



**MIDDLE EAST TECHNICAL
UNIVERSITY**



Software Engineering
SE 560 Fall 2008

RECIPE REPOSITORY SYSTEM

DESIGN REPORT

December 14, 2008

Fikret Selvi-H.Özgün Polat

Computer Engineering Department
Middle East Technical University

TABLE OF CONTENTS

1.Introduction.....	4
1.1. Project Definition.....	4
1.2.Goals and Scope.....	4
1.3.Constraints.....	5
2.SoftwareRequirements.....	6
2.1. Functional Requirements.....	6
2.1.1. Registration.....	6
2.1.2. User Requests.....	6
2.2. Non-Functional Requirements.....	7
2.2.1. Usability.....	7
2.2.2. Performance.....	7
2.2.3. Scalability.....	7
2.2.4. Security.....	7
2.3. Hardware Requirements.....	8
2.4. Software Requirements.....	8
3. Modeling.....	9
3.1.Data Model	9
3.1.1Entity Relationship Diagrams (ERD).....	10
3.1.2 Data Description.....	10
3.1.3 Explanation of Relations.....	12
3.1.3.1 <i>CategoryList</i> relation between Category and Recipe.....	12
3.1.3.2 <i>SeasonList</i> relation between Season and Recipe.....	12
3.1.3.3 <i>CuisineList</i> relation between Cuisine and Recipe.....	12
3.1.3.4 <i>PreparationMList</i> relation between PreparationMethod and Recipe.....	12
3.1.3.5 <i>RecipeBox</i> relation between User and Recipe.....	13
3.1.3.6 <i>Ingredientt</i> relation between Edible,Unit and Recipe.....	13
3.1.3.7 <i>Comment</i> relation between User and Recipe.....	13
3.1.3.8 <i>SearchRating</i> relation between Recipe and Recipe.....	14
3.1.3.9 <i>RecipeRelation</i> relation between Recipe and Recipe.....	14
3.1.3.10 <i>Edible Relation</i> relation between Edible and Edible.....	14
3.2. Pseudo Code.....	15
3.3. Use Case.....	16
3.4. Class Diagram.....	17
3.5. Class DiagramExplanation.....	19
3.6. Sequence Diagram.....	22

4. Project Organization.....22

4.1. Schedule.....22

1. Introduction

1.1. Project Definition

In this project, we will develop a web-based application to archive and retrieve recipes of any kind from a pool of ingredients and recipes. The systems aims to meet following main requirements:

- Users will access the Receipe Registration System by using a computer which is connected to the Internet and has a web browser.
- Before logging into system, the user will be able to
 - o Retrieve a recipe through simple searching its name, ingredients, keywords or advanced searching.
 - o View the recipe
- After logging into system, the user will be able to
 - o If the category of the recipe is not available on the system, user should cerate it
 - o If the recipe is not available on the system ,user should insert it.
 - o If the recipe is available on the sysytem, user should add comments and advices.
 - o If the ingredients are not available on the system, user should create its.
 - o Get information about users.
 - o View their recipe box.
 - o Update about their information or recipe box.

1.2. Goals and Scope

Time is evolving and users are becoming more dependent on the web-based technologies, since it is faster and efficient than the client applications. With the time, technology is also evolving gradually. Software technologies are trying to serve faster and solve the crisis on the demand from the clients.

Current technologies are competing with each other in such a way that they develop systems more usable, robust, user friendly, effective and attractive systems rather than the performance. The reason is most software solved the performance issue with the internet based systems for the moderate systems and tries to concentrate on the other issues as well. This is not valid for the real time systems obviously.

In the market, there are products for the retrieval of any recipes of any kind. As the content is complex and large, the related web sites concentrated on the attractiveness of the pages and be informative for the user. The goal is to understand the aim of the system and visualize the walkthrough of the system easily. Users are familiar with the

subject already, but the system should direct the user to explore the system and navigate within the recipes.

Although the goal of the system is to produce such a system to satisfy the requirements and give user ability to navigate through the recipes in a tree based structure and give them the ease of recording recipes in a recursive form. That means any recipe can include the details of any other recipe and that goes on in a recursive manner not to archive same recipes and ingredients again.

Since the current products on the internet are not too much scalable and flexible on archiving recipes, it's a good point to start from now on. Most recipe repository systems based their tools upon retrieval. That's why the scope is to enhance the features that are deficient in the previous products and add more functionality that satisfy the requirements of the recipe customers.

1.3. Constraints

At first sight, it sounds as if it's a typical repository system where user inserts records, delete, modify and query them. However, the necessity of dependency and integrity of the recipes makes the system more complex than it is imagined.

One of the obstacle the repository system faces is the complexity of the searching mechanism. The requirement says that the recipes include the other recipes as an ingredient also. That makes the structure to be built up in a tree based form, which is also complex to search. Because besides the recipe content, the ingredients are also in a tree based form and a recipe has many variables to define it clearly. In order to search for a recipe, users have many options to search for and when the user uses a keyword, s/he aims to target the correct results. That's why it needs a well defined structure.

