Architecture Design

for

Brew Day!

Final Version approved

Prepared by

Guo Rui 1630013011
Ji Jia 1630003023
Xie Qizhou 1630003056
Chen Mingxuan 1630003002

Atanasoff

23:45 23/05/2019

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Guo Rui, Ji Jia, Chen Mingxuan, Xie Qizhou.	03/04/2019	The first version.	Initial Version
Guo Rui, Ji Jia, Chen Mingxuan, Xie Qizhou.	23/05/2019	The final version	The final version

1. Overview

1.1 Project description

"Brew Day!" is an application that allows home brewers to maintain an organized database of their beer recipes. The application allows users to create, store and modify recipes, and later on delete them, if the user wishes to do so. And users can write note after using a specific recipe to brew beer.

1.2 References

[1]. IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998.
[2]. Simon Bennett; Steve McRobb; Ray Farmer, "Object Oriented Systems Analysis and Design Using UML", (4th Edition), McGraw Hill, 2010.
[3]. Guo Rui; Chen Mingxuan; Xie QiZhou; Ji Jia, "SRS_Atanasoff_v2.1", Atanasoff Company, 2019.

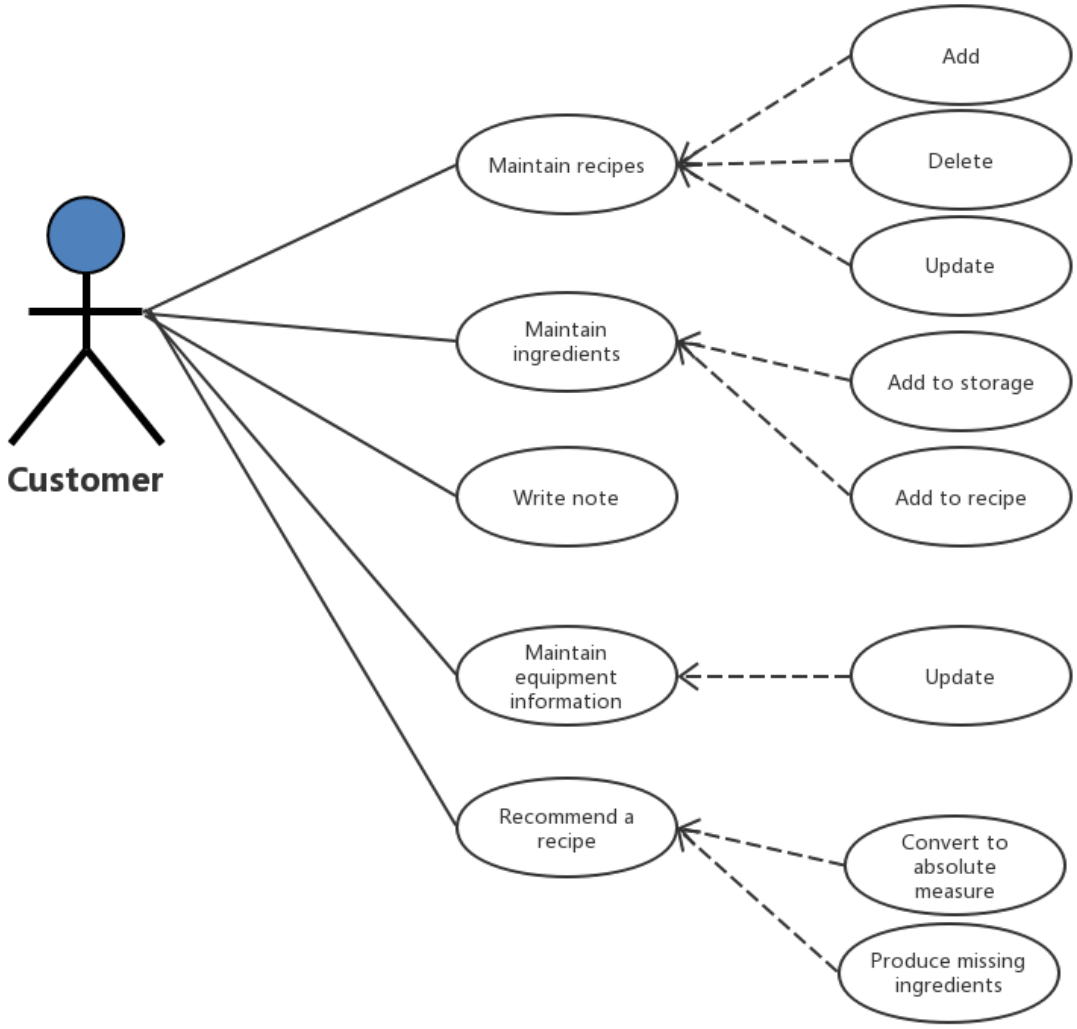
1.3 Design purpose

The architecture design document is to divide the system into several parts and let users work together in a more efficient way.

