__Final Project Information:__

You will notice very minor changes from the project proposal. I did not choose to specify them.

After creating the Final Project, the only **major change** I made was **removing this requirement:**

- **Option to sort recipes alphabetically** or by date added/**modified**.

Therefore, the ability to "sort" can be removed from the grading criteria, which should now read:

> ***__Grading Criteria:__***
> *The final project should be graded based on the following criteria:*
>
> *__Implementation of all required features (add, edit, delete, view, search, export).__* ***4%***
> *User-friendliness of the program's interface and user experience.* ***0.5%***
> *Efficient and clean code organization and structure.* ***1%***
> *Adequate commenting and documentation for the code.* ***2%***
> *Overall effectiveness, error-checking, and functionality of the program.* ***0.5%***

By default, the recipes will already be added/sorted in chronological order (one after the other). However, to clarify, I am not storing or displaying the exact time that the recipe was created or most recently modified. As the "recipe book" is stored in a Dictionary (Dict[int, Recipe]), technically, the recipes are unordered and thus could be written in any order, but each recipe has a permanent ID (1, 2, 3, etc.) that would indicate the order regardless.

The only other small addition is that the user can now view **any** recipe, not just all at once.

Please note the following:

- **ZIP File:** I have broken up the program into 3 files: 1 file for the Recipe class (*Recipe.py*), 1 file for global-scope functions (*RecipeBookMethods.py*), and 1 file for main (*main.py*). This is for increased readability and encapsulation purposes. I have appropriately imported each file into the other file as needed. In case it is not already clear from the file, RecipeBookMethods.py includes the functions that act on the recipe book – more specifically, a Dictionary, which serves as the parameter for each function. *In total, the ZIP file should have 4 files, including this document.*

- When modifying/removing a specific ingredient/instruction in a recipe, the input must be case-sensitive for simplicity. However, searching for recipes itself is not case-sensitive.

- The "category" and "cuisine" member variables of the Recipe class are intended to store **only one piece of information**. This is for simplicity purposes. Therefore, the type for both variables is string and not List, for example, unlike the "ingredients" and "instructions" member variables. Technically, the user could enter multiple categories/cuisines, but all this data would just be stored as one string. Nonetheless, the given recipe would still come up, if/when the user searches by category.

- I have tried to use conditional statements for error-handling (namely when dealing with numeric inputs), rather than try/except blocks for exceptions. That being said, I have used the ValueError exception when dealing with floats, as isdigit()/isdecimal()/etc. do not apply.