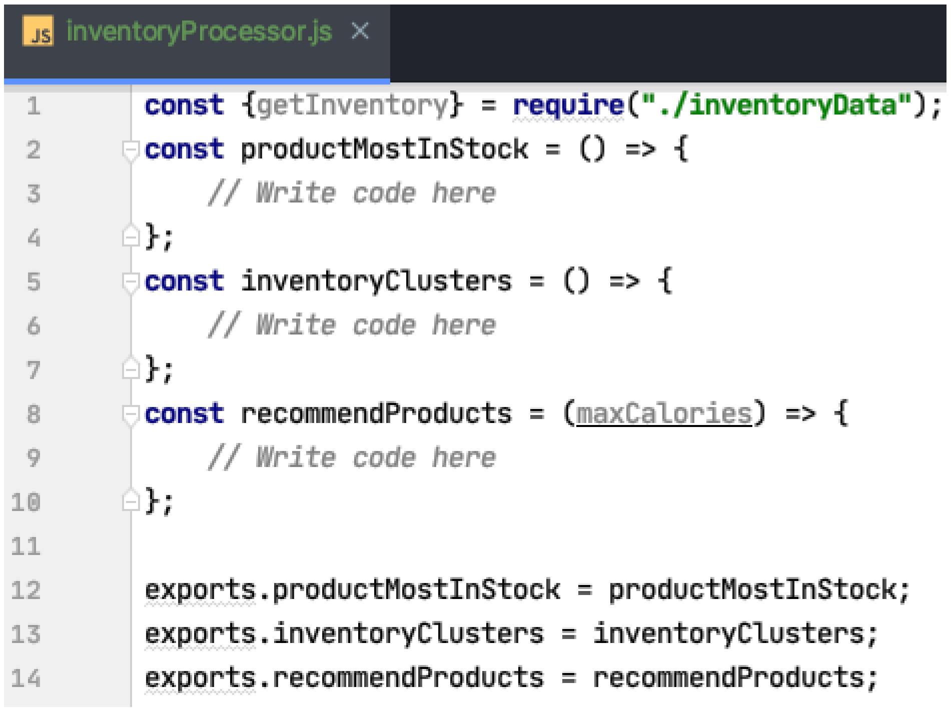
Suppose you have access to a list of food products in an inventory. You can obtain the list of products by calling the getInventory function within the inventoryData.js module. An individual product in the inventory looks like the following:



```
{
   productId: 1, // unique product identifier
   name: 'Mars Bar', // the name of the product
   producer: 'Mars Co', // the producer of the product
   categories: ['chocolate', 'energy bar'], // array of categories the product belongs to
   instockCount: 12, // the amount of the product available in stock
   servingCalories: 300 // the amount of calories the product has per serving
},
```

Note that a product can have one or more categories

All the code you will write will be in the inventoryProcessor.js. You will not change any other file in the codebase.



inventoryProcessor.js

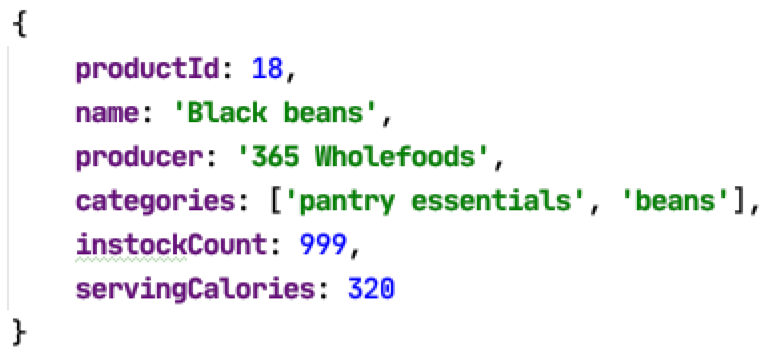
```
const {getInventory} = require("./inventoryData");
const productMostInStock = () => {
    // Write code here
};
const inventoryClusters = () => {
    // Write code here
};
const recommendProducts = (maxCalories) => {
    // Write code here
};

exports.productMostInStock = productMostInStock;
exports.inventoryClusters = inventoryClusters;
exports.recommendProducts = recommendProducts;
```

Important note: You can follow the test-driven approach and run the unit tests we have provided for you by running npm run test. When you first receive the question package, the tests fail and when you are done coding, the tests should all pass. We have set up AVA as the test runner for this project. It will be automatically installed when you run npm install.

