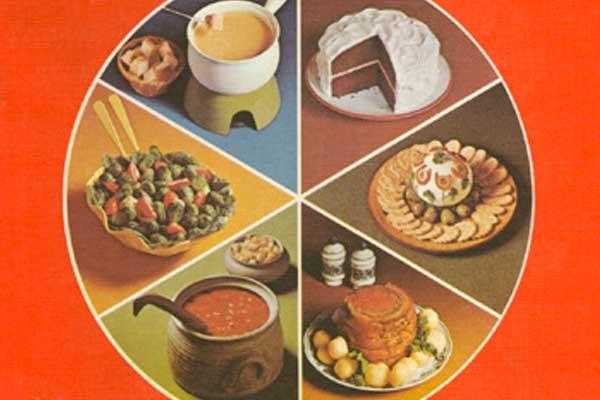


**Software Engineering II Project Report**

**Digital Cookbook**

****

**Group Three “Are You Hungry “**

Hua Yichen

Wang Jungang

Shan Jiaxiang

Kong Yu

Part task: 1. Login & Logout & Registration

2. PDF Export

3. like it – collection of favorite dishes

4. Search by ingredients

**Content**

[**1.** **Specification** 3](#_Toc450063039)

[**1.1.** **Description** 3](#_Toc450063040)

[**1.1.1.** **Cleaning help** **错误!未定义书签。**](#_Toc450063041)

[**1.1.2.** **Additional Task** 3](#_Toc450063042)

[**1.2.** **Product functions** 3](#_Toc450063043)

[**1.3.** **User characteristics** 3](#_Toc450063044)

[**1.4.** **Functional requirements** 3](#_Toc450063045)

[**1.5.** **Non-functional requirements** 3](#_Toc450063046)

[**2.** **UML Specification** 3](#_Toc450063047)

[**2.1.** **Use Cases** 3](#_Toc450063048)

[**2.2.** **Class Diagram** 3](#_Toc450063049)

[**2.3.** **E-R Diagram** 3](#_Toc450063049)

[**3.** **GUI Design** 3](#_Toc450063050)

[**3.1.** **Structure** 3](#_Toc450063051)

[**3.2.** **Screenshots** 3](#_Toc450063052)

[**4.** **Test** 3](#_Toc450063053)

[**4.1.** **Description** 3](#_Toc450063054)

[**4.2.** **Results** 3](#_Toc450063055)

[**5.** **Evaluation** 3](#_Toc450063056)

[**5.1.** **Group Work** 3](#_Toc450063057)

[**5.2.** **Task Responsibilities** 3](#_Toc450063058)

**6. Acknowledgements…………………………………………………………….……………………………………………………..1**

# **Specification**

# **Description**

# **Digital cookbook**

Digital cookbook is a software in which users can read recipes. With this software, users will be convenient to find solutions on how to make specific dishes.

A user can browse the whole list of recipes without any additional requirements. For those conservative users who prefer papery media, the software provides “Export to PDF” function. Also, if a user register for an account, he can add recipes to his favourites. When users have ingredients but only have a vapour idea on what dishes they can cook, the “search by ingredients” function can help them find which to choose.

# **Additional Task**

***➢ Export PDF file***

Users can export one recipe to a PDF file.

***➢ Search for ingredients***

Users can search recipes by the ingredients they selected.

***➢ Register and log in***

Users can register for their own account. With an account, users can add recipes to their favorites.

***➢ Add to favorites***

Users can add recipes to their favorites. The favorited recipes can be found in “My Favorites”.

# **Product functions**

Details of functions of cookbook are as follows:

When we open this system, this application will show a welcome page. After entering the correct username and password, the home page will appear with a search bar on the top and a list of brief description and details of all recipes at the button. Clicking on one recipe, the detailed information of this recipe will show in the right interface.

**➢ *PDF Exporting***

Users can extract the basic information of specific recipe - in form of PDF file - by clicking “PDF Exporting” button in the User Interface, through which the recipe can be shared to others in the Internet or be printed out so that the user can read it while cooking.

***➢ Searching Recipe***

Users can search recipes they want by offering key words regarding the basic information of the recipe (e.g. name of recipe or ingredients etc.) to the “Search Area” offered in the User Interface. The correct results will be shown in the area below the “Search Area”.

***➢ Editing/Adding Recipe***

Users can edit or add recipe to our system. For every recipe, there exists a “Edit” button, through which the users can feel free to edit on the information of the recipe (e.g. procedures, ingredients or upload an image etc.). In our User Interface, we also offer an “Add” button, through which users can add new recipes by providing basic information of a recipe.

***➢ Delete Recipe***

We offer in each recipe a “Delete” button, through which the user can delete the recipe whenever they find it totally wrong or out-of-date. In case of data security, all the deletion are logical.

***➢ Login & Logout & Registration***

For using our digital cookbook, the user must register. During the registration, the user must fill out the form about basic information like username and password. For every time the user want to search in the cookbook, they should login the system first. After finishing using the system, the user should logout.

***➢ “Like it” Button***

For every recipe the users can mark it as “Favorite” by clicking the heart icon in the left side, which mean the recipe is marked as favorite. The favorited recipes can be found in the favorite folder.

# **User characteristics**

***➢ Age***

The age range we target at is from 16 to 60. It is important that users are able to operate some cooking machines or tools. People who are too young or too old may face danger when operating.

***➢ Language and culture***

Users should be able to read in English because the contents are written in English. Besides, this application is friendly to users from different food cultures, because it will include many different kinds of recipes.

***➢ Motivations***

Users with great enthusiasm for cooking will find usability in this application.

***➢ Related skills***

Users must have basic computer operating skills to operate this application. Besides, basic cooking skills are expected because these skills won't be included in the digital cook book.

***➢Device compatibility***

User’s electronic device should be able to run JAVA programs otherwise this application won't work.

# **Functional requirements**

***➢ Welcoming & Login Panel and Tutorials***

The system should provide user with a welcome page, showing the name of our system, welcome information, a brief introduction to our application & login panel (username and password) and a start button to enter our application. Meanwhile, the system should also provide user with a registration button, when user click the button, a new page consists of information form will exist for users to filling in.

***➢ Page Shifting Buttons***

The system should provide user a column containing page shifting buttons which allow users to shift among different views.

***➢ Brief Recipe View***

The system should provide user a column space for displaying brief information of recipes like boxes piling up. This column should be next to the main recipe view and with clicking on the boxes, the main recipe view shall show the detailed information of the selected recipe.

***➢ Search Area and Results***

The system should provide user a search bar to type in the keywords (on Ingredients or Name of Recipes) and can use buttons to shift the search function between search by recipe name and search by ingredient name. The system should also provide user the view for search results which shares the similar design with the main recipe view and the brief recipe column.

***➢ Favorite Recipe – “Like It” Button***

The system should provide user functions to add a recipe into or remove the recipe from the favorite folder. The system should provide user a “Like it” button to click on. When user click the button, the recipe is sent to The Favorite Folder of the user as a collection of user’s favorite recipes.

***➢ Adding & Editing and Deleting Recipes***

The system should provide user functions to add a new recipe and edit existing ones. The User Interface should provide user a page for adding a new recipe or editing an existing one. The new page shows a form consists of basic information of the recipe. There should be functions to save the new recipes and save the changes on existing recipes. Moreover, the system should provide users with deletion button to delete a recipe logically.

**➢ *PDF Exporting***

The system should provide users with the ability of extracting the basic information of specific recipe - in form of PDF file - by clicking “PDF Exporting” button in the User Interface, through which the recipe can be shared to others in the Internet or be printed out so that the user can read it while cooking.

