Cooking Recipe Management System in C++

Student 1: **Cevher Sirin** (Back End Developer)
Student 2: **Oana Tudor** (Front End Developer)

I. Task Description

Student 1 (Cevher Sirin) – Back End responsibilities:

- Design and maintain the persistent data store (*recipes.txt*).
- Implement core operations: addRecipe(), editRecipe(), deleteRecipe().
- Implement SaveRecipesToFile() and LoadRecipesFromFile().
- Ensure data integrity, bounds checking, and error handling.

Student 2 (Oana Tudor) – Front End responsibilities:

- Create the console user interface and navigation flow.
- Display menus, recipe lists, details, and help.
- Validate user input and forward commands to the back end.
- Provide keyword search and coloured output.

II. File Structure

```
Recipe1 title:
Recipe1 description:
Recipe1 Ingredients:
####################
Recipe2 title:
Recipe2 description:
Recipe2 Ingredients:
####################
```

III. Console Commands

Application 1 – Back End

```
recipe_backend.exe add    "<title>" "<description>" "<ingredients>"
recipe_backend.exe edit   <index> "<new_title>" "<new_description>" "<new_ingredients>"
recipe_backend.exe delete <index>
recipe_backend.exe list
```

Application 2 – Front End Menu

```
recipe_frontend.exe menu                  # display menu
recipe_frontend.exe view                # list all recipes
recipe_frontend.exe view <index>        # view a specific recipe
recipe_frontend.exe search "<keyword>" # search by keyword
recipe_frontend.exe help                # show command summary
```

IV. Data and Classes overview

Recipe

Lightweight object representing a single cooking recipe.
Data:

- title (string)
- description (string)
- ingredients (string)

Functionality:

- getTitle, setTitle
- getDescription, setDescription
- getIngredients, setIngredients

Relationship: Base class

RecipeBook

Container that stores and manages many Recipe objects.
Data:

- vector recipes

Functionality:

- add
- edit
- remove
- list
- at

Relationship: Composition – owns multiple Recipe objects.

Storage (abstract)

Interface that defines persistent storage behaviour.
Data:
Functionality:

- virtual load(RecipeBook&)
- virtual save(const RecipeBook&)

Relationship: Base interface – implemented by FileStorage.

FileStorage

Concrete storage that reads and writes recipes.txt.
Data:

- filename (string)

Functionality:

- load overrides Storage
- save overrides Storage

Relationship: Inheritance – FileStorage is a Storage.

RecipeGUI

Console front end that interacts with users and calls RecipeBook.
Data:

- reference to a shared RecipeBook

Functionality:

- run (main loop)
- viewList

- viewDetails
- search
- viewHelp

Relationship: Uses RecipeBook but does not own it.