Another obstacle is the control of every new categories, recipes and records by the moderator. The user may carelessly ask for a new category that is already included within the system. That's why; the categories should be designed and defined clearly. Then, the moderator will not deal with the replacement of the categories and recipes existing within the recipe repository system.

It's not easy to handle the difficulties with a dynamic system. A dynamic system is the one where at least one door is open for the defects. In software quality assurance, the important thing is not detecting defects but preventing the defects. The heart of the prevention is the strength of the control mechanisms. It's not a real time system but it may be a financial system. Since there are competitive systems currently working, a small error or obstacle in the system can lead the customer to use other systems and stop using the product. That would be the greatest harm for the product. That's why the constraints should be handled properly and carefully.

2. Software Requirements

2.1. Functional Requirements

Here are some crucial functional requirements for the system:

2.1.1.Registration

Registration should be required for the users of the system, to be able to keep track of information about them (such as number of users, their professions etc...) which can be used to gather statistical data, which can be useful. Also it will allow us to communicate with the users via e-mail and get some feedback about the system.

2.1.2. User Requests

2.1.2.a.

Users will access the Recipe Repository System by using a computer which is connected to the Internet and has a web browser.

2.1.2.b

Before logging into system, the users view the recipes under the category and season,

2.1.2.c.

Before logging into system, the users view the daily recipe.

2.1.2.d

Before logging into system, the users can do simple search by insert text.

2.1.2.e.

Before logging into system, the users can do advanced search by select category, season, cuisine, ingredients and preparation method.

2.1.2.f.

After logging into system, the users will be able to view/update his/her personal information

2.1.2.g.

After logging into system, the users will be able to view/update his/her recipe box.

2.1.2.h.

Each recipe in the system will be placed under a category. These categories will be predefined, but the system will be flexible enough to allow the addition of new categories. When a user needs a category for a new recipe (after logging into system), a

request will be adding that category to the system. When the category is added, the recipe will be placed under that category.

2.1.2.i.

Each recipe in the system will be placed under a season. These seasons will be predefined, but the system will be flexible enough to allow the addition of new season. When a user needs a season for a new recipe(after logging into system), a request will be adding that it to the system. When the season is added, the recipe will be placed under that it.

2.1.2.j

Some recipe in the system will be placed under a cuisine. These cuisine will be predefined, but the system will be flexible enough to allow the addition of new cuisine. When a user needs a cuisine for a new recipe(after logging into system), a request will be adding that it to the system. When the cuisine is added, the recipe will be placed under that it.

2.1.2.k

Some recipe in the system will be placed under a preparation method. These methjod will be predefined, but the system will be flexible enough to allow the addition of new method. When a user needs a method for a new recipe(after logging into system), a request will be adding that it to the system. When the method is added, the recipe will be placed under that it.

2.1.2.l

The same walkthrough also applies for ingredients. They will also be predefined but it will also be possible to add new ones.(after logging into system)

2.1.2.m

When a user wants an advise for any recipe(after logging into system),the advise will be viewed under that recipe.

2.2.Non-Functional Requirements

2.2.1. Usability

Usability is one of the most important quality factors for this system. Especially on the web, variety of web sites exists in competition, and users often prefer the most usable, easy to use, and user-friendly one. Therefore usability becomes more an important factor to be able to compete with similar other products on the market.

2.2.2. Performance

Performance is another factor affecting the quality of the system. In fact, there should not be strict performance requirements since it is not a real-time or a performance

critical system. But the interaction performance of the system must be good enough to prevent any kind of annoyance. For example if the search results take too long to come out, or the system responds slower than the user expects, it will leave a bad impression on the user about the system.

2.2.3. Scalability

Scalability is very important on this system, since the possible number of users is not limited practically. Since what the system offers is something too general, the system can be used by a very wide range of users. Therefore special care needs to be taken regarding scalability.

2.2.4. Security

Usually security becomes important in online shopping systems. These sites most of the time gets credit card information of the user for the billing process. In these cases encrypted data is sent and received through the wires. In our system, there is no need for additional security since there is no selling or buying. But system security is essential and special care needs to be taken to protect the application against attacks since they happen in the web environment frequently.

2.3. Hardware Requirements

Processor: 600-megahertz (MHz) Pentium III-compatible or faster processor; 1-gigahertz (GHz) or faster processor recommended

Memory: 512 megabytes (MB) of RAM or more; 1 gigabyte (GB) or more recommended

Drive: CD-ROM or DVD-ROM drive

Display: Super VGA (1024x768) or higher-resolution video adapter and monitor

2.4. Software Requirements

Target Platforms:

Microsoft Windows 2000 Server™ with Service Pack 4

Microsoft Windows XP Professional™ with Service Pack 2

Microsoft Windows Server™ 2003

Minimum Software:

Microsoft Internet Explorer 6.0 with Service Pack 1.

