Project Document: Intelligent recipe book

1. Personal information

Title:	232 Intelligent recipe book
Name:	Taige Wang
Student number:	713672
Program:	Aalto Bachelor's Programme in Science and Technology
Year of studies:	1st year
Date:	24th Apr, 2019

2. General description

The program that maintains a cookbook that can be conveniently searched for a meal that meets the appropriate criteria.

The user of the program has a storage filled with groceries and accurate up-to-date information on how much of everything is in stock. The program can search for dishes that can be prepared either without going to the store. In addition, one can search for recipes using the ingredients.

Foods are labeled with an allergic factor. (For example, milk, cream, etc. containing lactose). The program is able to restrict the search so that the desired allergens are avoided.

Some raw ingredients can also be cooked from the recipes. For example, the recipe for minced beef meat contains meatloaf paste, which has its own recipe. Correspondingly, Christmas cakes are made of a dough, which can also be self-made. If the refrigerator does not have puff pastry dough, the program should try to create it from raw materials. Self-evidently some substances such as eggs and onions can be counted per piece.

Keywords:
Graphical user interface
Flat design
Ingredients and recipes management (Adding, Editing and removing)
Ingredients and recipes searching
Filtering by allergic factors
IO functionalities
Built-in Unit Test
Settings and limited customizations available (Including diagnosis setting)

3. User interface

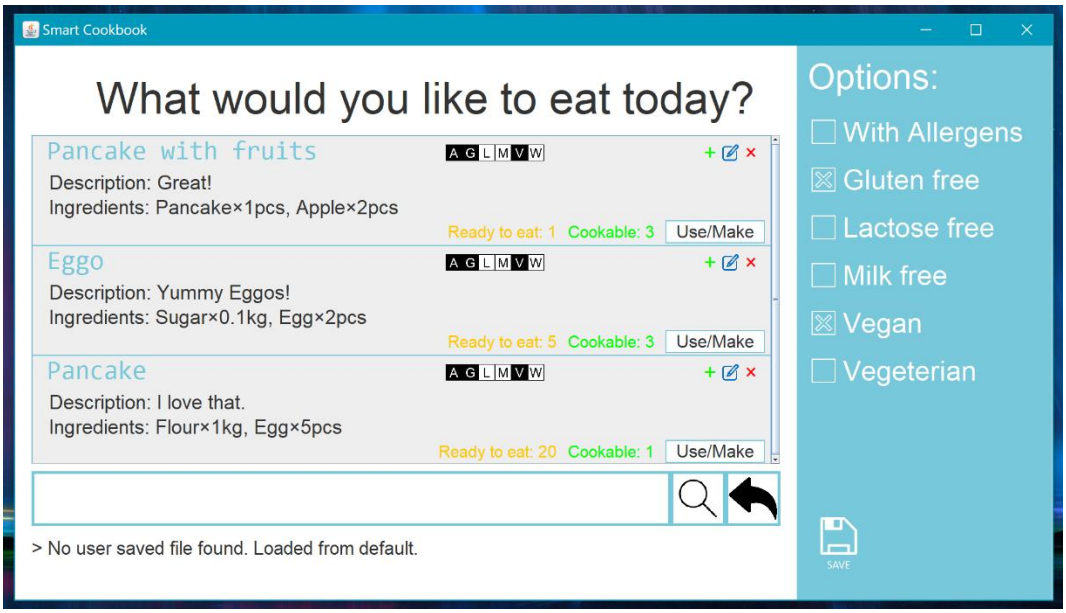


Diagram 1: Graphical User Interface

The user interface can be launched by running the project (see Diagram 1). If the user has saved some changes previously, then the program reads data from user saved data. If no user saved data found, the program reads from the default data. All data are stored under path /src/saved_data/, so the user does not need to import the data manually.
If a fatal error occurred in the data, the UI will pop up an interface with an error message, and the Save button automatically changed to Exit button, shown in the following diagram

