Architecture Design

for

Brew Day

Version 1.0 approved

Prepared by ZHANG Zhiyi, LUO Zichen, WANG Yuan, LU Guangxing

Cerf

2 April 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 1

2.1 Use case diagram and class diagram 1

2.2 Design model 1

2.3 System architecture 1

3. System architecture 1

3.1 Subsystem 1 */\*Replaced with the real name of subsystem\*/* 1

3.1.1 Description 1

3.1.2 Database 2

3.2 Subsystem 2 */\*Replaced with the real name of subsystem\*/* 2

3.2.1 Description 2

3.2.2 Database 2

3.3 Subsystem n */\*Replaced with the real name of subsystem\*/* 2

3.3.1 Description 2

3.3.2 Database 2

4. Assessment 2

4.1 Stability 2

4.2 Reusability 2

4.3 Scalability 2

5. Alternative design (optional) 3

6. More considerations 3

7. Appendix 3

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| ZHANG Zhiyi, LUO Zichen, WANG Yuan, LU Guangxing | 2019/4/2 | Initialize the Document | 1.0 |
|  |  |  |  |

# Overview

## Project description

For our project, we were writing an software called “Brew Day!”. This software is developed for home beer brewers, to give them some little help such as record recipes, keep track of ingredients, etc. This software will be desktop-based.

## References

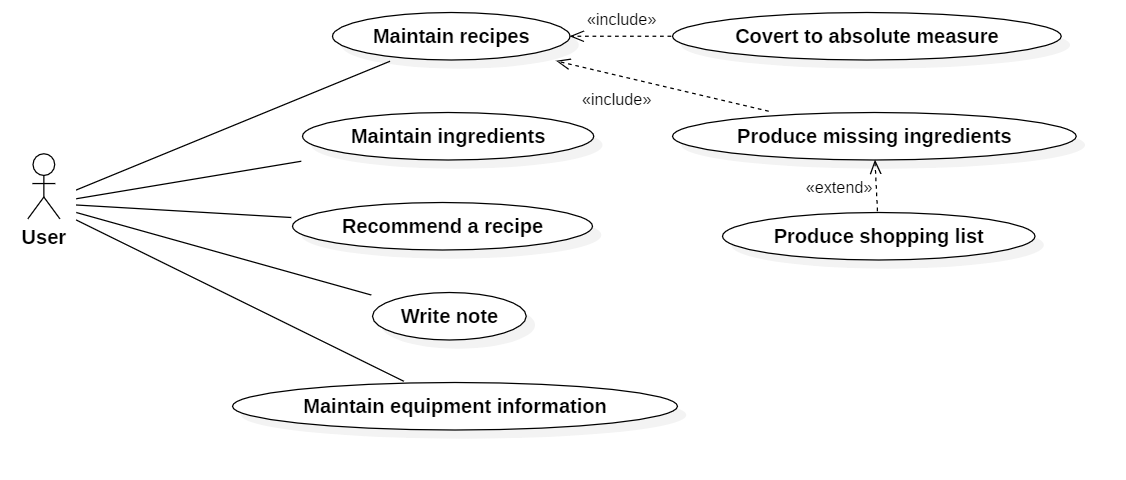
[1] Zhang Zhiyi , Luo Zichen ,Wang Yuan and Lu Guangxing, Software Requirements Specification For Brew Day!, version5.0.