Microsoft Data Access Components (MDAC) 2.8

Windows Installer version 3.0

.NET Framework 2.0

Internet Information Services (IIS)

Microsoft SQL Server™ 2005

3.Modeling

3.1.Data Model

3.1.1Entity Relationship Diagrams (ERD)

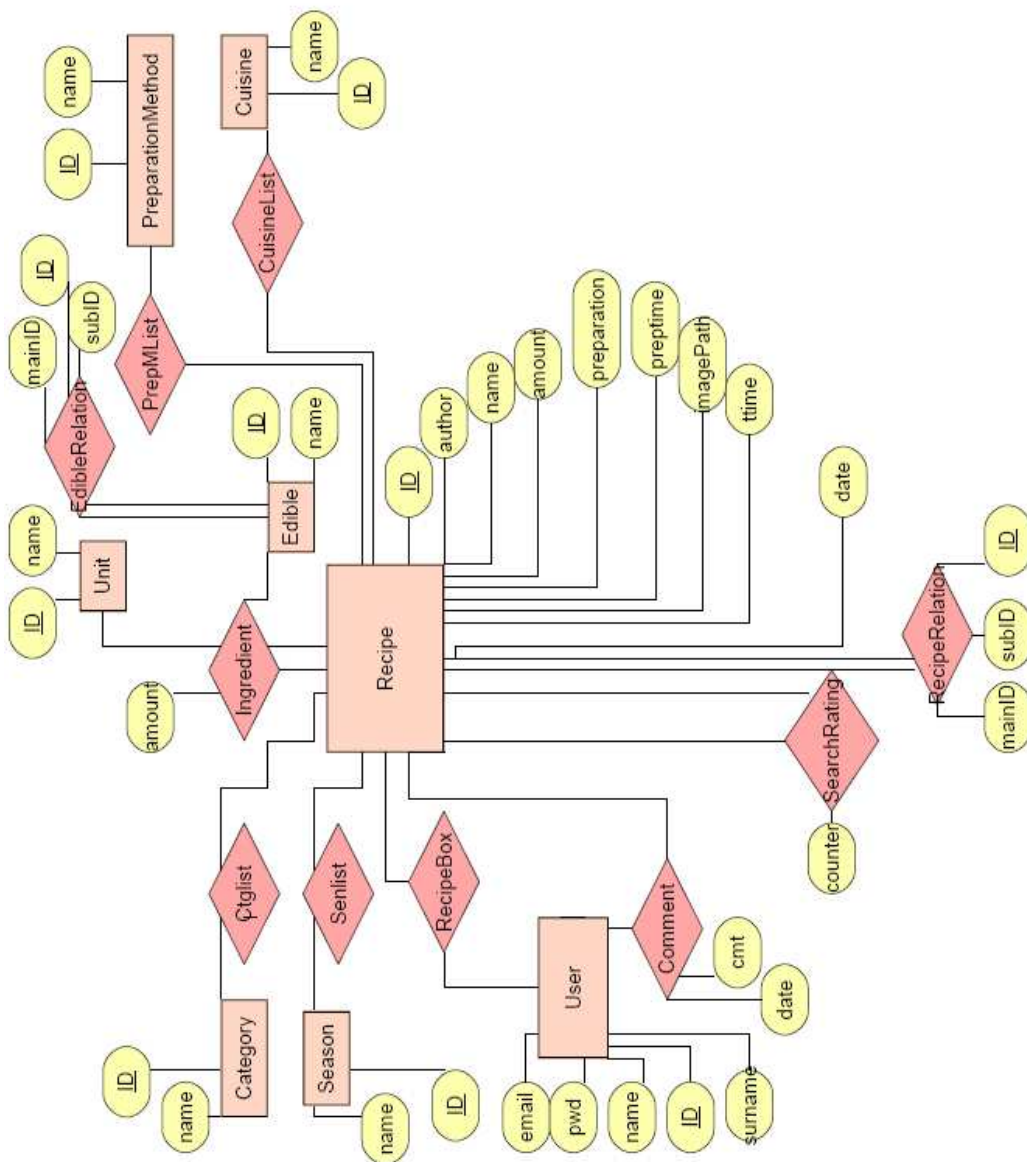


Figure 1.ER Diagram

3.1.2 Data Description

User:

Type	Entity
Attributes	
userid	Unique primary key for user
pwd	Password of the user
email	Email of the user
name	Name of the user
surname	Surname of the user

Season:

Type	Entity
Attributes	
ID	Unique primary key for season
name	Name of the season (ex:thanksgiving,christmas)

Category:

Type	Entity
Attributes	
ID	Unique primary key for category
name	Name of the category(ex:desserts,chicken recipes)

Cuisine:

Type	Entity
Attributes	
ID	Unique primary key for cuisine
name	Name of the cuisine (ex:French,Italian)

PreparationMethod:

Type	Entity
Attributes	
ID	Unique primary key for preparation method
name	Name of the preparation method (ex:gril,bake,fry)

Edible:

Type	Entity
Attributes	
ID	Unique primary key for edible
name	Name of the edible(ex:egg,rice,onion)

Recipe:

Type	Entity
Attributes	
ID	Unique primary key for recipe
name	Name of the recipe
image	Photo of the recipe
date	Date of the inserted of the system
amount	Quantity of the recipe
rate	Number of viewed of the recipe
preptime	Preparing time of the recipe
ttime	Total time of the recipe ready in
author	Name of the recipe adder
preparation	Explanation of the recipe

Unit:

Type	Entity
Attributes	
ID	Unique primary key for unit
name	Name of the unit(ex:gram)

3.1.3 Explanation of Relations

3.1.3.1 CategoryList relation between Category and Recipe

When users select any category item, the users will be able to view a recipes which relation to selected item.