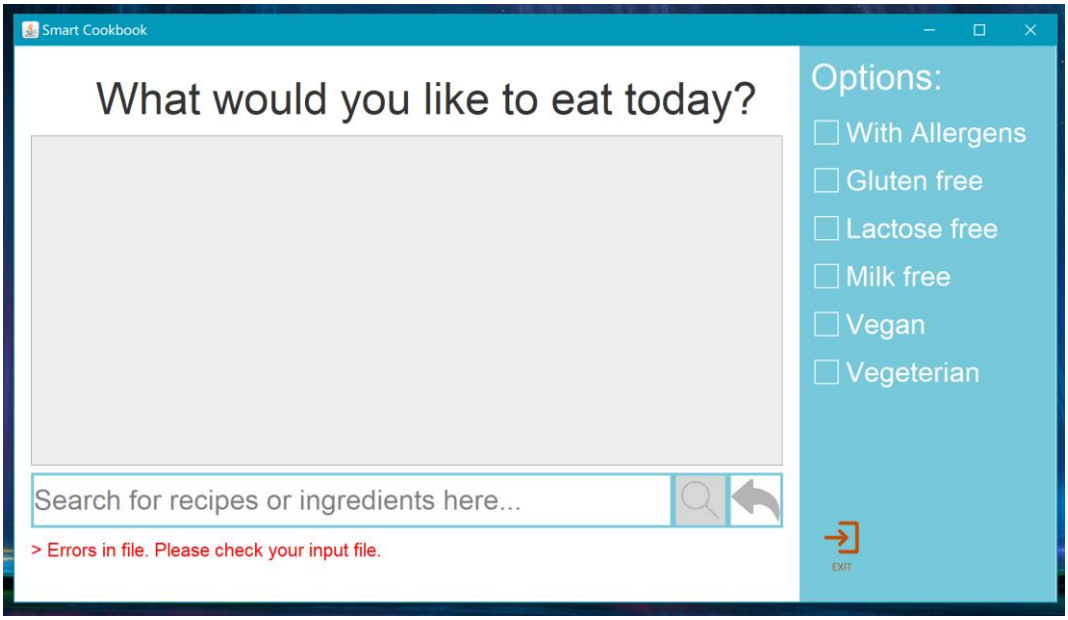


Diagram 2: Graphical User Interface with fatal IO error(s)

If there are no errors in the file, the User Interface pop up normally. UI is separated into two parts, the Navigation panel (on the left) and the Option panel (on the right). The

Navigation panel is responsible for presenting all recipes with relevant information, searching functionalities, editing functionalities.
Option panel is responsible for setting filters based on allergic factors. It also provides saving/quitting functions.

The Navigation panel includes 5 main parts, Welcome TextField, Content Field, Search section, Info bar, and Edit Section:

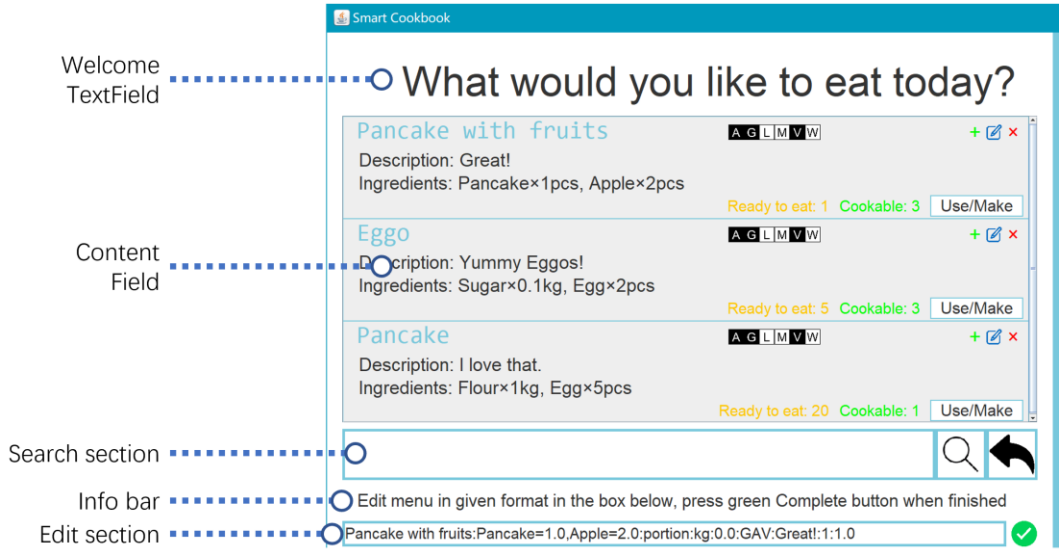


Diagram 3: Graphical User Interface with components

Functions of components in Navigation panel:

Welcome TextField

It shows the welcoming text. It does not change.

Content Field

The field that presents only recipes. Within this area, each recipe is listed in one box, containing the name, allergic factors, add/modify/delete buttons, descriptions, ingredients, availabilities, and use/make button.

The content field shows the result in a descending order based on the availability of food, so the most "available" food is always shown on the top.