# **Non-functional requirements**

***➢ Accessibility***

The application should be free of charge or partly free so that everyone can download it from the Internet. The interface should be easy to read which using simple buttons and bars. Time used to learn how to use this application should be less than half an hour.

***➢ Adaptability***

The application should be able to run on Java Runtime Environment 8 (JRE 8) or higher on different operating systems (e.g. Windows, Mac OS, Android, etc.).

***➢ Data management and security***

The application should avoid illegal modification of internal files at any time. It should also be able to maintain a normal running status of the database. The application should also have the visibility of program progress while downloading or uploading data.

***➢ Failure management***

The application should be able to work continuously for at least 12 hours without any errors. In the case that unexpected failure occurs, the application itself should be able to inform the user what exact kind of error occurred and then try to recover to the previous normal status.

***➢ Legal and licensing issues***

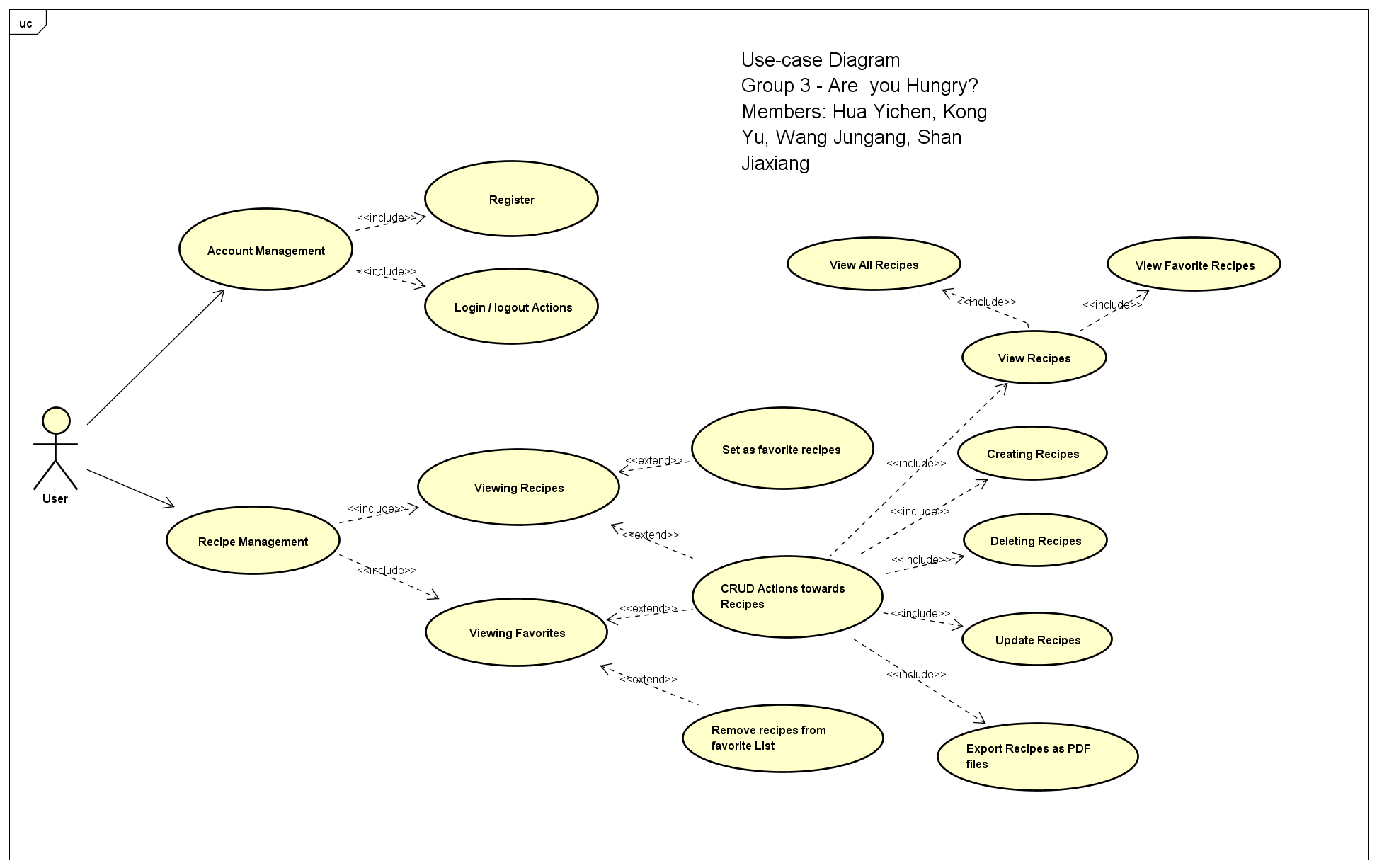
The application should not contravene local laws and regulations, especially the copyright problems should be avoided.

***➢ Maintainability***

The application should be easy to update or fix critical coding problems through legal access (usually by the programmer). The internal code should not be visible to everyone, but it should be visible to all programmers in the team. The code should be written according to the Java code conventions.