Task #1

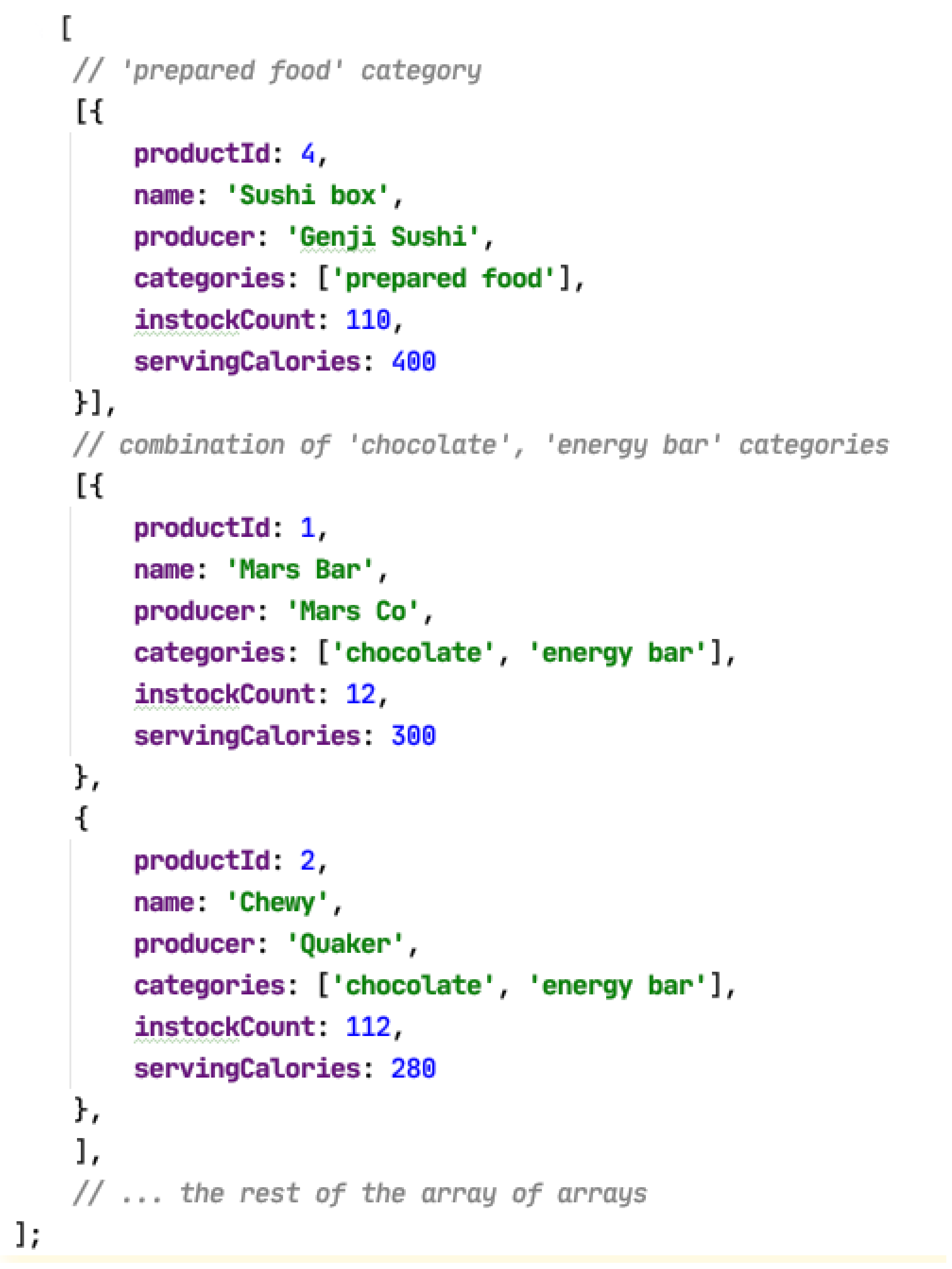
Write a function called productMostInStock that will return the product object that we have the most of in stock. For instance, if of all products in our inventory, we have the most 'Black beans', the function should return the following:



```
{
    productId: 18,
    name: 'Black beans',
    producer: '365 Wholefoods',
    categories: ['pantry essentials', 'beans'],
    instockCount: 999,
    servingCalories: 320
}
```