The use/make button consumes one portion of the recipe selected. If the recipe is already cooked (Ready to eat in the UI), pressing the button will reduce the stock amount of that recipe; if the recipe needs to be cocked form ingredients, pressing the button reduces ingredients.

With delete button, user can delete the recipe. With add/modify button, the user can either add a new recipe or edit the recipe selected. (More details in the description of Edit section)

Notice that the allergic factor is defined manually, it does not change by the ingredients.

Search Section

The search section allows user to search for specific name/ingredient/amount by typing

the result into the search bar and pressing Find button. The Content Field will be refreshed so that it shows the search results.

Notice that the search result also includes ingredients, which are not shown in the first view at the beginning. **This is not a bug, instead, it is designed deliberately.**

The whole idea of this design is, in order to make the user feel the program is easy to use, the first interface only shows recipes, so users can choose what to eat/cook without browsing raw ingredients, because users are not expected to eat those. However, in the search interface, users can either search recipes by ingredients or check the stock of ingredients. In this case, the user needs to know ingredients, or perform some changes to those ingredients (for example, after shopping).

The search box automatically interprets your input. The search is case-insensitive and it recognizes partial names (for example, keyword “fis” can search “fish”).

If your keyword does not contain any numbers, it does not perform a search based on availability. Otherwise, it performs all searches.

After searching, the user can switch to the original content field by clicking the back button, or perform a new search.

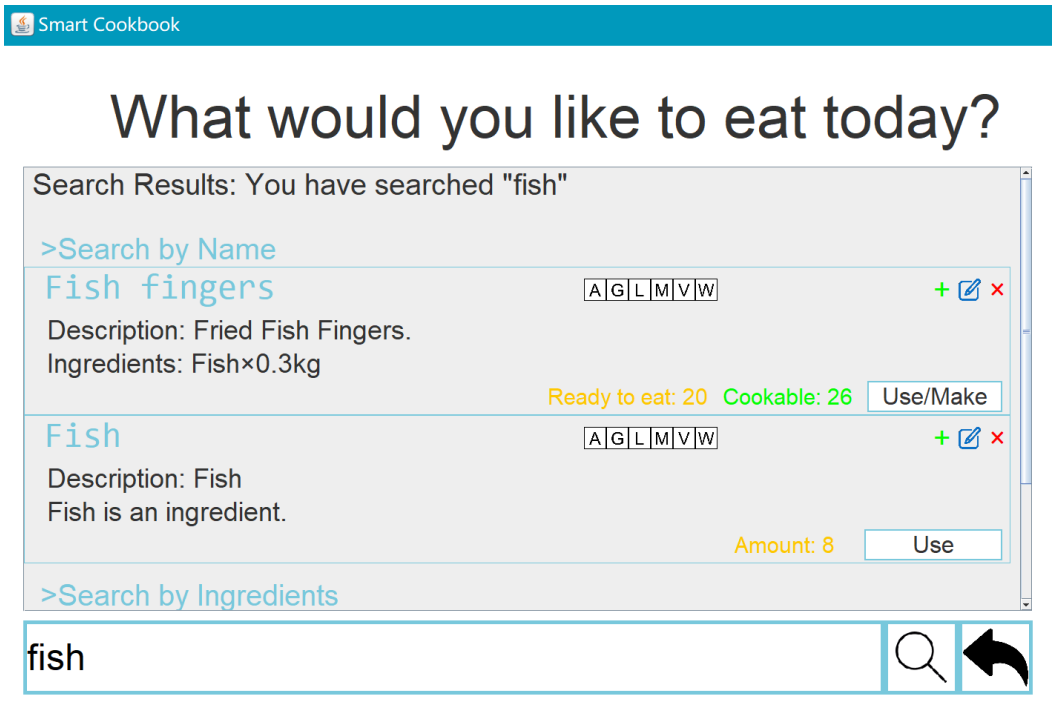


Diagram 4: Search Interface

Info bar

The Info bar provides necessary instruction for the user. It shows the status of the system, such as saving status, IO status, edit/add instructions, search/back instructions.