# **UML Specification**

# **Use Cases**



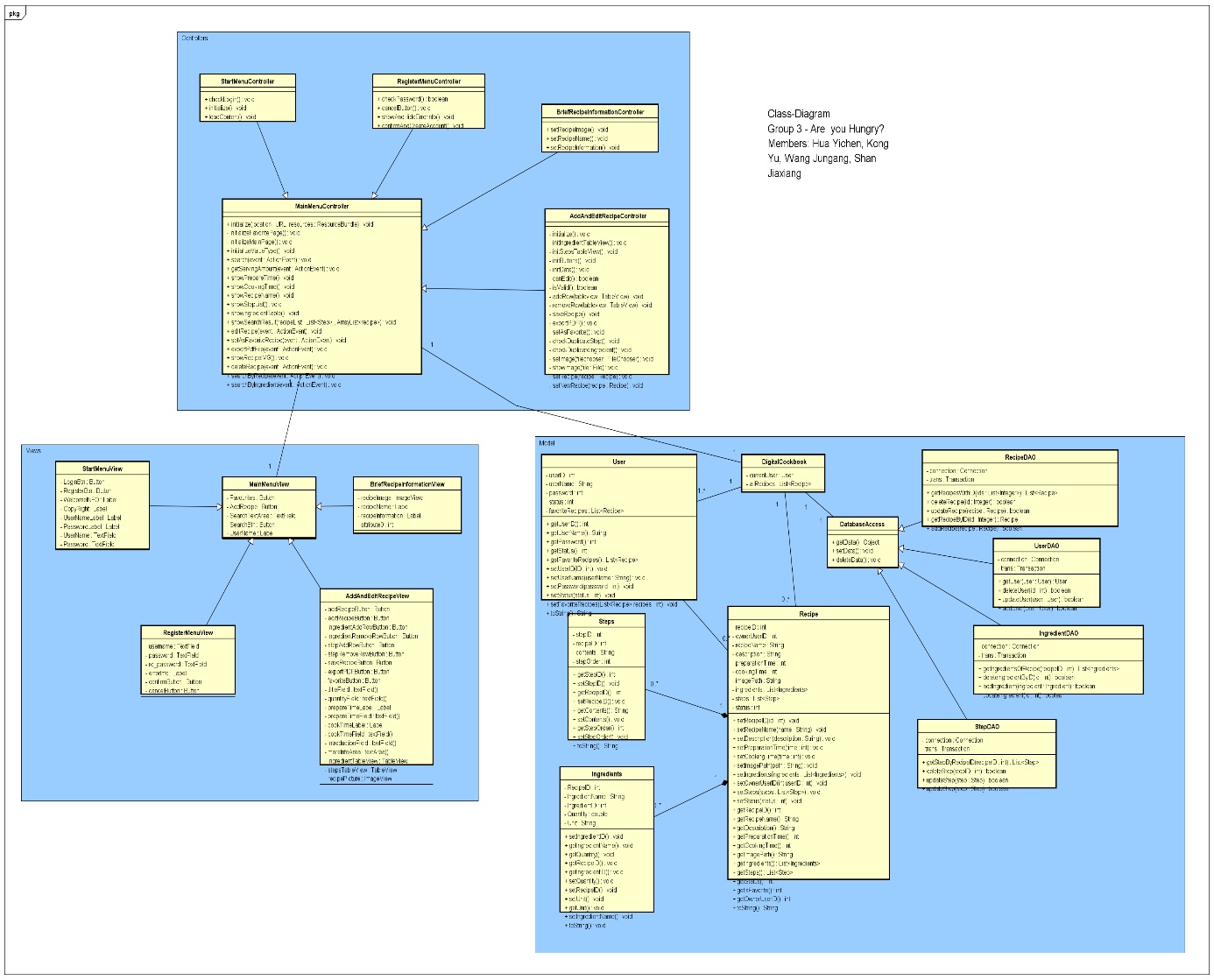
*Figure 2-1 Use-case Diagram*

As shown in the Use-Case diagram above, there are eleven use cases for the digital cookbook, which are explained in detail below:

We classify function of our system into two main categories: Account Management and Recipe Management.

1. Account Management – mainly involves with actions towards accounts.
2. When users open the digital cook book for the first time, they will be requested to register for a new account. Otherwise they cannot use the digital cook book.
3. After registration the page will switch to a log-in button, then users can use the registered account to log in. And they will get access to all the recipes. If the users want to log out, they can click on Account button, and choose to log out.
4. Recipe Management – mainly involves with actions towards recipes.
5. When the users have logged in, they can view all the recipes in the page. If they want to view a specific recipe or ingredient they can search what they want.
6. The digital cook book provide the users with the functionality of setting favorite recipes. The users can click on Favorites button to view their favorite recipes.
7. The user can set current recipe as favorite recipe when he/she is viewing the recipe by clicking the "favorite" button. The button will only appear in the recipe screen.
8. The user can set current recipe off favorite recipe when he/she is viewing a favorited recipe by clicking the "favorite" button again. The button will only appear in the recipe screen.
9. The user can do CRUD actions towards the recipes as follows:
10. When the user has logged in, he will be able to create a new recipe by click the “Create a recipe yourself!” button on the main page. A text field will be provided for writing.
11. The user can delete recipes created by himself by clicking the button “Delete” on the recipe page. The button will only be valid for the self-created recipes.
12. The user can update their own recipes by clicking the button “Update”. The button will only be valid for the self-created recipes.
13. The user can update their own recipes by clicking the button “Update”. The button will only be valid for the self-created recipes.
14. The user can export the recipes into PDF file by clicking the “Export to PDF” button.

# **Class Diagrams**

****

*Figure 2-2 Class Diagram*

The software of our group is designed basically on this Class diagram. The software uses MVC (Model, View, and Controller) model as shown in the diagram. It is divided into 4 parts: DAO, Model, View and Controller. This model separates the whole task of the software into 3 minor tasks which means each part only need to focus on its own task without considering other parts’ job. This helps a lot when programming, because it has a clear structure so that maintainability is secured. Also, we can just contribute work to each member simply according to the MVC model. Thus, we use GitHub to collect our work. The detailed structure is shown below:

1. Model

The Model part includes the main classes in this software: Recipe, Steps, Ingredients, User and Digital Cookbook.

* Recipe

The Recipe class stores the basic data of table ‘recipe’ in the database. It can also get attributes from other classes and set attributes to create or update a recipe.

* Steps

The Steps class stores the basic data of table ‘step’ in the database. It can also set attributes to create or update a step.

* Ingredients

The Ingredients class stores the basic data of table ‘ingredient’ in the database. It can also set attributes to create or update an ingredient.

* User

The user class stores the basic data of table ‘user’ in the database. It can also set attributes to create or update a user.

* Digital Cookbook

The Digital Cookbook class is the main function of this software and it is also the entrance of this software.

1. View

The View part is the interface which user can see directly. It has the task to show data or get user’s input by varies of methods. It consists of 5 parts: Start Menu View, Main Menu View, Register Menu View, Brief Recipe Information View and Add and Edit Recipe View.

* Start Menu View

Start Menu View is the first interface once the software is opened. It has welcome information and copyright information. User login in this view by enter username and password in the text field. If the user has not an account, he/she will be asked to go to the Register Menu View.

* Main Menu View

Once the user logged in, he/she will come to this view. It mainly has 3 buttons: Favorite button, Add Recipe button and Search button. User can choose his/her action towards the opening recipe or search for other recipes.

* Register Menu View

If the user has not an account, he/she will come to this view. It provides the user 3 text fields to type in his/her username and password. If any error occurs, error Info label will appear.

* Brief Recipe Information View

This view shows all the information of this recipe such as: recipe name, recipe image and recipe information.

* Add And Edit Recipe View

In this view, user can create or edit a recipe by clicking corresponding button. Editable text field exists so that user can do thing by his/her will.

1. Controller

The Controller part has to manage interaction between model and view. It is the bridge of user and background data. This software includes 5 controllers: MainMenuController, StartMenuController, RegisterMenuController, BriefRecipeInformationController and AddAndEditRecipeController.

* MainMenuController

Main menu is the interface to display recipe data. Its function includes showing data fetched from the database and provide buttons of other uses.

* StartMenuController

Start menu is the interface to login. Its function includes showing static data which is needed and check the login data provided by the user is correct or wrong.

* RegisterMenuController

Register menu is the interface to register. Its function is only check whether the two-password user inputted is the same thus show or hide error info and confirm registration.

* BriefRecipeInformationController

Brief recipe information menu is simply get recipeName, recipeImage and recipeInformation from the database and display it on the screen.

* AddAndEditRecipeController

In this menu, user can input string in text fields to create or edit a recipe. Thus, its function includes get data from database and set new data into the database. Regarding to the validity of some contributes, this controller can also check them.

1. DAO

The DAO class is created in order to fetch data from database and send them to the functions requiring these data. Also, it can get data from other functions and write it into database. In this part, we have 6 classes: BaseDAO, IngredientDAO, RecipeDAO, StepDAO, UserDAO, and ExportPDF.

* BaseDAO

BaseDAO is the base of the whole DAO. It offers basic functions which are necessary and frequently used by other DAO classes: getConnection () enables other functions to get access to database, closeAll () can release system resource, executeSql () and executeQuery () is made for running SQL statements.

* IngredientDAO

IngredientDAO offers the functions on add/delete/update/search relevant data in table “Ingredient”. The functions in controller can operate the result of some functions. (E.g. searchRecipeIdByIngredientsName ()).

* RecipeDAO

RecipeDAO offers the functions on add/delete/update/get relevant data in table “Recipe”. The functions in controller can operate the result of some functions (e.g. getFavoritedRecipes ()).

* StepDAO

StepDAO offers the functions on add/delete/update relevant data in table “Step”. The functions in controller can operate the result of some functions (e.g. searchStepByRecipeId ()).