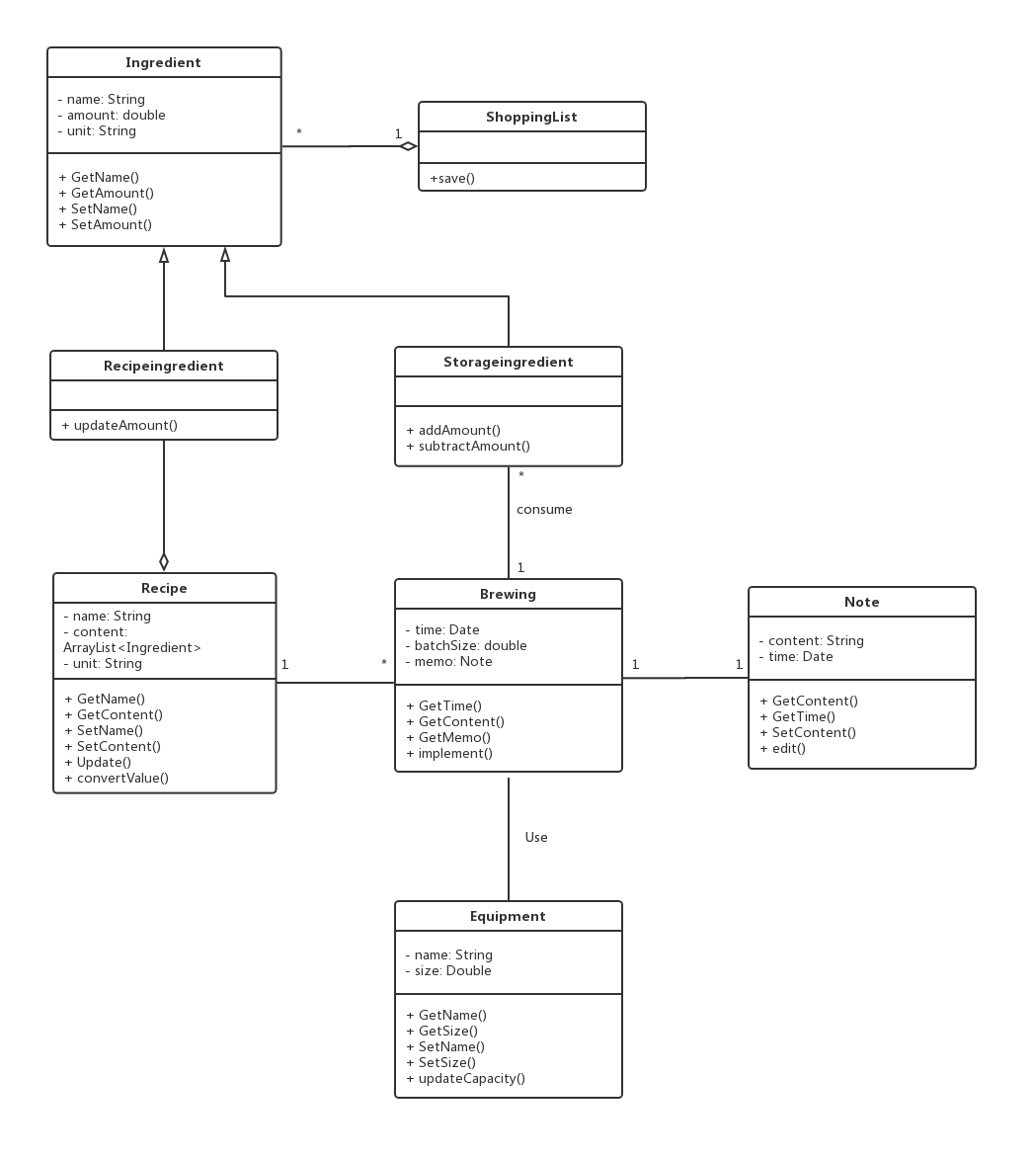
## Design purpose

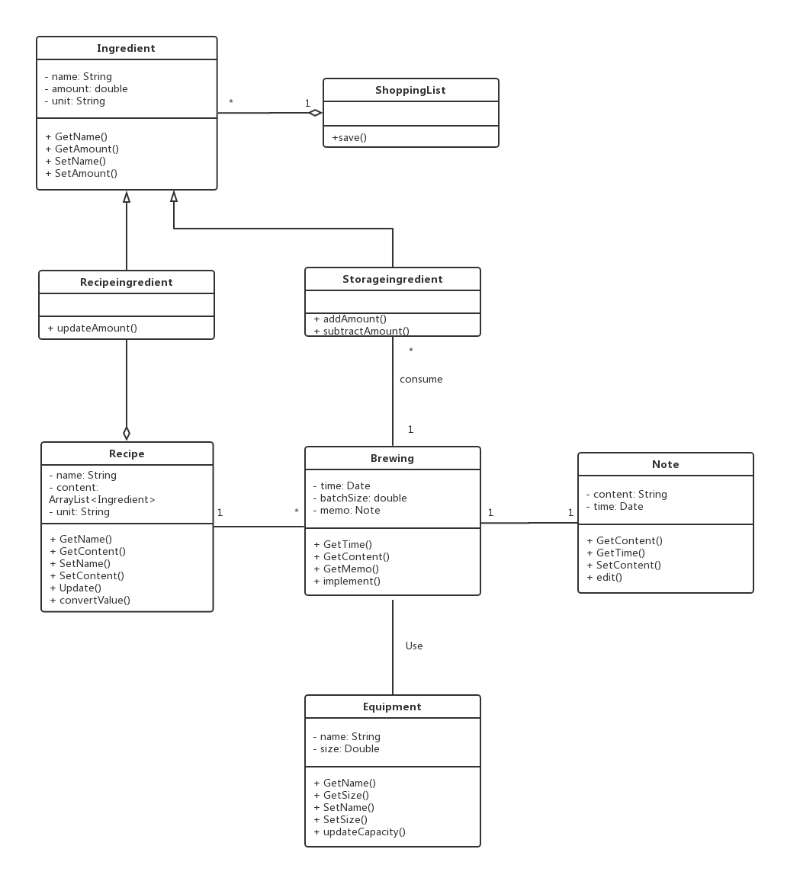
This software allows home brewers to maintain an organized database of their beer recipes. It allows users to create, store and modify recipes, or delete them. Also it allows brewers to be notified about missing ingredients for the next brew then add them to the shopping list. Meanwhile, this software will give you a list of recommended recipes for each day based on the raw ingredient residues after each brewing. The user will brew according to the equipment and ingredient, and each brew will be recorded. At the same time, the user can add note.

# Overall description

## Use case diagram and class diagram

2.1 Use case diagram

2.2 Class diagram



## Design model

We use MVC model, which is Model View Controller. We provide the UI as the View level to the user to present the View. The main features of the software are recipes, ingredient, and shoplist as the Controller level for interaction with users. Besides, we will use the recipe as a useful ingredient for conversion, which would be useful as a Mode.

## System architecture

# System architecture

<This section is used to describe each component in the system>

## MaintainSubSystem

### Description

Maintain subsystem include two features. Maintain recipes and Maintain ingredients. Maintain recipes involves Recipe class and RecipeIngredient class. It allows user to maintain and edit recipes. Maintain ingredients involves storageIngredient class and ShoppingList class. It allows user to edit storage ingredient and get shopping list.

### Database

|  |  |  |  |
| --- | --- | --- | --- |
| Recipe | | | |
| RecipeID | RecipeName | Quantity | *Unit* |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ingredient | | | | |
| IngredientID | IngredientName | Amount | Unit | RecipeID(for association |