Edit section

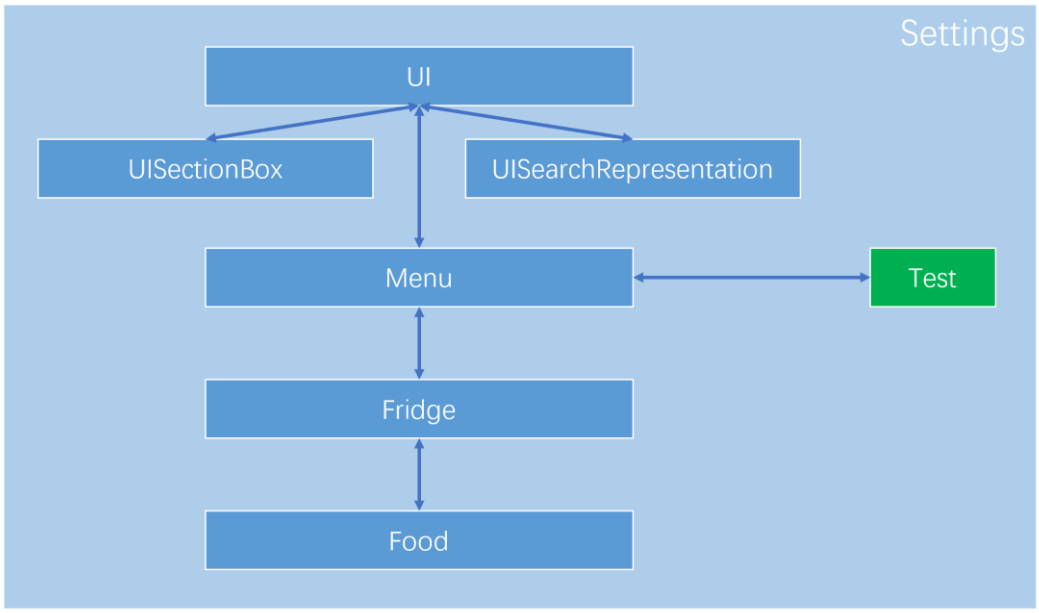
This section is only visible when users want to edit a recipe/ingredient, or add a new recipe/ingredient. The edit section accepts inputs of user, and the green icon submits the change when finished. After clicking the green button, the whole section disappears, and

the corresponding result of editing/adding (Success/Failed) will be shown in the info bar. The pattern of input is identical to the format in input file. The format looks like:
name : ingredients : main_unit : second_unit : density : tag : description : isMenu : amount
Example input:
Pasta:Tomato=3,Pork=0.2,Spaghetti=0.25:portion:kg:0::Tomato pasta with tomato sauce. Still hot.:1:8
When editing, the right option panel will not show up until the edit is finished.

The Option panel consists checkboxes and save/exit button. Each checkbox serves as an selection of allergic factor, so results shown are filtered. Pressing the save button triggers the save function, saving the data into data.txt under path /src/saved_data/.

4. Program structure

The project includes 8 parts, in the diagram shown below



UI class represents the graphical user interface. It also consists of input/output control.
UISectionBox class creates small box panel for each Food, provides to the UI.
UISearchRepresentation class creates the box panel to perform search result.
Menu class stores all recipes and has some methods of finding/managing menus.
Fridge class stores all Foods and has some methods of finding/managing/filtering foods.
Food class represents an instance of a food with its properties.
Test class is responsible for unit test, perform tests on methods in Menu, Fridge and Food mostly through Menu class.
Settings class manage the global settings of this program.
UI class has two sub UI classes, UISectionBox and UISearchRepresentation. These two classes responsible for sub-components used in UI. UI also includes input/output system so the data can be read from/write to text file. UI also represents Menu classes, which store the methods of filtering and searching. Under Menu class, Fridge is a class that stores all Food classes, it also has some methods of finding Foods. Food class is the basic

representation of ingredients or dishes with all properties.
The Test class is providing unit tests for Menu, Fridge and Food. It makes sure that all classes below UI works perfectly, as an insurance for UI working smoothly.
In addition to all classes, the Settings class work as a global setting for all classes mentioned above, to provide settings, diagnosis options and customize possibilities.
With this structure,

5. Algorithms

The most frequently used algorithm is filtering algorithm. It was done by the built-in filtering function.
The most challenging function is to test the availability of a food. If a food consists several ingredients, and its ingredients can be made from sub-ingredients so on and so forth, therefore the recursive structure is used to determine the amount.
To make one food, it first test if ingredients are enough for the food, if all ingredients are enough, it searches for the possibilities for making one more portion. If some ingredients are missing, it checks whether it is possible to make those ingredients from sub-ingredients, and this continuous until one ingredient is unavailable and no ingredients can be found. The algorithm also detects and prevents the infinite loop, so it prevents from infinite loop. It has certain offset, if certain offset is reached then the program goes to the next stage without adding the food.
Other approaches, such as doing it for food separately, or converting to raw materials is not feasible. In the first case, if A consists of one B and one C, and both B and C is made of one D. If there are 10 Ds, then available amount for B is 10, then available amount for C is 10, but the amount of A will become 10 as well if using the result from B and C, but in fact only 5 A can be made; In the second case, if A consists of one B and one C, and B can be made of 1 C. If there are 6 Bs but no Cs, then A cannot be made, but if converting Bs and Cs to Cs and recalculate, 3 As can be made, which is not correct.