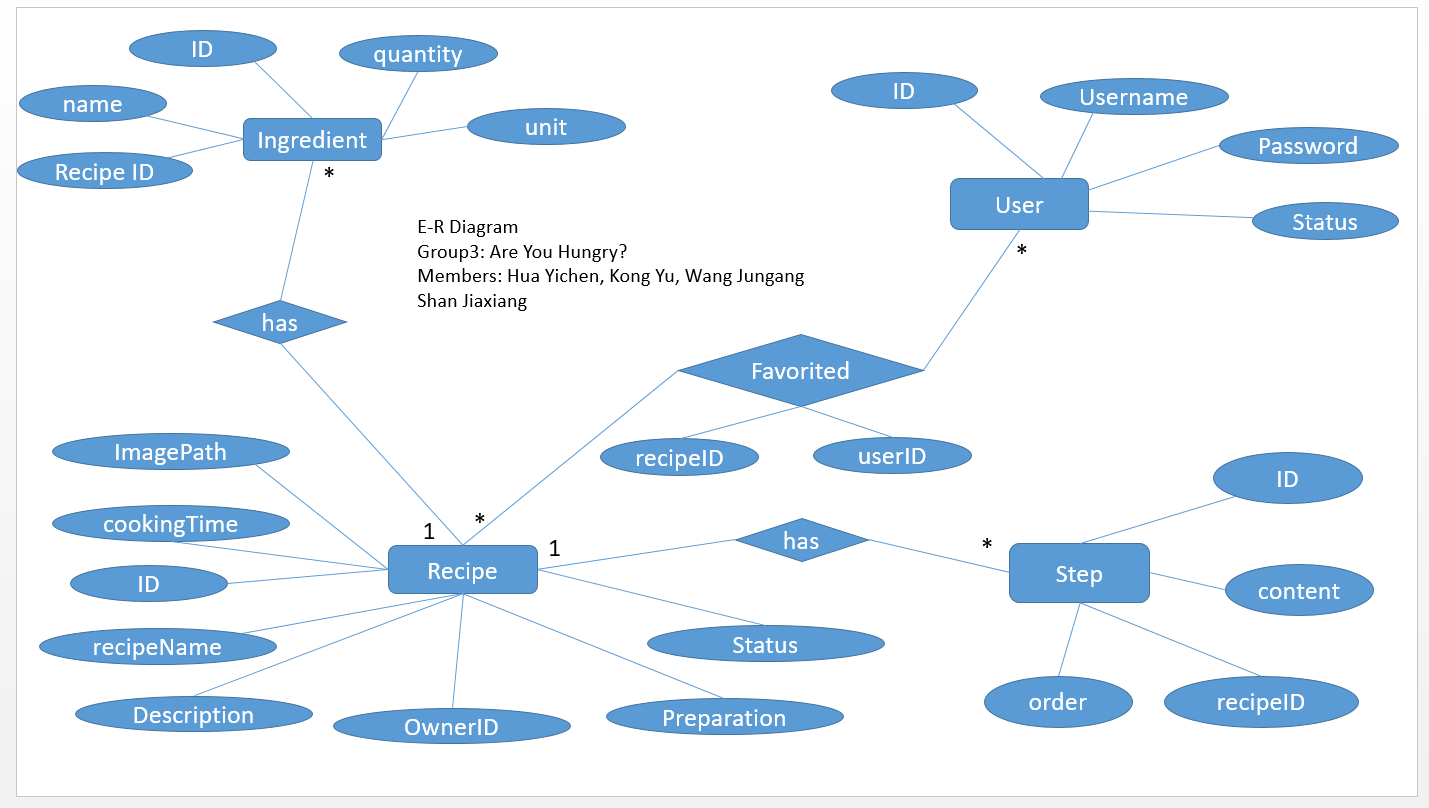
* UserDAO

The most important function of UserDAO is validatePassword (), which ensures the safety of the software. It compares the input password to that in database. In UserDAO, there are also functions getUserByID () and add/deleteUser ().

* ExportPDF

This class realizes the function of “Export recipe to PDF”. It can fetch data from database and reorganize it into a PDF file.

# **E-R Diagrams**



*Figure 2-3 E-R Diagram*

Our database is composed of 4 tables: Recipe, ingredient, User and Step. The constraints and relationships between tables are realized by Java instead of SQL, which aims at increasing the stability of the software. All tables have the “Status” column which indicates whether the object exists. (1 for exist and 0 for non-exist.)

➢ Ingredient Table

In this table stores the information of every ingredient used by recipes. To create a new ingredient item, one needs to identify quantity, name, RecipeID, unit and comments. The ID of the ingredient can be created automatically as well as status.

➢ Recipe Table

The information of recipes is stored in this table. The info includes Image Path, cooking Time, recipe ID (can be generated automatically), recipe Name, Description, OwnerID, Preparation and Status. A user can “favorite” recipes. In this relationship, recipeID and userID are recorded.

➢ Step Table

A recipe usually has more than one step. In this table responsible for storing the information of every single step. The order, recipeID, content and stepID is recorded.

➢ User Table

This table is used to store the info of users including userID, Username and password.

# **GUI Design**

# **Structure**

In our digital cookbook, the MVC model is mainly used for designing the UI. The View consists of four user interfaces, which are realized by the FXML codes generated in the Scene Builder and their further load by the FXML Loader in Java; The Controllers, basically Java Objects implements the interface Initializable, consist of Java attributes and methods that regulate the actions performed by the JavaFX elements (e.g. Button, TextArea and etc.); The Models consist of Data Access Object, entities objects and basic Utilities. With applying the MVC model, we do not have to change codes of the other two components while we are altering one of the components. Next, I will show each part of MVC in detail.

As for View, we have four FXML files, auto-generated by Scene Builder, which are “*Template.fxml”, “MainRecipeFrame.fxml”, “AddOrEditRecipe.fxml” and “BriefRecipeInMainPageView.fxml”.* *MainRecipeFrame* is for searching, presenting recipe information and “Like it” function, and *AddOrEditRecipe* is for editing or adding a new recipe. Moreover, *Template* is mainly responsible for User login and registration – we have designed two Input area for users to input the basic information of their accounts and two buttons to confirm login and registration.

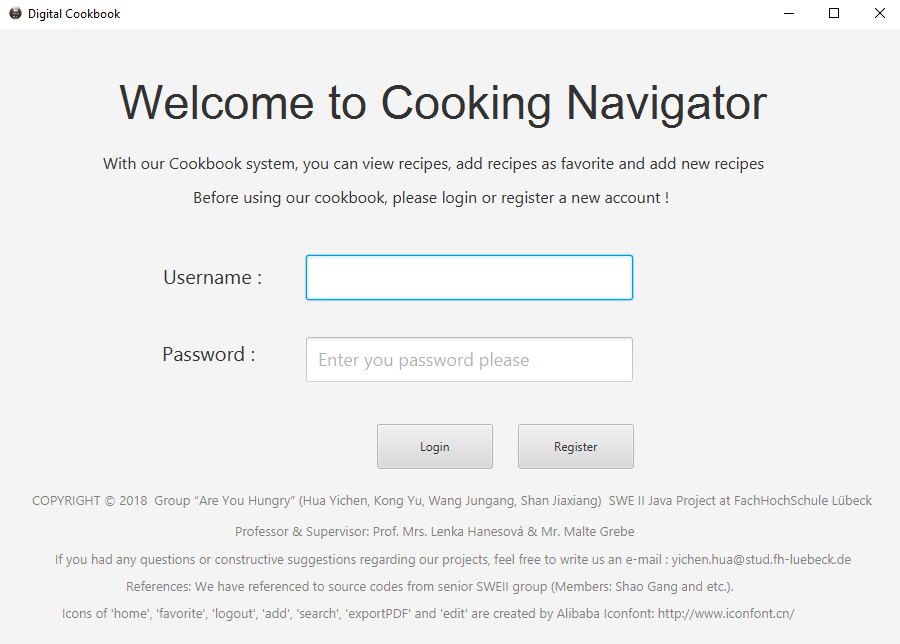
As for controllers, for every view we bind a controller with it. So there are four controllers in our project and we name them as *“TemplateController”, “MainFrameController”, “AddOrEditRecipeController” and “BriefRecipeInformationController”*. Each controller is responsible for getting users’ inputs with GUI, and pass the parameter generated by users’ interaction to the methods, exists in Model, which can query the database to return results. Then controllers send those result data back to the GUI and the whole procedures realize what we called the Interaction.