Task #2

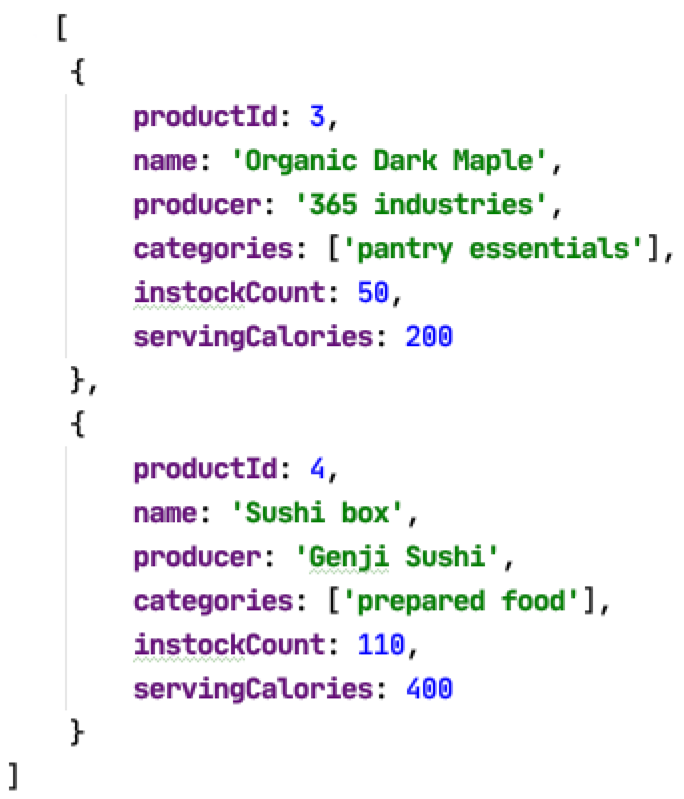
Write a function called inventoryClusters that returns products grouped by their categories. The function will return an **array of arrays**. Each sub-array shall contain products that have the *exact* same categories (i.e. the names of the categories and their total number must match.) Here is an example of the expected output:



```
    [
     // 'prepared food' category
     [{
          productId: 4,
          name: 'Sushi box',
          producer: 'Genji Sushi',
          categories: ['prepared food'],
          instockCount: 110,
          servingCalories: 400
     }],
     // combination of 'chocolate', 'energy bar' categories
     [{
          productId: 1,
          name: 'Mars Bar',
          producer: 'Mars Co',
          categories: ['chocolate', 'energy bar'],
          instockCount: 12,
          servingCalories: 300
     },
     {
          productId: 2,
          name: 'Chewy',
          producer: 'Quaker',
          categories: ['chocolate', 'energy bar'],
          instockCount: 112,
          servingCalories: 280
     },
     ],
     // ... the rest of the array of arrays
];
```

Task #3

Write a function called recommendProducts that takes maxServingCalories as input and returns an array of ***random*** products that have an aggregate number of calories that is equal or less than than maxServingCalories (i.e. every time the recommendProducts function is called it should produce different random results). maxServingCalories input cannot be higher than 1,500. Also, each product can only be included in the output array once . For instance, if the function is called as recommendProducts(600), one possible result could be the following:



```
    [
     {
         productId: 3,
         name: 'Organic Dark Maple',
         producer: '365 industries',
         categories: ['pantry essentials'],
         instockCount: 50,
         servingCalories: 200
     },
     {
         productId: 4,
         name: 'Sushi box',
         producer: 'Genji Sushi',
         categories: ['prepared food'],
         instockCount: 110,
         servingCalories: 400
     }
]
```

Optimization bonus: The closer the sum of the calories are to 1500, the better

Evaluation of the coding question

Your submission will be evaluated on the following criteria:

1. All the unit tests in the test.js file pass and the algorithm written actually solves the problems introduced (i.e. running npm run test in the command line will return 4 tests passed)
2. Readability and simplicity of your code. The simpler you solve the problem the better!
3. Time and spatial complexity of the algorithm
4. Usage of modern javascript syntax (ES7+)

Design Assignment

Overview

You have been approached to build a News Notification System with the following requirements:

1. Each news article will have multiple tags.

2. Tags can be further divided into sub-tags. For instance, there may be a *Politics* tag. That tag itself may have the sub-tags of *American Politics, Canadian Politics,* and so on.
3. A user is able to subscribe to tags that they find interesting. **When an article that has the tags the user has subscribed to is published or updated, they should be notified**.
4. If a user subscribes to a specific tag, then by default the user is subscribed to its sub-tags also. For instance, if a user subscribes to *Politics*, they are automatically subscribed to *American Politics*, *Canadian Politics*, and so on. They may, however, subscribe to individual sub-tags too if they want. A user, for instance, could subscribe to just *Canadian Politics*.
5. The sub-tags attached to a specific tag may change from time to time. For instance, we may add a sub-tag for *British Politics* that is attached to the *Politics tag.* If a new sub-tag is attached to a tag, the user should be notified about any articles that have that sub-tag too.

Your assignment

1. What are the questions you would ask the Product Manager on this project to better understand how to architect the system?
2. What are some of the things you will consider when designing this system from a technical perspective?
3. (optional) Please provide a diagram and architectural overview for a system design that meets the above requirements.
4. (optional) Please provide any supporting documentation that explains any assumption you've made and helps to justify the decisions you made.

Submitting your response

Please email us the following files

- For the coding question, send a copy of the inventoryProcessor.js containing your solutions
- For the design question, send a copy of a document containing your answers and system diagram (optional.)