The program also has some auto-correct features if error is not fatal. For example, it detects the negative values or unbelievable big values, so it makes the program works smoothly.

6. Data structures

The data is stored in a txt file in csv format. An example looks like the follows
Pancake:Egg=5,Flour=1:pcs:kg:0::I love that.:1:20
The data stored in the program is mostly in maps. The fridge stores Food in food_list in format Map[Food, Double]. This shows the amount of the quantities.
The data of Food itself is stored in variables in Food class, so all properties of a Food can be found in the Food class.

7. Files and network access

All data writes under the path /src/saved_data/.
Under the path it contains 1 or 2 files.
The file default.txt is the default file that stores the example file for the first launch. If the

user does not save changes, the program will always use this file at start-up.
The file data.txt is created when the user saves the file. The next time user launches the program, the program reads data from data.txt directly.
No network access is required for the program.

8. Testing

The project includes a unit test under the path. It tests methods in class Menu, Fridge and Food class.
Under Settings class it also has a variable called **diagnosis**. The test result and feedback from UI interface prints in the console **only if the diagnosis is on (=1)**. By default, the test is on (=1) to help in testing.

9. Known bugs and missing features

Bug:

On UNIX based system, font Consolas cannot be founded, therefore it will cause the allergic factor label and add/edit/delete button aligned incorrectly. The problem is caused by assuming that the monospaced font is used, but when the system does not have the font system uses an un-monospaced font instead. The font is included in the source folder, therefore installing the font in the system should solve this problem.
On small screen, the interface may not be shown fully. This is caused by the fact that my resize ratio is 200%, because Swing does not provide native Hi-DPI solutions for Windows, so I have to make it bigger. However, it works on most PCs and school PCs correctly, so I do not think this is likely to happen.

Missing features:

Missing N ingredients. The description is ambiguous, so I do not know if that means missing N for each ingredient, or in total missing N for all ingredients, or missing N types of ingredients.
Unit conversion. Due to time limit I do not have time for implementation.
Allergic tags do not follow the ingredient tags.

Otherwise no known problems have been found.

10. 3 best sides and 3 weaknesses

+ Well-designed flat UI with Icons
+ Only two input boxes so it makes error less likely to occur by the input of user
+ Algorithm
- Missing N missing function
- Example menu can be better, might be some minor errors in the file
- The adding button is not that intuitive, no way to add if no existing menu
  (compromised by the design so that add/edit/delete all lie on the same line)

11. Deviations from the plan, realized process and schedule

My schedule was delayed by doing Programming 2 and Y2 project, so my schedule was

later than planned, but still finished before the deadline.
The GUI and algorithm is the most time consuming part in this project.