As for Model, it consists of Entities and Data Access Objects. There are five classes are constructed to represent the entities of our software. They are “*User.java”, “Cookbook.java”, “Recipe.java”, “Ingredient.java” and “Step.java”.* The Entities classes are responsible for data storage in the memory, you can also see the term ORM for Object Relation Mapping, holding attributes that are consistent with the tables in the database. For instance, when showing all the recipes we have in our system, the system firstly executes query and fetch information of recipes, corresponding with their steps and ingredients, from the database, then load data from the Result Set into Entities for further use. Hence, entities classes will help us store the data in a more structured way. Moreover, for DAO as Data Access Objects, they are mainly responsible for data fetch and storage in the database, we divide DAOs into five categories, according their serving targets, *“IngredientsDAO.java”, “RecipeDAO.java”, “StepDAO.java”, “UserDAO.java” and “BaseDAO.java”. “IngredientsDAO.java”, “RecipeDAO.java”, “StepDAO.java”, “UserDAO”* are responsible for CRUD operations with their corresponding entities classes and “*BaseDAO”* defines basic connection properties toward database.

In summary, since it is not a large application, using MVC model takes us extra efforts to follow such a structure. But after applying this model, we actually feel the flexibility of using it. Since during coding period, we can construct (working on) model, view and constructor simultaneously. This model separates not only the tasks between different layers but also separate workload of each contributors by allowing them to focus only on one part of model. And when bugs happen, this model help us to locate which part of model breaks, and then able to inform the responsible person to fix the bug.

# **Screenshots**

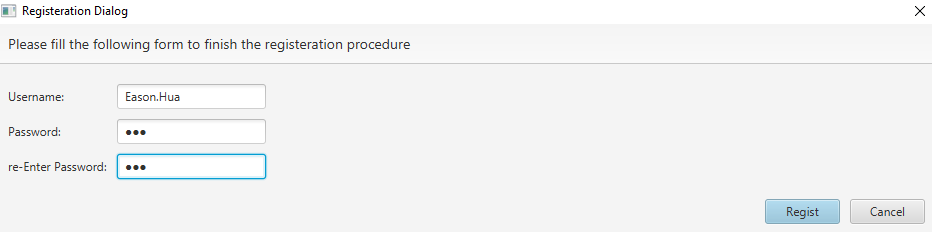
***➢* Welcome page**



*Figure 3-1 Welcome & Login Page*

When the user opens the application, the first view shown to the user is the welcome page. The welcome page is very concise, it gives brief information about this application and will guide the new user to register an account or let the old user to login, because only by registration the user can use our application.

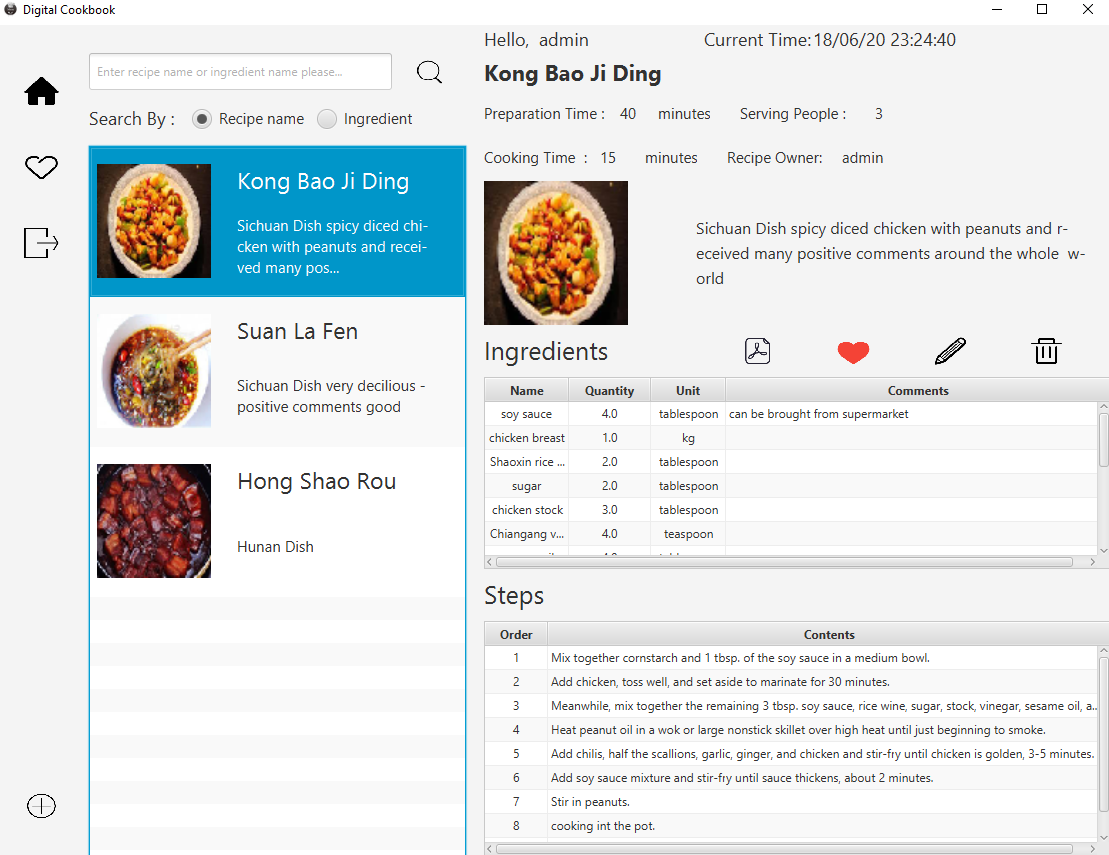
***➢* Registration page**



*Figure 3-2 User Registration Page*

When the new user presses the register button, the welcome page to switch to a registration page, in this page the user need to create a username and password. Re-Enter password helps to validate the consistency of password. If the consistency is not satisfied, the user cannot register successfully. With precondition, clicking on “register” button, the user can register successfully, and he can use this account to log in.