Type	Relation
Attributes	
ctgID	Foreign key for category
rcplD	Foreign key for recipe

3.1.3.2 SeasonList relation between Season and Recipe

When users select any season item, the users will be able to view a recipes which relation to selected item.

Type	Relation
Attributes	
sID	Foreign key for season
rcplD	Foreign key for recipe

3.1.3.3 CuisineList relation between Cuisine and Recipe

When users select any cuisine item, the users will be able to view a recipes which relation to selected item.

Type	Relation
Attributes	
csnID	Foreign key for cuisine
rcplD	Foreign key for recipe

3.1.3.4 PreparationMList relation between Preparation and Recipe

When users select any method item, the users will be able to view a recipes which relation to selected item.

Type	Relation
Attributes	
preplD	Foreign key for preparation method
rcplD	Foreign key for recipe

3.1.3.5 *RecipeBox* relation between User and Recipe

Users add a recipe which they like in a recipe box ,thus making it possible to view it without searching.

Type	Relation
Attributes	
userID	Foreign key for user
rcplD	Foreign key for recipe

3.1.3.6 *Ingredient* relation between Edible Recipe and Unit

The relation includes ingredients of the recipe needs

Type	Relation
Attributes	
edibleID	Foreign key for edible
rcplD	Foreign key for recipe
unitID	Foreign key for unit
amount	Amount of edible in recipe

3.1.3.7 *Comment* relation between User and Recipe

Users will add a comment for any recipe.

Type	Relation
Attributes	
userID	Foreign key for user
rcplD	Foreign key for recipe
cmt	Comment about recipe
date	Date

3.1.3.8 *SearchRating* relation between Recipe and Recipe

The relation includes count of the recipe views.

Type	Relation
Attributes	
counter	The number of the recipe viewed.
rcplD	Foreign key for recipe

3.1.3.9 *RecipeRelation* relation between Recipe and Recipe

The relation includes main recipe and its sub recipes.

Type	Relation
Attributes	
ID	The unique primary key for relation
mainID	key for main recipe
subID	keys for sub recipe in the main recipe

3.1.3.10 *EdibleRelation* between Edible and Edible

The relation includes main edible and its sub edibles.

Type	Relation
Attributes	
ID	The unique primary key for relation
mainID	key for main edible
subID	keys for sub edible in the main edible

3.2. Pseudo Code

3.2.1. Search Algorithm

- FOR all criteria entered by user
 CALL Fitler.add() with value of Criteria and Column
Name
- CALL DBFactory.select() with Fitler and Recipe Table
- COMPUTE Recipes' ratings as the return of method
SearchRating.getByRecipeID
- SORT the Recipes using their ratings
- RETURN the results

3.3. Use Case

In the figure below, you can see the use case diagram for the actor user.

