Architecture Design

for

<Brew Day!>

Version 1.0 approved

Prepared by <Kexin CAO>

<Changyu SHEN>

<Yuan ZHANG>

<Xianggao GU>

<Frege>

<4/2/2019>

Table of Contents

Table of Contents ii

Revision History ii

1. Overview 1

1.1 Project description 1

1.2 References 1

1.3 Design purpose 1

2. Overall description 2

2.1 Use case diagram and class diagram 2

2.2 Design model 3

2.3 System architecture 3

3. System architecture 3

3.1 Recipe Subsystem 3

3.1.1 Description 4

3.1.2 Database 4

3.2 Brew Subsystem 4

3.2.1 Description 5

3.2.2 Database 5

4. Assessment 6

4.1 Stability 6

4.2 Reusability 6

4.3 Scalability 6

5. Alternative design (optional) 7

6. More considerations 7

7. Appendix 7

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Frege (all) | 4/2/2019 | Preliminary manuscript of the project | 1.0 |

# Overview

## Project description

“Brew Day!” application allows home brewers maintain their beer recipes. For instance, brewers can create, store, modify and remove recipes conveniently. In addition, “what should I brew today?” function make good use of the available ingredients and recommend the recipe for brewers. Optionally, brewers may choose to change ingredients manually, or do it automatically after brewing.

## References

Software Requirements Specification for < Brew Day!>. (2019). 2nd ed.

## Design purpose

This design is given to separate the whole system into different subsystems, so that each programmer can work independently, and improve work efficiency significantly.

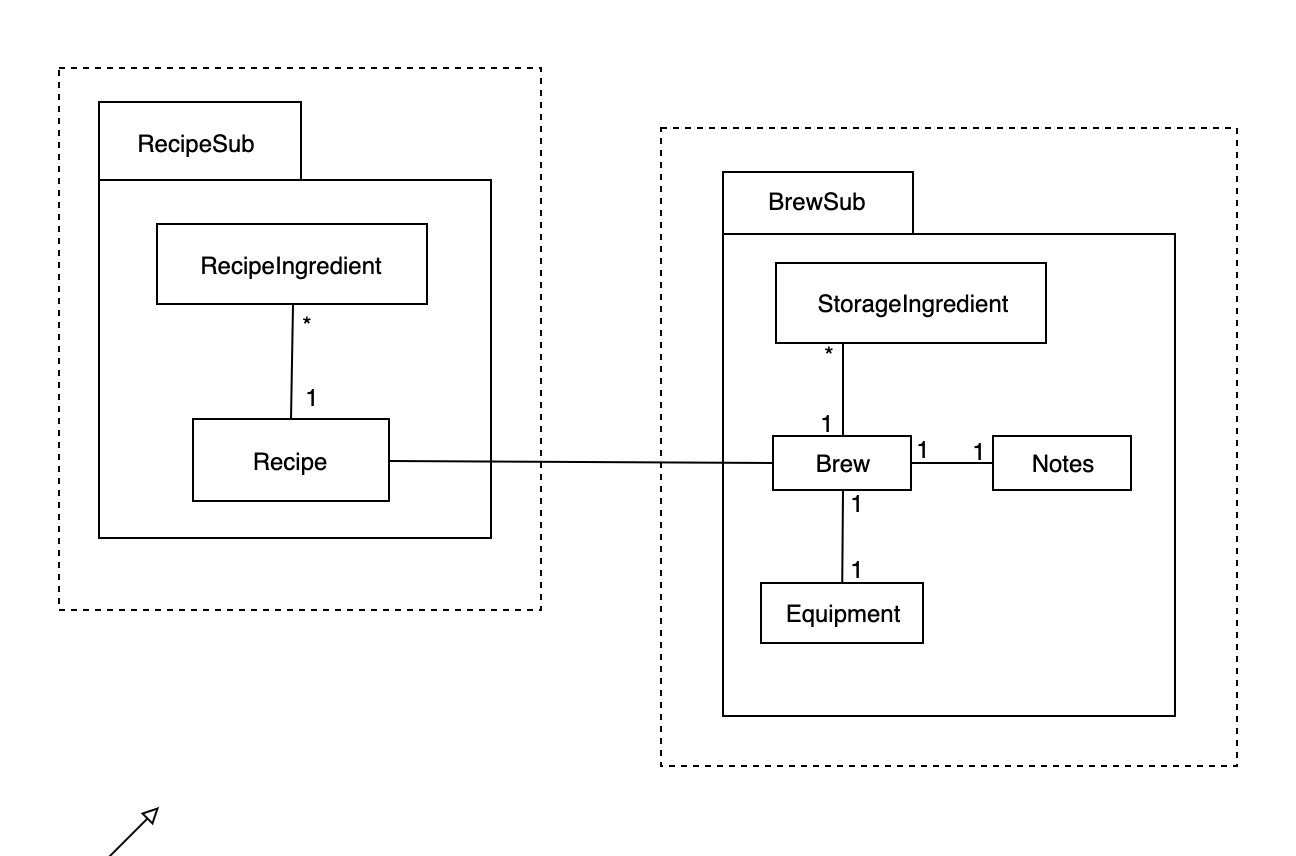
# Overall description

## Use case diagram and class diagram

## Design model

We use MVC system architecture model. It has three subsystems: one is Model subsystems; one is View subsystems; the other one is Controller subsystems. This style is suitable for interactive system because Controller subsystem gets input from users and send messages to model subsystems, and View displays the information in model and are notified of the changes in the model subsystems.