12. Final evaluation

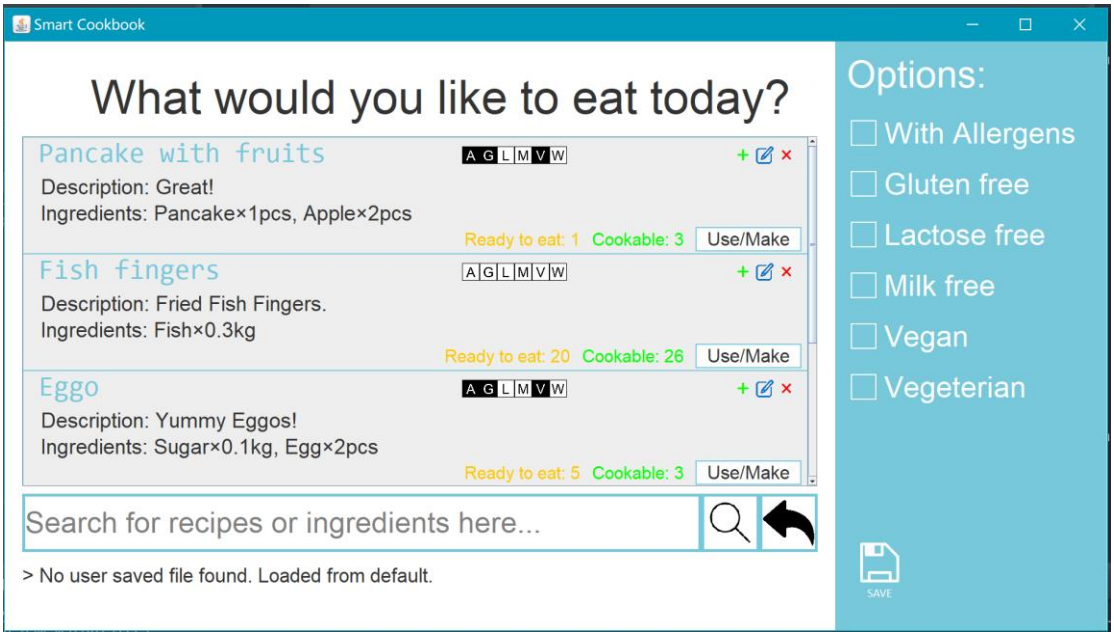
My project includes a decent graphical user interface for a recipe book. User can choose selections or filtering/searching menus from the user interface and save the modified data if necessary.
The User Interface is friendly, and the color can be customized as well. The UI is easy to use, with many icons redesigned.
The code is quite clear, but with so many components make the readability lower than expected. An improvement can be done by moving all read/write functions to a new class.
The project has quite good ability, because most components are dependent to each other and they are modularized.
At this time, I do not think I will change anything if redoing the project.