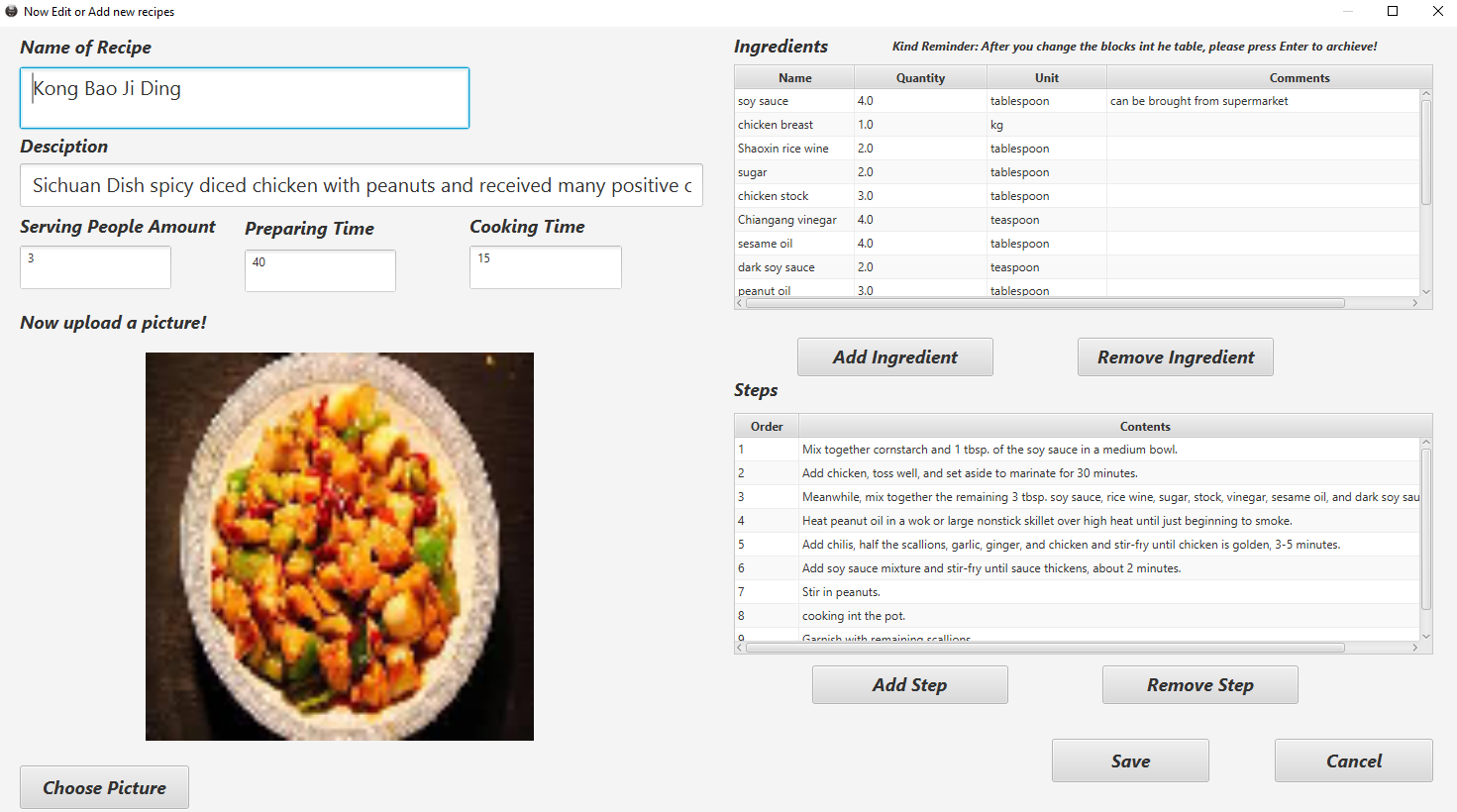
***➢* Main page**

****

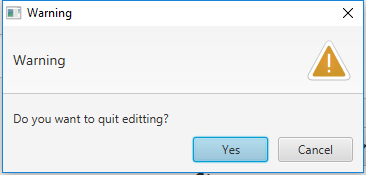
*Figure 3-3 Main Page*

When the user finishes registration, the application shows the main page. The left side is the main control bar, it is responsible for the whole structure of the application. The user can switch from any page to the main page or log out. The user can also add his own recipe by click on the “add” button. The middle compartment is searching page. The user can search for recipe either by recipe name or by ingredient, and the result will show. By choosing a specific recipe, the details will show on the right. The right compartment shows the contents for the recipe, as is shown above. If the user is the owner of this recipe, he can edit or delete his own recipe. But other users do not have authority to make these changes, in another word, these editing buttons will become invalid. Any user can set a recipe as his favorite and export the recipe as a PDF file.

***➢* Edit page**



*Figure 3-4 Edit & Add Recipe Page*

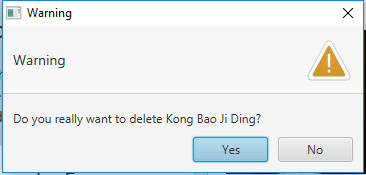


*Figure 3-5 Warning Alert on Edit Page*

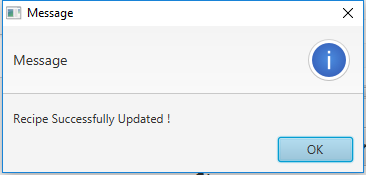
When the user clicks on the “add” button or “edit” button, this page will show. The user need to fill in the information shown above and provide a picture of the recipe. By pressing “save” button, the recipe will be added to the database and will be accessible to anyone use the application. By pressing “cancel” button, the user will leave the current page and all the edits will not be saved.

***➢* Warning / Error / Notifying Alters**

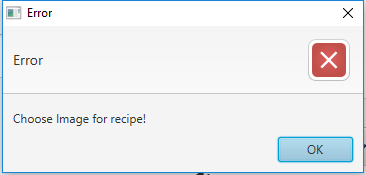
Warning alerts will be displayed when users want to make decisions such as confirm Edition on recipes; Error alters will be displayed when users made a mistake on specific area such as forget to add image for recipe; Notifying alters will be displayed after the users’ confirmation and no invalid input is detected, such as save recipes successfully.



*Figure 3-6 Warning Alert for deletion*



*Figure 3-7 Notification Alert for update recipes*



*Figure 3-8 Error Alert for not choose pictures for the recipe*

# **Software Test**

Junit is a unit testing framework for Java programming language, which is important in the development of test-driven development. The basic idea of Junit tests is to design a test method which compares expected results and real results using Assertions. In our Digital Cookbook project, we main use Junit 4 test framework (linking with its JAR file) to test and prove the correctness and running efficiency for all the methods in the package “de.fhluebeck.group3.DAO”.

# **Junit Test**

* + 1. **Junit Test Descriptions**

We used Junit test and we tested all classes in the package “DAO”, including “IngredientDAO.java”, “StepDAO.java”, “UserDAO.java” and “RecipeDAO.java”.

For “RecipeDAO.java”, we tested the get recipes, add/ delete/ edit recipe, find recipe by user methods, and add to favorite / remove from favorite methods. For IngredientDAO, we tested the get ingredients methods, add/ delete/ edit ingredient methods. And for “StepDAO.java”, we tested the get steps methods, add/ delete/ edit step methods. And for “UserDAO.java”, we tested the get steps methods, add/ delete user and validate password methods.

We used @Before annotation to setup before the tests, @Test is for methods that to be tested, and @After is for teardown the setup, we used it to recover the origin data.

* + 1. **Junit Test Results**

We have run the Junit tests successfully in all test classes and then we use EclEmma to analysis Junit test coverage. The coverage result the shown in the following graphs:

C:\Users\komgyu\Documents\Tencent Files\1547482652\Image\Group\5(OOC23A`N)$1YU3_BDHY[A.png

*Figure 4-1 Coverage report of StepDAOTest.java*



*Figure 4-2 Coverage report of RecipeDAOTest.java*



*Figure 4-3 Coverage report of IngredientDAOTest.java*



*Figure 4-4 Coverage report of UserDAOTest.java*

Generally, the overall coverage on the “DAO” package is more than 80%, which is a good sign that we have tested most of the methods.