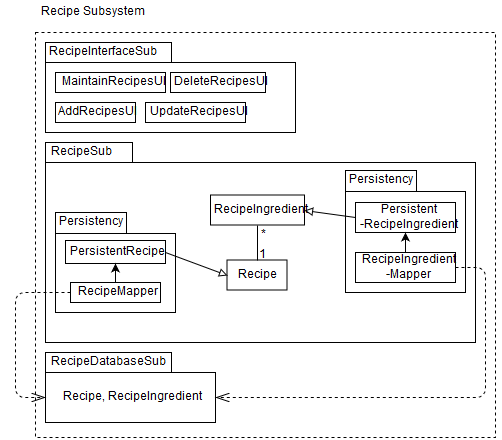
## System architecture



# System architecture

## Recipe Subsystem

There is a three-layers subsystem, which contains RcipeInterfaceSub, RecipeSub and RecipeDatabaseSub.



### Description

For the first layer, RecipeInterfaceSub, it contains all the interfaces in this subsystem.

For the second layer, RecipeSub, it contains all the entities and their relationships. The Recipe entity represents the recipes, it used for specify the format of RecipeIngredient. RecipeIngredient stores the ingredient information in particular recipe. ShoppingList entity represents the ingredients list provided for brewer to buy, so it is composed by Ingredient. Finally, the Recipe entity represents the recipes.

The third layer RecipeDatabaseSub represents the storage of this subsystem, RecipeIngredient maps with RecipeIngredient mapper, Recipe maps with Recipe mapper.

### Screen%20Shot%202019-04-03%20at%2023.24.44.pngScreen%20Shot%202019-04-03%20at%2023.24.37.pngDatabase

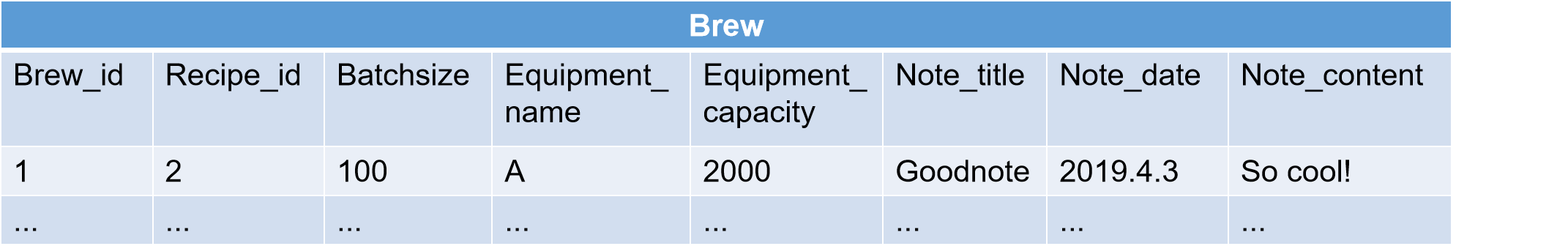
## Brew Subsystem

In this subsystem, we got three parts: BrewInterfaceSub, BrewSub and BrewDataBaseSub.

### 微信图片_20190403231110Description

In BrewInterfaceSub, all UI design are put in this part. In BrewSub, we have StorageIngredient which store the ingredients we all have; Brew, which recommend the recipe and record the brew history; Notes and equipment, which record the every note and equipment state. In BrewDataBaseSub, we got two databases, StorageIngredient maps with StorageIngredient mapper, Brew maps with Brew mapper.

### gxg1.pngDatabase



# Assessment

## Stability

It relatively is a stable system architecture based on the MVC architecture.

In the recipe subsystem and brew subsystem, they are cohesive respectively.

The recipe subsystem is divided into recipe interface subsystem which including all the user interfaces relative the recipe subsystem, recipe subsystem which including all the operation about it and recipe database subsystem which saves all the data of recipes with ingredients. Once do operation on the user interface, the data in the database will change immediately. The brew subsystem is divided into brew interface subsystem which including all the user interfaces relative the brew subsystem, brew subsystem which including all the operation about it and brew database subsystem which saves all the data of brew information. Once do operation on the user interface, the data in the database will change immediately.

And the two subsystems are coupled. The brew subsystem is corresponding to the recipe subsystem. Once brew a beer, the brew interface subsystem will be used and data will be saved in the brew database, and the recipe data which it is using will be record in the database as also. If the recipes are changed, the recipe database will be changed and it will influence brew subsystem.

## Reusability

No, it is not reusable because it is a personal-designed software and it is used for one user to record brewing information. Every component is designed for specific function and it has unique name. If similar software wants to use these components, it needs changing the names, databases, user interfaces.

## Scalability

The system is easy to extend with the MVC architecture. It’s easy to add or change views. Transaction logic is encapsulated in Model, so when adding a new view, it is not necessary to change the model, but because the business logic is the same, so it only needs to add a new view class. And it can easily update each independent software module independently. Since the application is separated into three software modules, we can change one module independently without affecting the other two modules. For example, changes to a recipe or an ingredient only changes to the recipe database.

# Alternative design (optional)

<TBD>

# More considerations

<TBD>

# Appendix

<TBD>