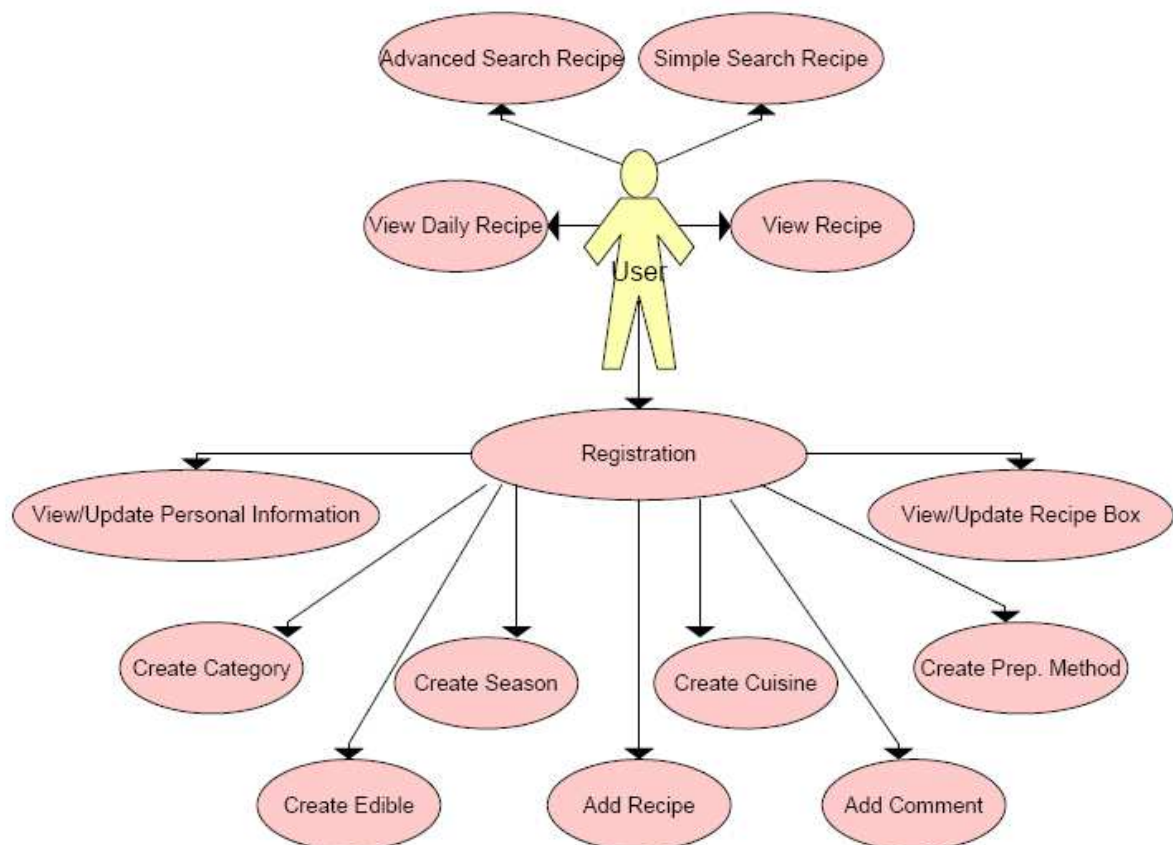


Figure 2.Use_Case Diagram

In this diagram, we can see the possible operations performed by the system with the interaction of the user. A user can add a recipe operation such as adding a new recipe or modifying a recipe. Also user request add a new category, season, edible, cuisine, preparation method. In addition, it is also possible for a user to view a recipe and search for recipes. A user may also want to comment to recipe. Also user can view or update his/her personal information and recipe box.