|  |  |  |  |  |

## BrewingSubsystem

### Description

Brewing includes adding note and consuming batchsize, which is to record the time and equipment of the brewing. Equipment includes equipment name and equipment size which can be updated.

Note is to create the note which is a string of the brewing, including contents and creating time.

### Database

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Brewing | | | | | |
| BrewingID | Time | RecipeID | IngredientID | EquipmentID | batchSize |
|  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Equipment | | |
| EquipmentID | EquipmentName | EquipmentSize |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Note | |  |  |
| NoteID | Time | Content | BrewingID |
|  |  |  |  |

# Assessment

## Stability

Our system is stable, because it’s well coupled. operations communicated by calling each other, and via parameter passing. If some changes are made to certain components, the modification should focus on that components.

## Reusability

The ability of our system’s reusability remains questionable, because our system has a high level of cohesion. Parts of our components works as one function. Whether the new usage and environment fits our current functions is remain unknown and it will impact the level of reusability.

## Scalability

Our system is easy to be extended with extra function, data set or sub-functions. Current operations can be called, the interface can be inherited and new module can access specific data by calling get() and modified through set() functions.

# Alternative design (optional)

<In case some special reasons that make the above design infeasible, an alternative is ready. Give the brief description and the reference design document here>

# More considerations

<The designer can give readers considerations and suggestion here >

# Appendix

<Put any appendix here>