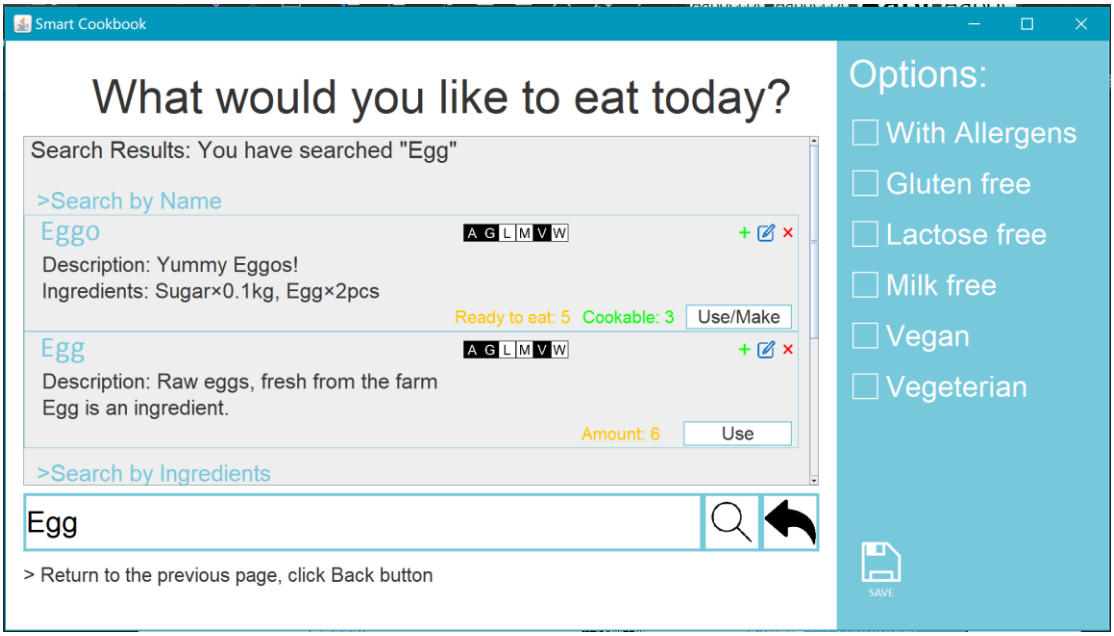
13. References

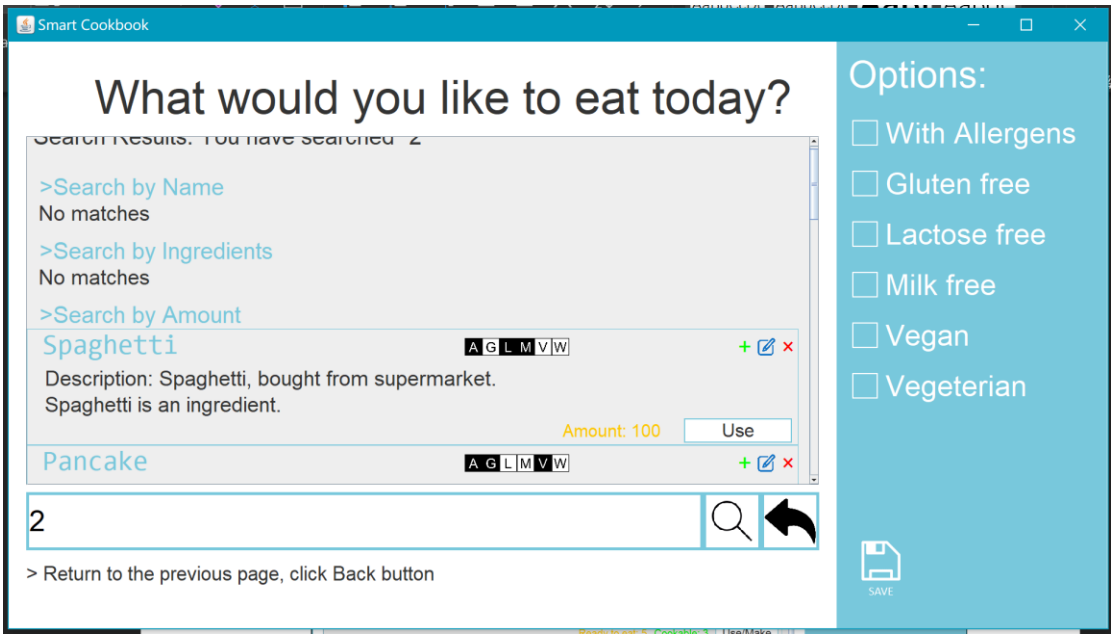
Project description page
Scala API library document
Swing API library document
Java Swing library document (To change colors and icons)
GUI Programming: http://otfried.org/scala/gui.html