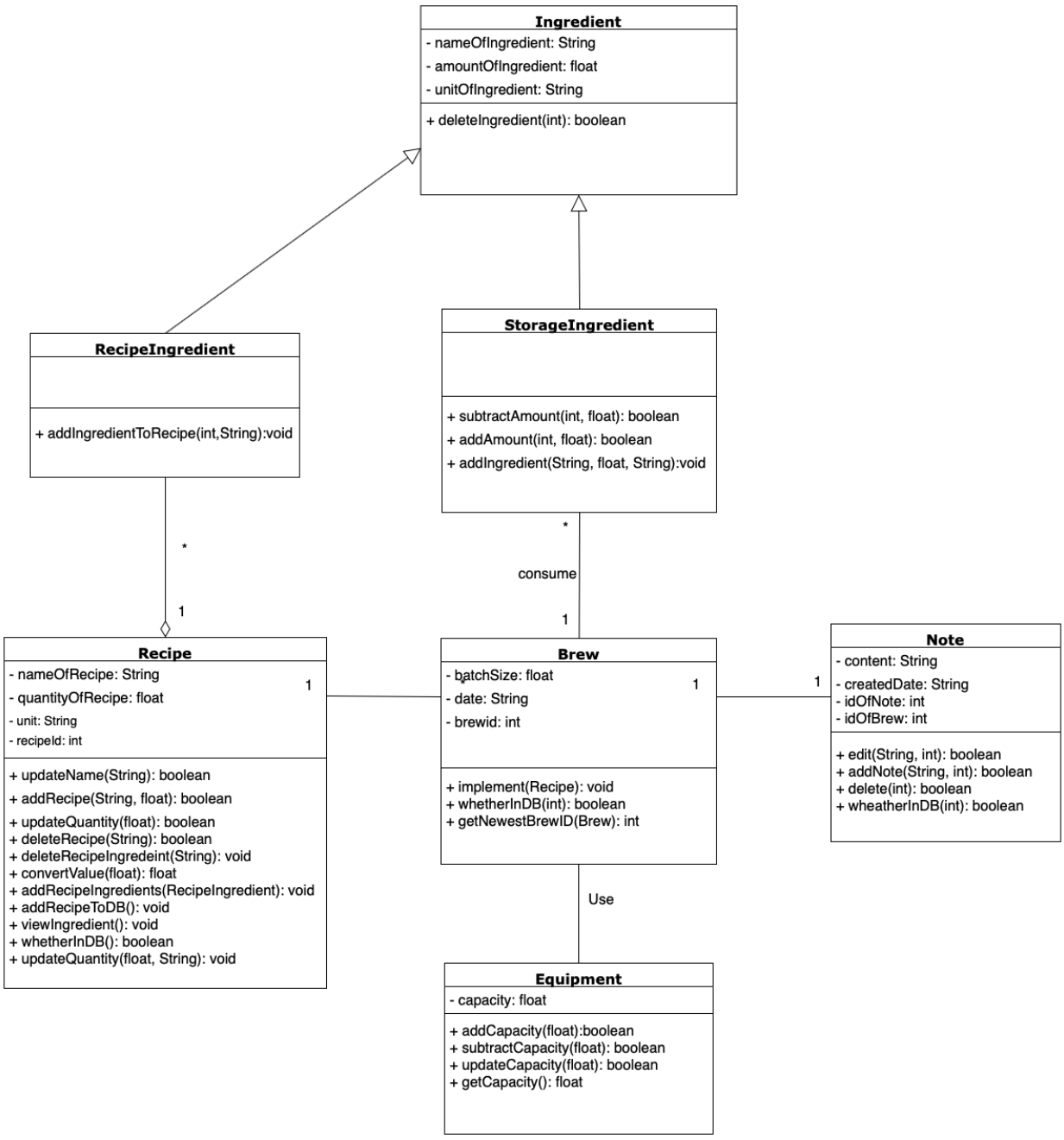
2. Overall description

2.1 Use case diagram and class diagram

Use case diagram.



