Project and Assignment 2: Kitchen Buddy

The goal of this project is to develop an application to track kitchen ingredients, so that you are:

- less likely to have some go to waste
- more likely to buy what is missing

The project can feature a variety of extensions, which will form the basis of how to extend the assignment into the final project. Let's look at the main functionality first.

Assignment 2: Basic functionality

For the first iteration, we are interested in the most basic functionality, which revolves around:

- efficiently entering kitchen ingredients into the application.
- listing the available ingredients according to various basic queries.

Data entry of kitchen ingredients

The application should make data entry as efficient as possible. The application's UI should feature bottom tabs, with a first screen dedicated to entering new ingredients in the system. When entering ingredients, the user has to fill the following data:

- The name of the ingredient
- Select the category of the ingredient (e.g, fruit, vegetable, dairy, fish, meat, liquid, etc)
- Select the location of the ingredient (e.g, fridge, freezer, pantry)
- Select the confection type (e.g., fresh, canned, frozen, cured)
- Enter the expiration date. This should either be the date on the ingredient's packaging (an exact date can be entered), or an estimate of it, such as for ingredients that only have a confection date (e.g. some meats), or that have no such date (e.g., some vegetables). For these cases, common estimates can be used (e.g., 1 week from now, 10 days from now, a month from now), and edited.

Once an item is added, the screen resets and awaits the addition of further ingredients. Of all the fields above, only the ingredient name is mandatory.

Querying ingredients

The application provides additional tabs to query the ingredients in the system. When a list of ingredients is presented, all the ingredients there can be edited, which brings the user to a screen similar to the data entry one, but with the existing values filled. All the list of ingredients also support textual search.

Queries that the app should support are:

- Show a list of all the ingredients that are going to expire soon (with a control saying how soon)
- Show a list of ingredients that are missing data (so that this data can be added later on)
- The most recently added items
- Items that are in the same location (fridge, pantry ...)
- Items of the same food category or confection type

Of these queries, "ingredients expiring soon" should be in a dedicated tab. The other queries should be accessible from a third tab.

Handing in assignment 2

As with assignment 1, you are expected, beyond functionality to conform to the style of writing code seen in the class. Also, assignment 2 is not mandatory, but if you hand it in you may get bonus points on your grade.

The due date for the assignment is **Sunday, April 30th**. To hand in the assignment, you can simply provide the link to the expo project that you are working on. This assignment is individual.

Final project

The final project will expand on assignment 2. The deadline for the final project will be **Sunday, June 4**. This project can be done in groups, but the scope of the project will grow when more people are involved. A pair will also implement a grocery list extension. **If you plan to be in a team, you must inform me.**

Better ingredient handling (everyone)

The application is extended with additional functionality and ingredient data.

- Persistence: the application's data persists accross execution.
- Ripeness: fresh ingredients have a ripeness or maturity status (e.g., green, ripe/mature, advanced, too ripe). This maturity status can be edited, and the date when it was edited is stored (something ripe a week ago might not be good anymore!).
- Frozen: fresh ingredients can be frozen. This also extend their expiration date to be at least 6 months.
- Open: some ingredients last only a short time after being opened (e.g., a yogurt) When an ingredient is set as open, their expiry date can be changed to account for this.
- Barcode scanning: the application uses Expo's BarCodeScanner API to read barcodes. It can then query OpenFoodFacts, and if succesful, it retrieves data about the item automatically.
- Brand: some items can have brands, in addition to names.

Additional queries are implemented:

- Items that have a ripeness status need to be checked regularly. If the last check was more than 3 days ago, they are added.
- Items that are ripe, and open items, are added to the "expiring soon" query; items that are frozen are removed from it (unless their new expiry date is coming up)

Grocery list (additional work for a group)

The application is extended with:

- Quantities: ingredients support quantities, either as numbers (e.g., a dozen of eggs), or as fractions of the initial content (e.g., the milk or jam is half empty). The quantity can be decreased when an ingredient is used. Note that adding new ingredients of the same type is done differently (as they likely would have different expiration dates).

An additional tab is added to the application, to manage a grocery list. This tab contains the list of ingredients that needs to be bought. Items can be added to the grocery list in in the following ways:

- there is an additional query that shows ingredients that have either a low quantity, or that are empty. Each of these has an "Add to groceries" button. These items are added to the list, but without the data that is not relevant (e.g. the expiration date depends on the actual item to buy).
- the "grocery list" panel has a "quick-add" entry, where grocery list items can be added via text only (just adding "pasta").

When in the shop, items can be bought, and are removed from the list. However, you wouldn't to the complete data entry while in the shop. Instead, you would do that at home:

- A "recently bought" query is added to allow edition of such items.
- In addition, if an item is "re-bought", the application can propose an expiry date based on the previously bought item. For instance, if the previous item was bought with an expiration date of two weeks from then, the same time interval is proposed, but can be edited.

Finally, shops can be defined by their location, and their type (general, butcher's shop, etc ...). The application then automatically switches to the grocery list tab when near a shop rather than at home. It is also able to sort the priority list based on location, (e.g., don't show the fish if we are in a butcher's shop).

Important details

Functional programming Your project should, as much as possible, follow functional programming principes. Functions should be small, do one thing only, and should not modify their inputs.

- Instead of changing an input, return an updated version of your input as the output of the function. Remember that the spread operators in Javascript can be used to make copies of objects and lists, which will help you for this.
- An example of doing one thing only would be that instead of directly printing to the console, you should define functions that format the data in the correct way, and define separate functions that actually print on the console.

Typescript Your project should, as much as possible, use type annotations. At the very least, any data structure you define should have a type definition, and the functions you define should have type annotations for their arguments and return types.

Grading In addition to functionality and scope of tasks, adherence to the guidelines above will be evaluated.