3.4. Class Diagram

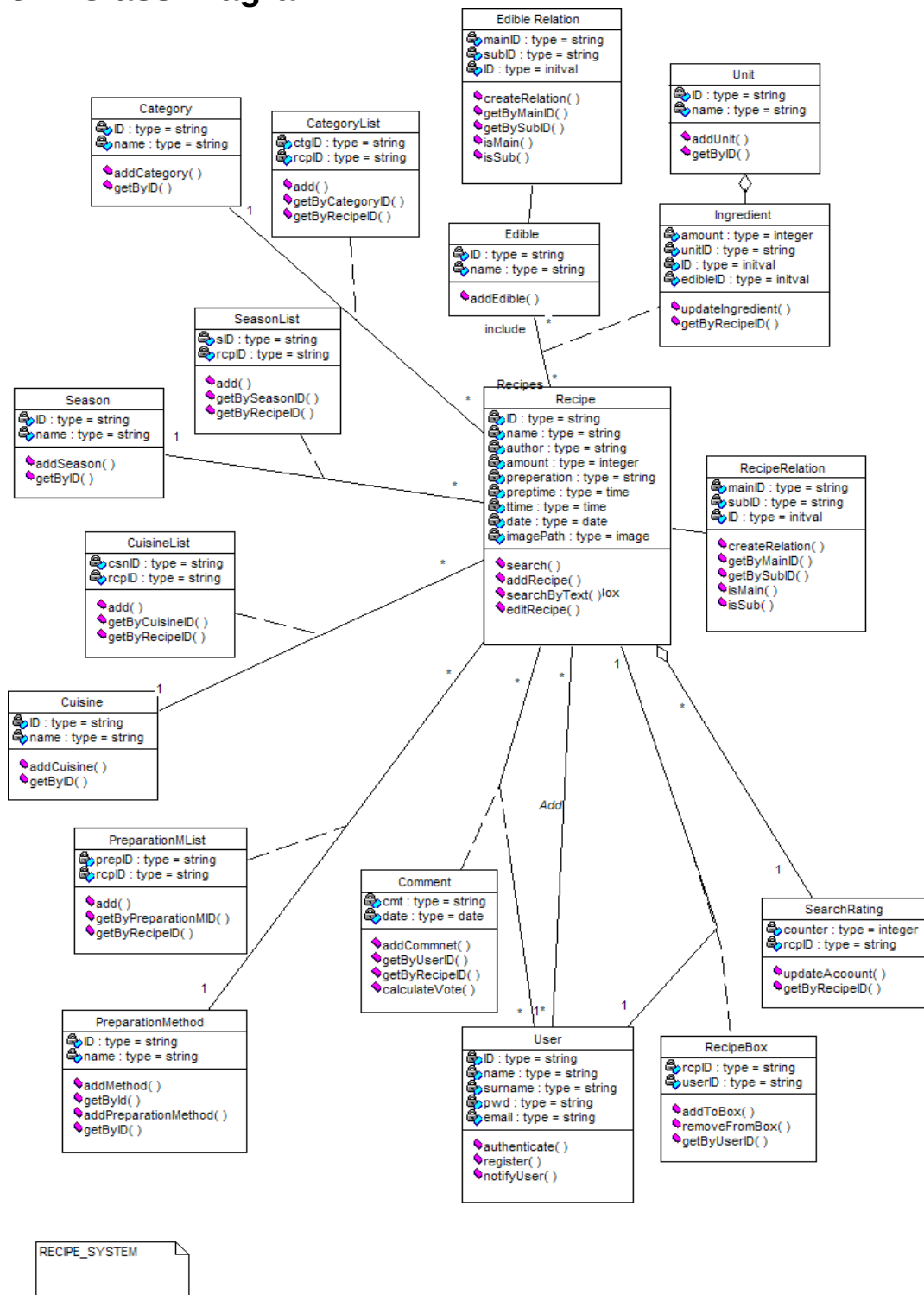


Figure 3 .Class Diagram1

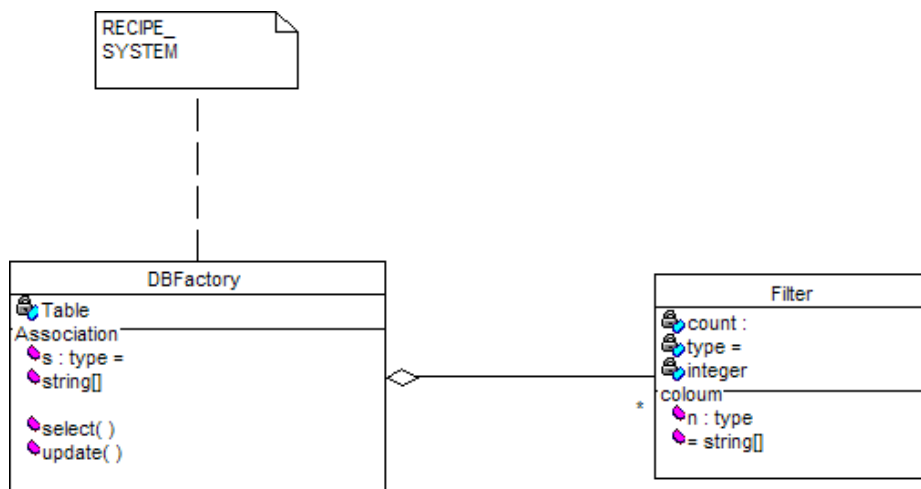


Figure 4.Class Diagram2

3.5. Class Diagram and Method Explanations

Category Class:

addCategory() : This method gets a Category object as an argument and inserts it to the database table as a row

getById(): Input of the method is an ID and returns the name of the Category.

CategoryList Class:

add(): Creates a Category – Recipe relation by using their ID's.

getByCategoryId(): Returns the ID of the Recipe by using the Category ID.

getByRecipeID(): Returns the ID of the Category by using the Recipe ID.

Comment Class:

addComment(): Calls when the user wants to add a comment to a recipe. Creates an - Comment typed- object and adds it to the database table as a row.

getUserID(): Returns all comments that added by the input User ID.

getByRecipeID(): Returns all comments that is relational with the input Recipe ID.

calculateVote(): Calculates an average vote for the given Recipe ID. This function can be used to define the popularity of a recipe.

Cuisine Class:

addCuisine(): Inserts a new row to the database table that is defined by user.

getById(): Returns the Cuisine's name with the given ID.

CuisineList Class:

add(): Creates a Cuisine – Recipe relation by using their ID's.

getByCuisineID(): Returns all recipes that is related with the given Cuisine ID.

getByRecipeID(): Returns the cuisine that is related with the given Recipe ID.

DBFactory Class:

select(): Generates an Select SQL Query for the given table and filter.

insert(): Generates an Insert SQL Query for the given table and filter.

delete(): Generates an Delete SQL Query for the given table and filter.

update(): Generates an Update SQL Query for the given table and filter.

findTableName(): Returns the name of the table by using the object's class definition.

Edible Class:

addEdible(): Inserts a new row to the database table that is defined by user.

getById(): Returns the Edible's name with the given ID.

EdibleRelation Class:

createRelation(): Creates a Main-Sub relation between to Edible objects.

getByMainID(): Returns all the Edible ID's those are the sub ID of the argument.

getBySubID(): Returns all the Edible ID's those are the main ID of the argument.

isMain(): Returns a boolean value for the given Edible ID that shows the Edible is a sub object or not.

isSub(): Returns a boolean value for the given Edible ID that shows the Edible is a sub object or not.

Filter Class:

add(): Adds a new criteria for the SQL by getting the value and the column name.

Ingredients Class:

updateIngredients(): Adds, deletes or updates the Edible list of a Recipe.

getByRecipeID(): Returns all Ingredients that is related with the given Recipe ID.

PreparationMethod Class:

addMethod(): Creates a new Preparation Method which is defined by user.

getByID(): Returns the name of the method by using it's ID.

PreparationMethodList Class:

add(): Creates a PreparationMethod – Recipe relation by using their ID's.

getByPreparationMethodID(): Returns all recipes that is related with the given Preparation Method ID.

getByRecipeID(): Returns the Preparation Method that is related with the given Recipe ID.

Recipe Class:

search(): The advanced search method that gets the criterions, which are defined by user, as input and finds the results with showing them in a suitable order.

searchByText(): The simple search method that gets only a keyword as a input and returns the results.

addRecipe(): Defines a new Recipe object and inserts it to the database table as a row.

editRecipe(): This method is called when a user edits a recipe.

RecipeBox Class:

addToBox(): Adds the choosen recipe to the user's shortcut list.

removeFromBox(): Removes the choosen recipe from the user's shortcut list.

getUserID(): Returns all recipes that is in the list of the User by using the User ID.