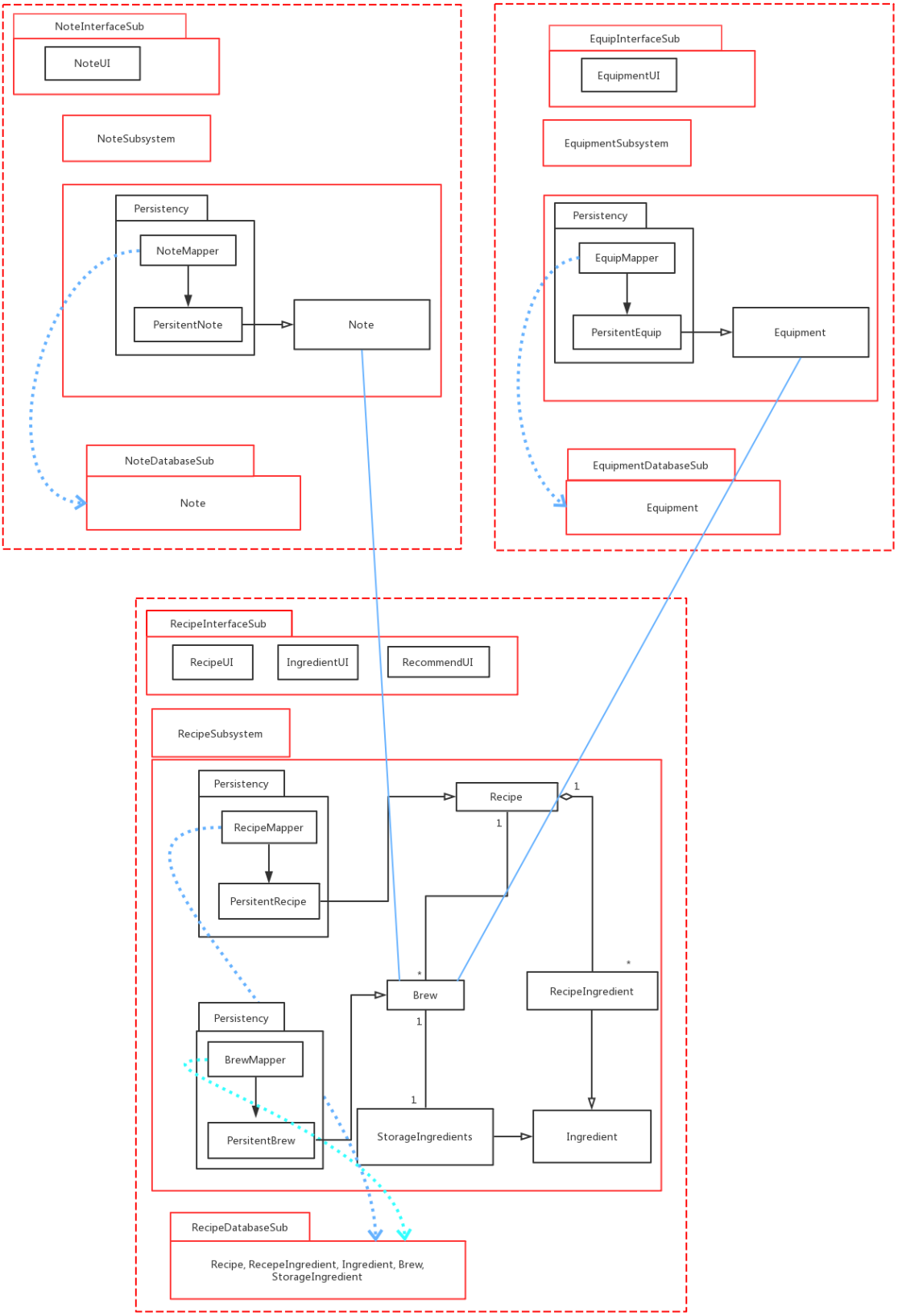
Class diagram.