# **Usability Test**

**4.2.1 Usability Test Descriptions**

Usability is a measure of how easy it is to use a product to perform prescribed tasks and it is a user-centered attribute of a good software. Therefore, the users must be included in the usability test process. So far, only the four group members are involved in developing the digital cook book, but the designers cannot represent the users, because they have taken part in the process of developing and they know exactly how things are going, namely, they have more intimate knowledge. In a result, the flaws can only be detected by testing the users.

***➢ Target Users and Sample Choice***

The target users should be people who understand English aging from 16 to 60. Besides, they should have basic computer skills to operate the software. The users’ devices should be able to run java programs. Hence, we decide to choose 8 students from FHL without professional software developing knowledge to help us to complete the test.

***➢ Methods and Goals***

There are five quality components of usability: learnability, efficiency, memorability, errors, and satisfaction. To test the five components, we decide to set tasks to our users and see how they accomplish these tasks without help by making records. Besides, we will design a questionnaire to the users. The questionnaire will focus on questions how the software looks like from a user’s perspective.

***➢ Tasks***

1. Open the software

2. Finish registration (register a new account with the valid username and password.)

3. Log in/ log out (with the new created account)

4. Search recipe by name/ingredient (e.g. shao, soy source and etc.)

5. See details of a recipe (click specific recipe)

6. Set / remove one recipe as / out of favorite (right click the heart button)

7. Create your own recipe (right click add button and add a new recipe like “Fan Qie Chao Dan”)

8. Edit your own recipe (edit only the newly created recipe)

9. Export one recipe as PDF file

***➢ Records***

1. The time the users need to accomplish every single task.

2. Record the amount of errors the users make during every task.

***➢ Questionnaire***

1. Do you think it is easy to accomplish these tasks?

A. Very easy B. Easy C. Normal D. Difficult

2. How do you like the layout of the interface?

A. Very clean B. Clean C. Normal D. Messy

3. How do you like the interface from an aesthetic way?

A. Very beautiful B. Beautiful C. Normal D. Ugly

4. Do you think the software is helpful in real life when you need a cooking navigator?

A. Very helpful B. Helpful C. Just so- so D. Not helpful

5. How many stars will you give to the software?

A. ★★★★★ B. ★★★★ C. ★★★ D. ★★ E. ★ F. None

6. In which way do you like the software?

A. Easy to operate B. Beautiful interface C. Multiple functions D. Satisfy my demand

7. Do you have any suggestions for this software?

* + 1. **Usability Test Results**

# **Equivalence Test**

According to our specification before, in our application there are four situations where we need to get input from the user. They are recipe search (by recipe name and by ingredient name), add recipe, edit recipe and user-login and registration. We implemented Equivalence tests for all the three situations.

**4.3.1. Equivalence Test for recipe search**

***➢ The EC class are defined as below:***

|  |  |  |
| --- | --- | --- |
| Tested Field | Valid EC Classes | Invalid EC Classes |
| The search bar for recipes | EC1 = {text | text includes letters, numbers and special characters} | EC2 = {text | empty text} |

*Figure 4.3.1 - 1 The EC Class for search bar*

***➢ The EC test results:***

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| EC1 | chicken | Giving all the recipes including the phrase ‘chicken’ |
| EC2 | empty | No filtered results and show all the recipe stored in the system. |

*Figure 4.3.1 - 2 The EC Class test results for search bar*

**4.3.2. Equivalence Test for add and edit recipes**

The input field for adding and editing recipes are the same and share the same interface so they are tested together.

***➢ The EC class are defined as below:***

|  |  |  |
| --- | --- | --- |
| Tested Field | Valid EC Classes | Invalid EC Classes |
| The name of recipe | TI1 = {text | Text includes letters, numbers and special characters; length <= 50} | TI2 = {text | Text includes letters, numbers and special characters; length > 50} |
| Brief Description | BD1 = {text | Text includes letters, numbers and special characters; length <= 100} | BD1 = {text | Text includes letters, numbers and special characters; length > 100} |
| Serving People | SP1 = {number <= 1000} | SP2 = {text | number > 1000} |
| Preparation Time | PT1 = {number <= 1000} | PT2 = {text | number > 1000} |
| Cooking Time | CT1 = {number <= 1000} | CT2 = {text | number > 1000} |
| Ingredient. Name | IN1 = {text | text includes letters, numbers and special characters; length <= 20} | IN2 = {text | text. length > 20} |
| Ingredient. Quantity | IQ1 = {number <= 100} | IQ2 = {text | number >100} |
| Ingredient. Units | IU1 = {text | text. length < 10 (units like ‘kg’ and ‘ml’ is strongly recommended)} | IU2 = {text | text. length > 10 or numbers} |
| Ingredient. Comments | IC1 = {text | text includes letters, numbers and special characters; length <= 50} | IC2 = {text | text includes letters, numbers and special characters; length > 50} |
| Step. Order | SO1 = {Integer} | SO2 = {text or double or other forms of data} |
| Step. Content | SC1 = {text | text includes letters, numbers and special characters; length <= 50} | SC2 = {text | text includes letters, numbers and special characters; length > 50} |

*Figure 4.3.2 - 1 The EC Class test results for add and edit recipes*

***➢ The EC test results:***

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| TI1 | ‘Fan Qie Chao Dan’ | Succeeded in this stage and the user can continue to the next steps |
| TI2 | Empty or texts more than 50 characters | Prompt Error messages and ‘Save’ button becomes unable |
| SP1 | 6 | Succeeded in this stage and the user can continue to the next steps |
| SP2 | ‘#yes@’ or 5000 | Prompt Error messages and ‘Save’ button becomes unable |
| BD1 | ‘This is one of the more popular dishes in China’ | Succeeded in this stage and the user can continue to the next steps |
| BD2 | Texts more than 1000 characters | Prompt Error messages and ‘Save’ button becomes unable |
| PT1 | 20 | Succeeded in this stage and the user can continue to the next steps |
| PT2 | ‘#yes@’ or 5000 | Prompt Error messages and ‘Save’ button becomes unable |
| CT1 | 30 | Succeeded in this stage and the user can continue to the next steps |
| CT2 | ‘#yes@’ or 5000 | Prompt Error messages and ‘Save’ button becomes unable |
| IN1 | ‘Soy Source’ | Succeeded in this stage and the user can continue to the next steps |
| IN2 | Texts more than 20 characters | Alert Error messages and do not archive the change |
| IQ1 | ‘3.0’ | Succeeded in this stage and the user can continue to the next steps |
| IQ2 | ‘#yes@’ or 5000 | Alert Error messages and do not archive the change |
| IU1 | ‘kilogram’ | Succeeded in this stage and the user can continue to the next steps |
| IU2 | 100 or Texts more than 20 characters | Alert Error messages and do not archive the change |
| IC1 | ‘can be obtained from the supermarket’ | Succeeded in this stage and the user can continue to the next steps |
| IC2 | Texts more than 50 characters | Alert Error messages and do not archive the change |
| SO1 | 1, 2, 3 | Succeeded in this stage and the user can continue to the next steps |
| SO2 | 2.0 or ‘#yes@’ | Alert Error messages and do not archive the change |
| SC1 | ‘Mix the eggs with salt together’ | Succeeded in this stage and the user can continue to the next steps |
| SC2 | Texts more than 50 characters | Alert Error messages and do not archive the change |

*Figure 4.3.2 - 2 The EC Class test results for adding and editing recipes*

***➢ The Boundaries:***

|  |  |  |
| --- | --- | --- |
|  | *Input* | *Output* |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**4.3.3. Equivalence Test for user-login and registration**

# **5.** **Evaluation**

# **Group Work**

**5.1.1. General**

***➢ Group Work***

We four as a group, hold regular internal meetings every week on Saturday and external meetings with the supervisor – Mr. Malte Grebe every week on Wednesday. Both of the two meetings cover the problems regarding the project, the feedbacks from the supervisor as well as the further plans for project development. Moreover, after the meetings, I - as the team leader - will always make arrangement about the future tasks for my teammates based on the previous work and evaluations.

