Backend API Technical Challenge

We'd like to get to know you better, so we've prepared a technical task that will tell us more about you. You can find more information about it below.

If you have any questions or you don't fully understand something, do not hesitate to ask us for clarification. Also, we are always looking to improve our technical test so feel free to send us any feedback on the exercise.

Scenario

Gousto's technical infrastructure includes an API Gateway that offers a number of operations related to recipes. The recipes are very detailed, containing information such as cuisine, customer ratings & comments, stock levels and diet types.

Your task is to design, develop and deliver to us your version of a set of recipe operations. The solution should meet our functional and nonfunctional requirements below.

Functional Requirements

Use Case 1: As an API client I want to see a recipe's details

> **Given** that I am an API client
>
> **When** I fetch a recipe by ID
>
> **Then** I can see recipe fields

Use Case 2: As an API client I want to see a paginated list of recipes by cuisine

> **Given** that I am an API client
>
> **When** I fetch a recipe by cuisine

Then I can see a list of recipes

And the list is split into paginated results with 10 recipes per page

And each recipe has to contain only the fields ID, title and description

Use Case 3: As an API client I want to update one or more recipe's fields

Given that I am an API client

When I update one or more recipes fields

Then I can see the updated recipe fields

No-Functional Requirements

- The service should provide a set of RESTful JSON based routes API and you should not include any client code as HTML or JavaScript;
- The service must be built using a modern web application framework like Lumen, Ruby on Rails, Django, etc;
- The code should be "production ready" - that means:
    - it should run;
    - satisfy the requirements (functional and non-functional);
    - be stable;
    - be readable;
    - be maintainable;
    - and have automated tests (preferable following TDD);
- The service should use the accompanying CSV (in the zip you downloaded) as the primary data source, which can be loaded into memory;
    - Please don't use a database, though SQLite would be acceptable;
    - If it helps feel free to generate additional test data based on the scheme;
- In order to understand your solution, we would like to see the following sections in the Readme:
    - How to use: details on how to use your solution (how to configure the local environment, how to run tests and how to run the local server);
    - A list of missing functional requirements (if any), and an explanation on why you didn't complete them;
    - Possible improvements/functionality: anything that you wish you could've added if you had more time;
    - Anything else you think is relevant to your solution.
- Your completed exercise should be submitted via public **Git Repository**(Github/ bitbucket/ GitLab/ etc);