2.2 Design model

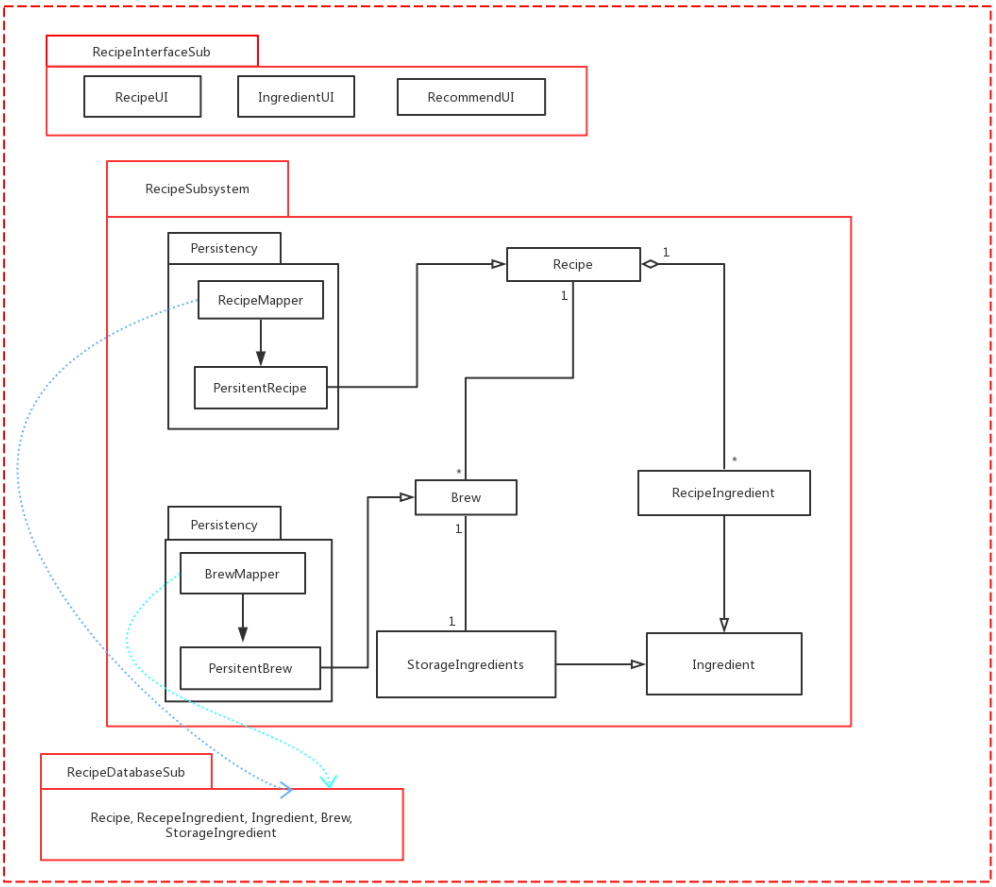
We built the architecture on MVC model. Because MVC can be developed simultaneously. And MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together, which has really high cohesion. It is easy to develop and modify in the future, because of the separation of responsibilities.

2.3 System architecture



3. System architecture

3.1 Recipe Subsystem



3.1.1 Description

In recipe subsystem, it contains recipes, ingredients and recommendations. The recipe mapper and brew mapper will stored as permanent data.

3.1.2 Database

Brew table

BrewID	BatchSize	Date	RecipeID
1	10	2019-05-23 00:55:19	1
2	90	2019-05-23 01:03:10	4
3	10	2019-05-23 01:12:56	4
4	10	2019-05-23 01:23:14	2
5	0.5	2019-05-23 01:25:39	3
6	10	2019-05-23 01:27:59	3

Recipe Table

RecipeID	Name	Quantity	Unit
1	beer	10	L
2	wine	20	L
3	happywater	14.9	L
4	test	100	L