RecipeRelation Class:

createRelation(): Creates a Main-Sub relation between to Recipe objects.

getByMainID(): Returns all the Recipe ID's those are the sub ID of the argument.

getBySubID(): Returns all the Recipe ID's those are the main ID of the argument.

isMain(): Returns a boolean value for the given Recipe ID that shows the Recipe is a sub object or not.

isSub(): Returns a boolean value for the given Recipe ID that shows the Recipe is a sub object or not.

SearchRating Class:

updateCount(): Updates the value of the count when a recipe is chosen after a search.

getByRecipeID(): Returns the value of the count that is related with the given Recipe ID.

Season Class:

addSeason(): Inserts a new row to the database table that is defined by user.
getByID(): Returns the Season information by using it's ID.

SeasonList Class:

add(): Creates a Season – Recipe relation by using their ID's.
getBySeasonID (): Returns all recipes that is related with the given Season ID.
getByRecipeID (): Returns the cuisine that is related with the given Recipe ID.

Unit Class:

addUnit(): Inserts a new row to the database table that is defined by user.
getByID(): Returns the Unit information by using it's ID.

User Class:

authenticate(): Login the
register(): Creates a new User Object and inserts it to the database.
notifyUser(): Notifies User by sending a e-mail. Called when a registration completes, a recipe of the user edited by another user or a comment made to a recipe of the user.

3.6. Sequence Diyagrams

3.6.1.Search Sequence Diagram

Diagram shows the recipe search. User insert a text in label. This nformation takes and add the fitler and DBFactory can do database functions.