Overall, we uniformly regard the discussion we had ourselves as well as the meeting with the supervisor crucial and helpful. By gathering together in a meeting, every member of the team will have the equal chance to express their ideas – including practical suggestions towards the project and feasible solutions towards previous bugs. Meanwhile, from the meeting with the supervisor, we can get useful suggestions from an experienced person who is very familiar with these systems, which helps us to avoid potentially wrong directions to go.

Regarding the tools for efficient team collaboration and the version control, we would like to recommend the GitHub. At the very first stage of our project, we have use the GitHub as the tool for version control. Every time we finish our part, we just have to commit and push new codes into remote server while the Merge step is done by the service provided by the GitHub, which is very convenient and always accurate. Although sometimes we may have the conflicts that is needed to be solved.

***➢ Work Distribution***

As a good team, we, of course, pay enough attention towards the work distribution – even distribution among team members, neither too relaxing nor too stressed. As the team leader, I create the project at the very first stage. Then we together make contributions about the Database design, different diagrams and project structure design. Then our tasks are distributed into specific areas – we have separated the user logic into four groups: User, Recipe, Step and Ingredient, each part includes the Model Class, the DAO Classes, GUI design and its corresponding Controller Class. For the details about the work distribution, please see Section 5.2.

***➢ Schedule Management***

The time management for the project is also well-arranged. Started in April in the Software Engineering class, we got the idea of this digital cookbook in the first 2 weeks gradually and then finished the specification. For the next 2 weeks, we reviewed the lessons learned in SWE I and made UML diagram, Use-case Diagram and E-R Diagram together. Then we discussed UML diagram with our supervisor and reviewed it and uploaded it a week later.

In the late May, we began to code according to the diagrams and specifications designed before. We started to work on different parts as soon as the teams are divided. We programed step by step and eventually the whole project. The time management of code is good, and we left 4 days to conduct Junit test. In the late June, we have already finished the coding part and started to write the report. Time for test is a little tight, but we finished in time.

**5.1.1. Individual Contributions**

***➢ Hua Yichen***

As the leader of the group, I am in charge of the whole structure and development progress of the project. To be more specific, I made the final decision on how our digital cookbook system will look like and what is the structure of system we will apply in our project (MVC).

After the early stage of decision, I created the project including all the packages and classes according to the designs and diagrams. Moreover, I also created the database and extracted the SQL commands that can be executed by my teammates. I myself mainly made code contributions to UserDAO.java, UserDAOTest.java, all the utilities, controllers and BaseDAO.java, while provided my teammates with practical solutions to their problems while programming, so you can see my contributions, more or less, on nearly every classes.

As a group leader, it is certainly my responsibility to distribute tasks to my colleagues. I set my teammates every week a task based on the time schedule of our project and set a reasonable deadline for them so that everybody is not too busy and not too lazy.

To synchronize the team work, I proposed our team to use GitHub and uploaded the primitive project with all packages and classes created, for other members to contribute to our project. Furthermore, I was also responsible for solving any errors or conflicts when we use the version control tool.

***➢ Kong Yu***

I – as a member of the tram – takes part in GUI, DAO and Junit Test. I design the view of editing recipe interface with scene builder in JavaFX. As to DAO, I am in charge of writing “StepDAO.java”. Besides, I write the Junit Test of “StepDAO.java”. Although I am not quite experienced in coding and developing software, I am trying to work and communicate with group members and finally make it.

***➢ Wang Jungang***

Generally speaking, my job is to finish all the classes that are relevant to the ingredients part. Firstly I build the ingredient table in database, and then the ingredient model, which is the basis of this project. After that I started coding on IngredientDAO. When IngredientDAO is finished, other members made a graph on GUI design. I helped adjusted some parts and wrote functions in “AddOrEditRecipeController.java”. After the development stage finished, I did Junit test on “IngredientDAO.java”.

***➢ Shan Jiaxiang***

In this program, Shan Jiaxiang is mainly responsible for “RecipeDAO.java” in DAO package, “recipeDAOTest.java” in Test package and the “|MainFrame.fxml” designing. He also made additional contributions on Class Diagram designing and Explanation, E-R Diagram designing and explanation and Use case Diagram drawing.

# **Task Responsibilities**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Hua Yichen | Kong Yu | Wang Jungang | Shan Jiaxiang |
| Controller.AddOrEditRecipeController.java | ○ | ● |  |  |
| Controller.BriefRecipeInformationController.java | ○ |  | ● |  |
| Controller.MainFrameController.java | ○ |  |  | ● |
| Controller.TemplateController.java | ● |  |  |  |
| DAO.BaseDAO.java | ● |  |  |  |
| DAO.ExportPDF.java | ● |  |  |  |
| DAO.RecipeDAO.java | ○ |  |  | ● |
| DAO.UserDAO.java | ● |  |  |  |
| DAO.StepDAO.java | ○ | ● |  |  |
| DAO.IngredientDAO.java | ○ |  | ● |  |
| entrance.CookbookApp.java | ● |  |  |  |
| model.Cookbook.java | ● |  |  | ○ |
| model.Ingredient.java | ○ |  | ● |  |
| model.Recipe.java | ○ |  |  | ● |
| model.Step.java | ○ | ● |  |  |
| model.User.java | ● |  |  |  |
| test.IngredientDAOTest.java | ○ |  | ● |  |
| test.RecipeDAOTest.java | ○ |  |  | ● |
| test.StepDAOTest.java | ○ | ● |  |  |
| test.UserDAOTest.java | ● |  |  |  |
| util.EncrypyUtil.java | ● |  |  |  |
| util.FileUtil.java | ● |  |  |  |
| util.StringUtil.java | ● |  |  |  |
| View.Template.java | ● |  |  |  |
| view.AddOrEditRecipeView.fxml | ○ | ● |  | ○ |
| view.BriefRecipeInformation.fxml | ○ |  | ● | ○ |
| view. MainRecipeFrame.fxml | ○ |  |  | ● |
| view. Template.fxml | ● |  |  | ○ |

*Table 5-1 Group Contribution*

In this table, “●” represents the member was responsible for the file and did the main job and “○” represents the member helped or did an assistant contribution for the file.

# **Acknowledgement**

[1] Icon of buttons including “Export PDF”, “Home”, “Logout”, “Like it”, “Delete Recipe”, “Edit Recipe” and “Add Recipe” is downloaded from Alibaba “Icon Font” website: http://www.iconfont.cn/

[2] Presenting image of the preface of this project report is from the passage “Cookbook Village”: <https://www.cookbookvillage.com/blogs/cookbook-collecting/5247962-top-10-most-collectible-cookbooks>

[3] We have learned from the source codes from seniors group “Kitchen Socket Team” (group members: Chen Sijie, GU Qiwen, Liu Yuefeng, Shao Gang and Shi Wenbin) as well as inspired from them about the design and layout of User Interface.