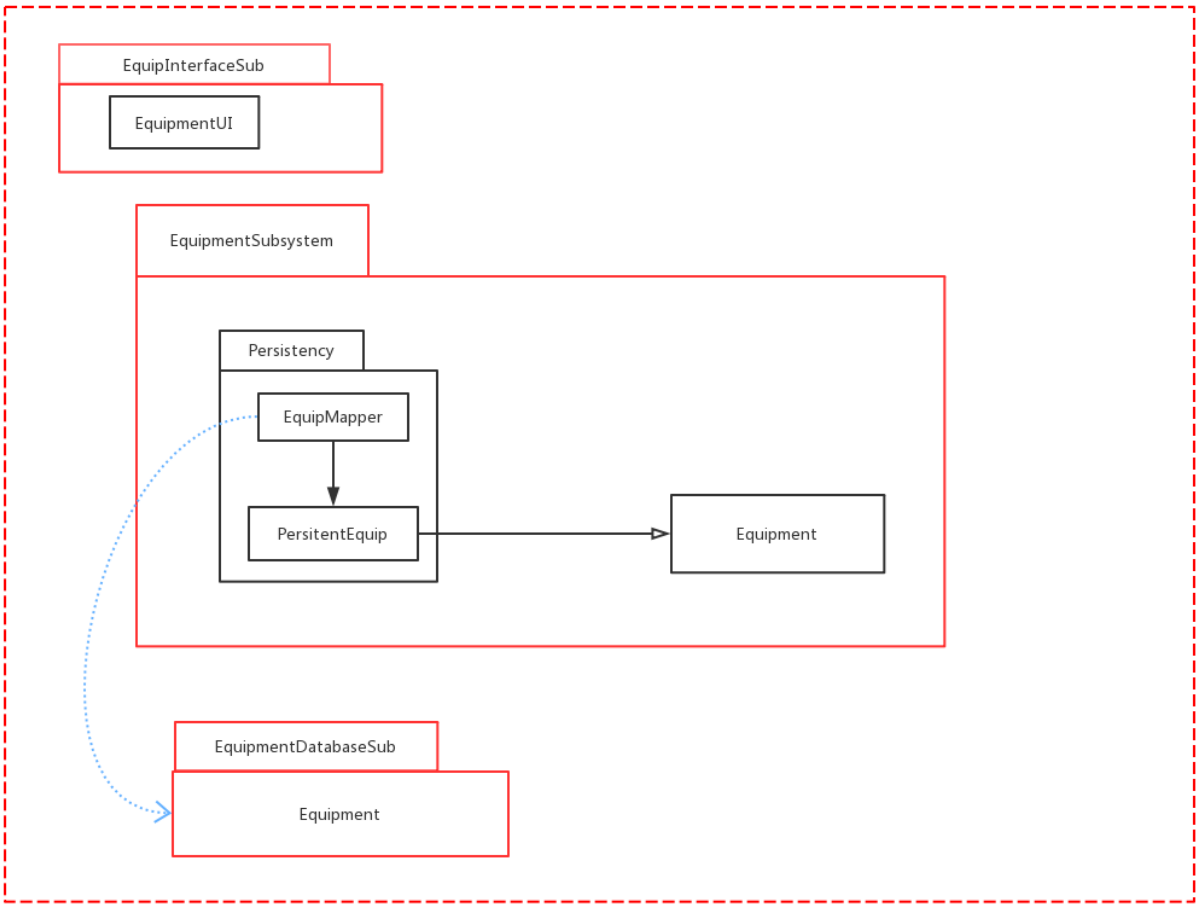
RecipeIngredient Table

RecipeIngredientID	Name	Amount	Unit	RecipeID
1	water	5	L	1
2	malts	5	g	1
3	hops	8	g	1
4	yeasts	9	g	1
5	sugars	20	g	1
6	additives	2	g	1
7	water	12	L	2
8	hops	25	g	2
9	yeasts	6	g	2
10	additives	5	g	2
11	water	5	L	3
12	hops	10	g	3
13	additives	52	g	3
14	water	99	L	4

StorageIngredient Table

StorageIngredientID	Name	Amount	Unit
1	water	956.159	L
2	malts	941.45	g
3	hops	963.309	g
4	yeasts	985.545	g
5	sugars	980	g
6	additives	962.441	g

