Exercise 1:

Please use Spring Boot / Java 8

A wine is made up from grapes of different sources after being blended at the winery. These grapes have different properties such as year, variety (e.g. Chardonnay, Pinot Noir) and region (e.g. Yarra Valley, Mornington Peninsula, Macedon Ranges, etc). When bottling a wine, the winery needs to know the percentage breakdown of these properties so they know what they can legally claim on the wine's label.

Your first mission is to create some simple REST APIs that return some JSON data that describes a breakdown of the TOTAL percentage of year, variety, region and year + variety information for a specific wine, ordered from highest percentage to lowest. The four endpoints are:

- **/api/breakdown/year/{lotCode}** - a breakdown of the TOTAL percentage for each unique year value
- **/api/breakdown/variety/{lotCode}** - a breakdown of the TOTAL percentage for each unique variety value
- **/api/breakdown/region/{lotCode}** - a breakdown of the TOTAL percentage for each unique region value
- **/api/breakdown/year-variety/{lotCode}** - a breakdown of the TOTAL percentage for each unique combination of year and variety

We need an implementation that provides a simple data structure for the caller to use. Each of the mentioned endpoints will need to take a parameter for the lot code of the particular wine details (as provided in the JSON sample files).

For example, in /api/breakdown/year/11YVCHAR001 (for the sample data provided in 11YVCHAR001.json) we want to see something like:

```
{
    breakDownType: "year",
    breakdown: [
        {
            percentage: "85",
            key: "2011"
        },
        {
            percentage: "15",
            key: "2010"
        }
    ]
}
```

... and so on for the other endpoints. Each endpoint returns the percentages of the unique properties and should be **sorted** from highest percentage to lowest.

Please use Spring Boot / Java 8. Your solution should be set up using mvn so that it can be easily run from the command line. Include a README.md file for any instructions to run (the fewer the better!).

Exercise 2:

Build a small ReactJS application that allows a user to search for a wine, then display its details.

Use the following Figma design to create the user interface:

https://www.figma.com/file/q813iaYNn4unQQXbBPngal/Code-Challenge-Wine-Viewer?node-id=1%3A1065

(please note: you will need to create a free figma account in order to inspect the elements in the designs to view the CSS properties).

Build on the API you created in **Exercise 1** to add an endpoint to support searching for a wine by the lot code or description. Also add an endpoint that will return the wine details for the search result clicked/tapped by the user.

As per the Figma design, implement this search as an auto-complete (results appear as you type).

When an item from the list is clicked/tapped, navigate to a view showing the basic details of a wine, as well as allowing the user to see the Year, Variety, Region and Year + Variety breakdown (the client should fetch these details on demand from the API created in Exercise 1).

Summary of what the data represents for each wine (in order of importance):

Lot code (text, usually 10-12 characters) [reference code used throughout a wine's production]

Description (text, usually about 40-60 characters) [human readable description of the wine]

Volume (floating point number, but rounded to zero places) [how much wine in litres there is of this wine]

Tank (text, usually about 5-6 characters) [the code of the physical tank that the wine is currently in]

Product state (text, usually 15-20 characters) [the state that the wine is in... changes throughout production]

Owner (text, usually about 15-20 characters) [who the owner of the wine is]

As per the design, allow the user to choose the type of composition breakdown they are interested in seeing (Year, Variety, Region or combined Year & Variety). The composition breakdowns can get quite long in some cases. When having a glance at the data a user will always want to see the compositions in order of the percentage they make up (highest first). Assume that if the composition breakdown is long that a user would probably only want to see the first 5 elements, but would expect to have an option to see more details (it's okay for the client to fetch all the data from the API in one call, but we don't want to overwhelm the user by displaying it all unless they have explicitly asked to "Show more". Make sure your implementation of "Show more" works this way).

You'll note in the design that there is an Edit button in the top right corner. For the purpose of this exercise, this can just display a message indicating that it has been clicked.

Your solution should be set up using npm or yarn so that it can be easily run. Please include any instructions in a README.md file.

The accompanying zip file includes some JSON files showing some wine details. Your search API should be able to search through this data to return matching results. The endpoint to retrieve the wine details should take the lot code of the wine as a parameter to return the relevant wine's details.

Look at the data in the JSON files to give you an idea of any potential edge cases that may have an impact on the user interface. Assume that when I am looking at your submission, I will be viewing each of the wine details as provided in the sample JSON files.

Please submit back to me via a GitHub / GitLab repo link.

Good luck!