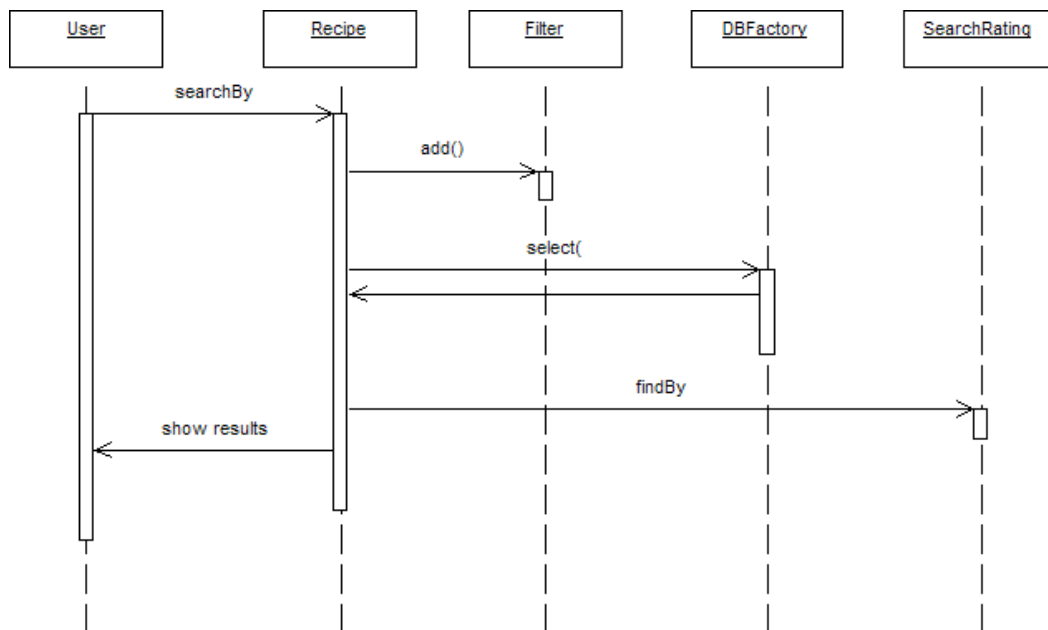


Figure 5.Search Sequence Diagram

3.6.2.Racipe Box Sequence Diyagram

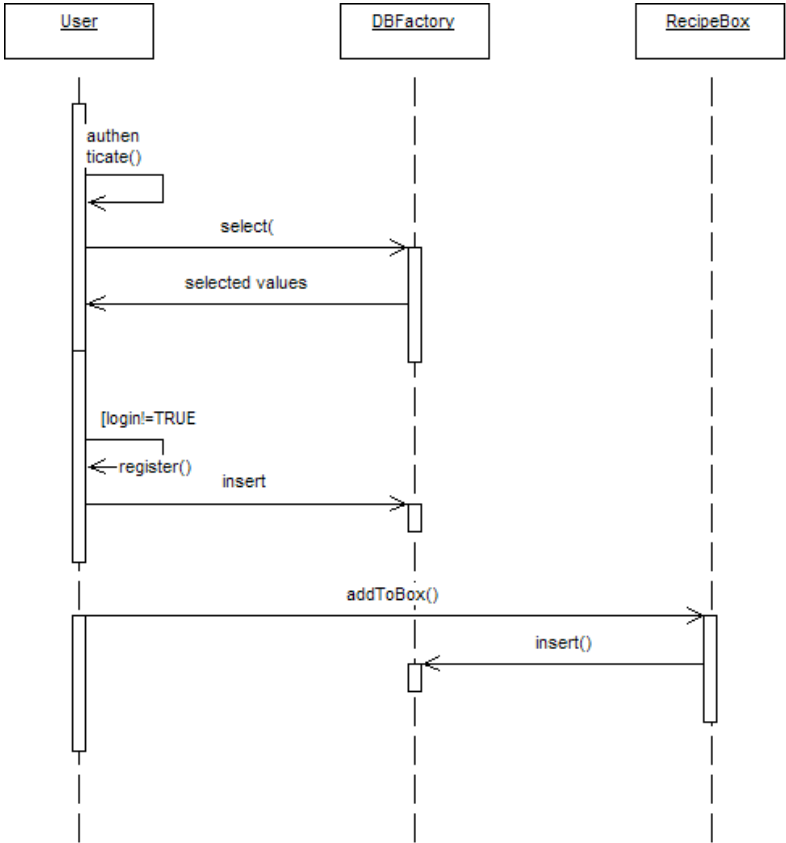


Figure 6.Recipe Box Sequence Diagram

6. Project Organization

6.1. Schedule

The Gantt chart of the project is depicted as follows: (drawing with [RFFlow 5.04](#))

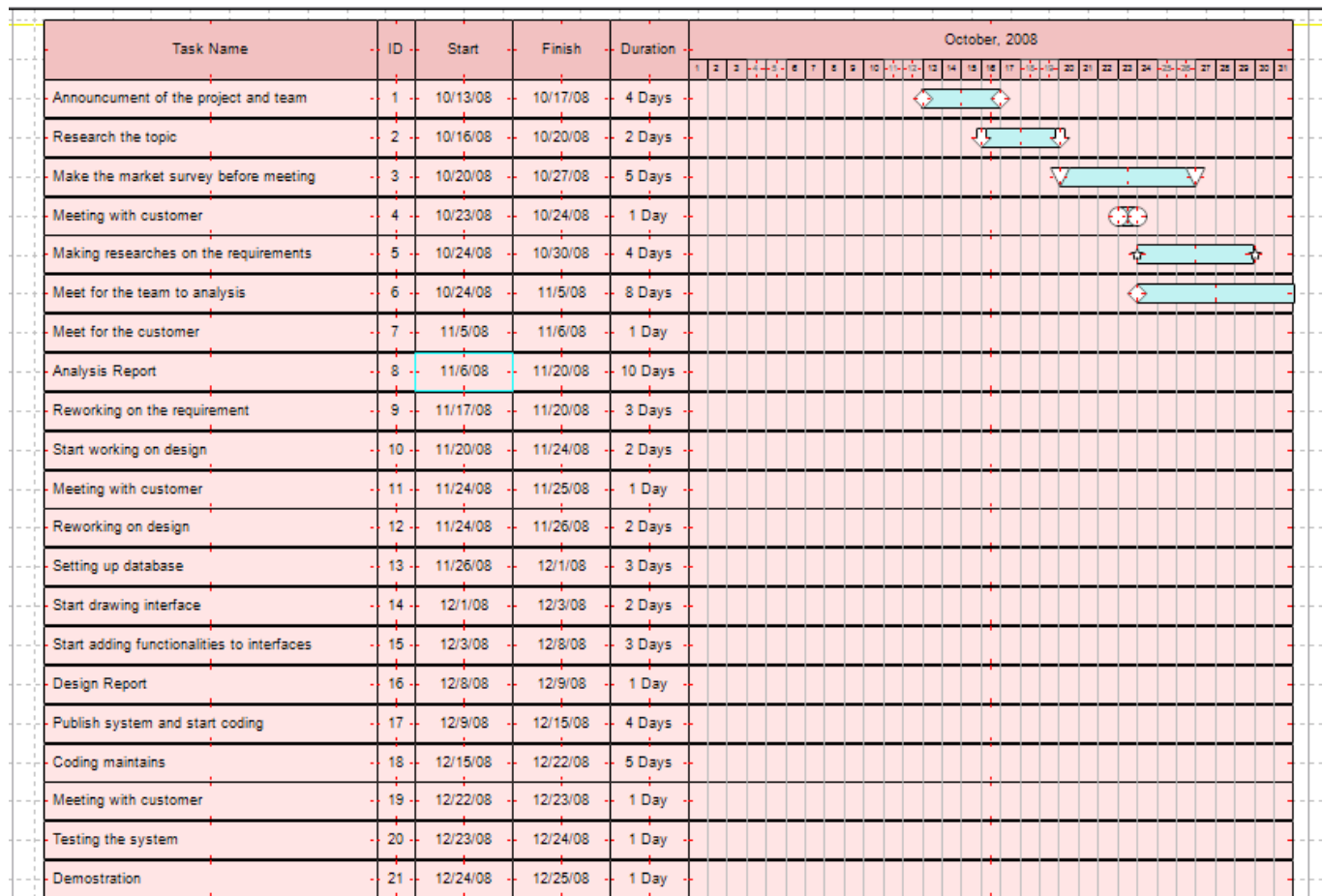


Figure 10.1: Gantt chart