3.2 Equipment Subsystem



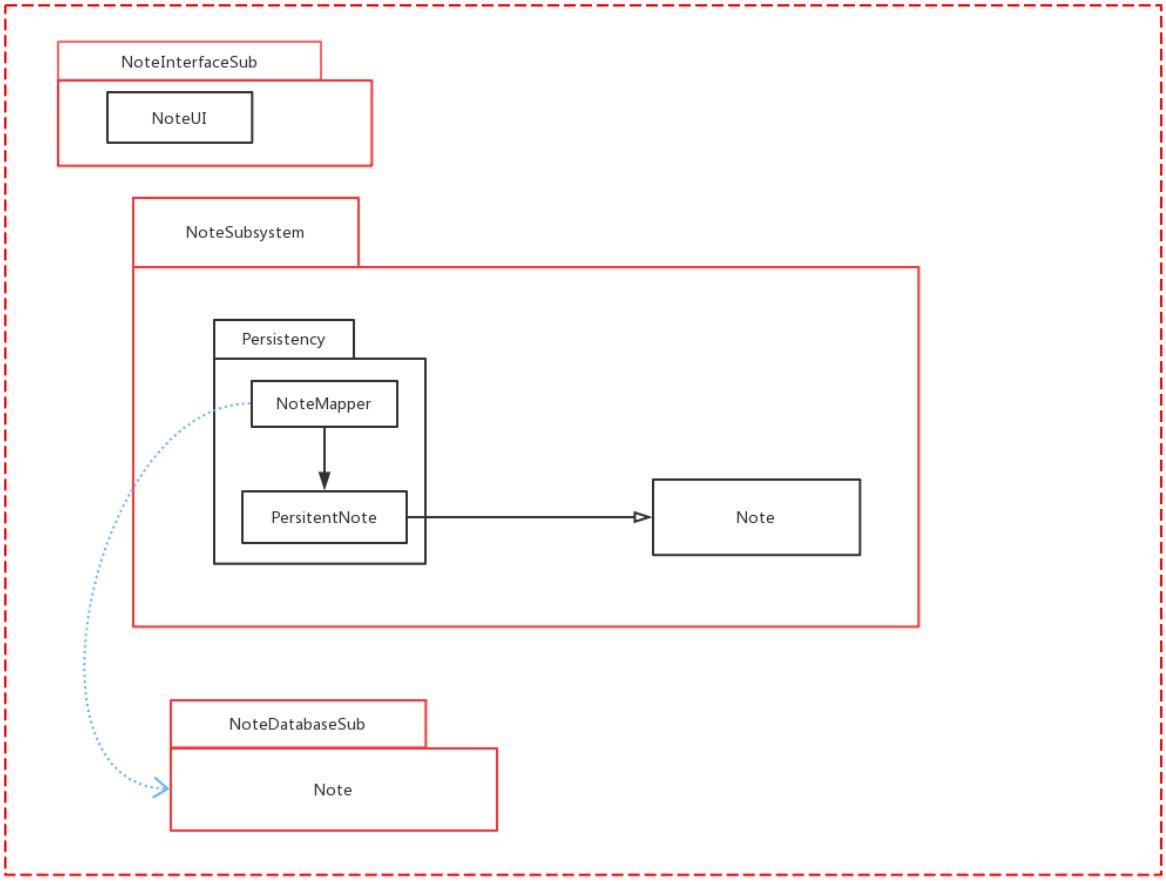
3.2.1 Description

In equipment subsystem, it is used to control the equipment. The equipment mapper will stored as permanent data.

3.2.2 Database

Capacity
300

3.3 Note Subsystem



3.3.1 Description

In note subsystem, it is used to control the writing note part of the whole system. The note mapper will stored as permanent data.

3.3.2 Database

NoteID	Content	createDate	BrewID
2	123brew directly	2019-05-23 01:03:23	2
5	88881111111	2019-05-23 01:24:09	4
6	taste good!	2019-05-23 01:28:11	6
7	very very good	2019-05-23 01:32:04	7
8	nonono	2019-05-23 18:19:06	8

4. Assessment

4.1 Stability

This architecture that the system is based on is really stable. Because we design the system by MVC model. If some changes are made to certain components, there will not be influenced to other subsystems components.
In MVC model, controller can relate all the other parts together to have a high cohesion. And the coupling in controller, view and model in MVC model framework is low. Thus, the system is really stable.

4.2 Reusability

The components of the system can be used in other place without changes or with just a little change. For example, in the model component, all the data will connect to the database in the model. And MVC model allows access to the same controller with a variety of different styles of view, because multiple views can share a single model.

4.3 Scalability

It is easy for the system to extend the architecture. Just like reusability, we can add component all based on the database in the model. And we can design different views, the UI, for the system. All the developers can easily extend with the architecture.

5. Alternative design (optional)

6. More considerations

We will use java language for this system.

7. Appendix