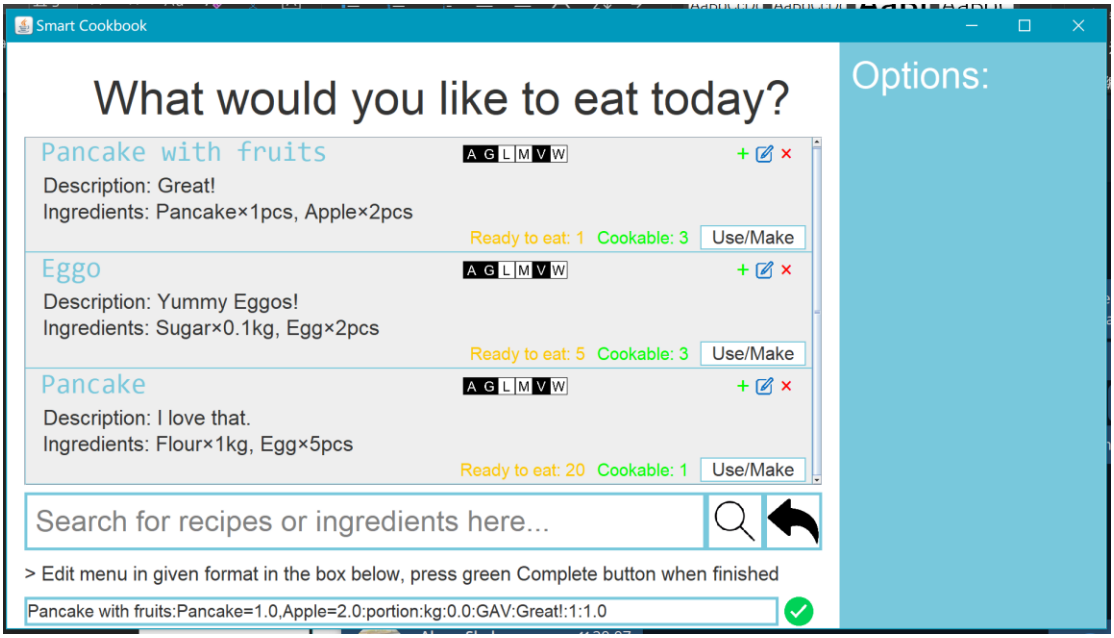
14. Appendixes

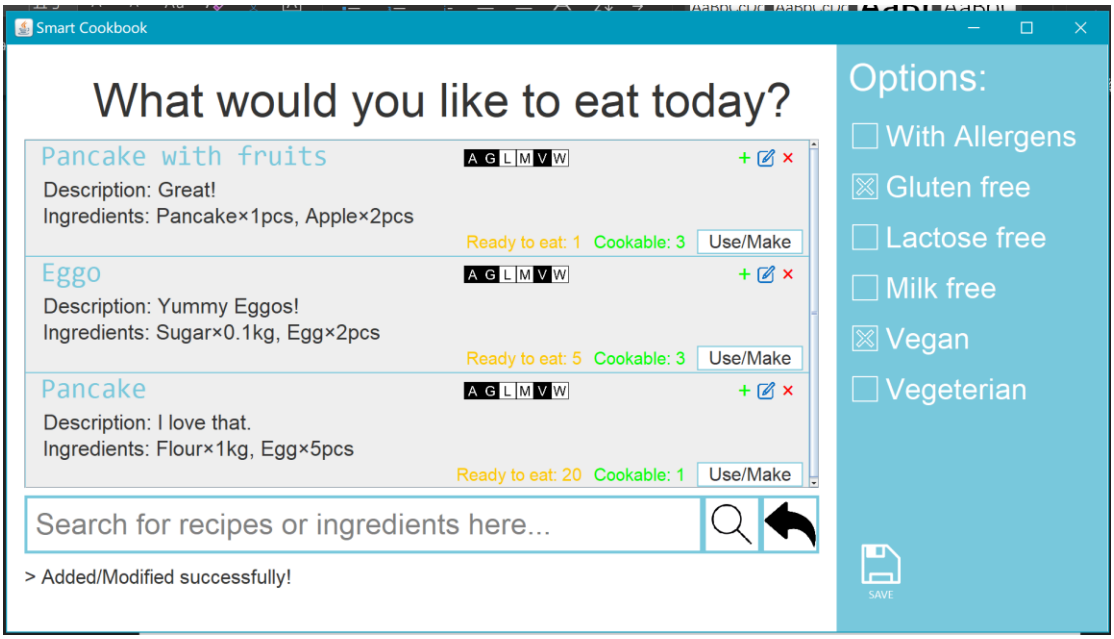
Source code (Attacted)
Screenshots




Smart Cookbook
What would you like to eat today?
Search Results: You have searched "Egg"
>Search by Name
Eggo
A G L M V W
Description: Yummy Eggos!
Ingredients: Sugar×0.1kg, Egg×2pcs
Ready to eat: 5
Cookable: 3
Use/Make
Egg
A G L M V W
Description: Raw eggs, fresh from the farm
Egg is an ingredient.
Amount: 6
Use
>Search by Ingredients
Egg
> Return to the previous page, click Back button
Options:
With Allergens
Gluten free
Lactose free
Milk free
Vegan
Vegeterian
SAVE


Smart Cookbook
What would you like to eat today?
>Search by Name
No matches
>Search by Ingredients
No matches
>Search by Amount
Spaghetti
A G L M V W
Description: Spaghetti, bought from supermarket.
Spaghetti is an ingredient.
Amount: 100
Use
Pancake
A G L M V W
2
> Return to the previous page, click Back button
Options:
With Allergens
Gluten free
Lactose free
Milk free
Vegan
Vegeterian
SAVE


Smart Cookbook
What would you like to eat today?
Pancake with fruits
A G L M V W
Description: Great!
Ingredients: Pancake×1pcs, Apple×2pcs
Ready to eat: 1 Cookable: 3 Use/Make
Eggo
A G L M V W
Description: Yummy Eggos!
Ingredients: Sugar×0.1kg, Egg×2pcs
Ready to eat: 5 Cookable: 3 Use/Make
Pancake
A G L M V W
Description: I love that.
Ingredients: Flour×1kg, Egg×5pcs
Ready to eat: 20 Cookable: 1 Use/Make
Search for recipes or ingredients here...
> Edit menu in given format in the box below, press green Complete button when finished
Pancake with fruits:Pancake=1.0,Apple=2.0:portion:kg:0.0:GAV:Great!:1:1.0
Options:


Smart Cookbook
What would you like to eat today?
Pancake with fruits
A G L M V W
Description: Great!
Ingredients: Pancake×1pcs, Apple×2pcs
Ready to eat: 1 Cookable: 3 Use/Make
Eggo
A G L M V W
Description: Yummy Eggos!
Ingredients: Sugar×0.1kg, Egg×2pcs
Ready to eat: 5 Cookable: 3 Use/Make
Pancake
A G L M V W
Description: I love that.
Ingredients: Flour×1kg, Egg×5pcs
Ready to eat: 20 Cookable: 1 Use/Make
Search for recipes or ingredients here...
> Added/Modified successfully!
Options:
With Allergens
Gluten free
Lactose free
Milk free
Vegan
Vegeterian
SAVE