

Software Design Description (SDD)

Recipe Repository System

Project Members:

Emine CIMEN
Abdullah OZTURK

TABLE OF CONTENTS

1. INTRODUCTION	4
1.1 Project Definition	4
1.2 Goals and Scope	4
1.3 Design Constraints	4
2. ARCHITECTURAL DESIGN	5
2.1 Deployment Diagram	5
3. USE CASE DESIGN	7
3.1 User profiles	7
3.2 Use-cases	7
3.2.1 Unregistered User	7
3.2.2 Registered User	8
3.2.3 All Users	11
4. DATA DESIGN	14
4.1 Entity Relationship Diagram	14
4.1.1 Entities	14
4.1.2 Relationships	15
4.2 Data Dictionary	15
5. MODELING	18
5.1 Class Diagram	18
5.2 Sequence Diagram	25
5.2.1 Add New Recipe	25
5.2.2 Display All Recipes	26
5.2.3 Search for a Recipe	27
5.2.4 Edit an Ingredient	28
6. USER INTERFACE DESIGN	29
6.1 Home Menu	29
6.2 Recipes Menu	29
6.3 Ingredients Menu	31
6.4 Search Menu	33
7. PROJECT PLAN	36
7.1 Gantt Chart	36

FIGURES

Figure 1 Deployment Diagram	5
Figure 2 Actors	7
Figure 3 Register to the system use case	8
Figure 4 Registered user use cases	9
Figure 5 All users use cases	12
Figure 6 ER Diagram	14
Figure 7 Class Diagram	18
Figure 8 Add new recipe sequence diagram	25
Figure 9 Display all recipes sequence diagram	26
Figure 10 Search for a recipe sequence diagram	27
Figure 11 Edit an ingredient sequence diagram	28
Figure 12 Home page	29
Figure 13 Add/Edit recipes page	30
Figure 14 Show recipes page	31
Figure 15 Add/Edit ingredients page	32
Figure 16 Show ingredients page	33
Figure 17 Search recipes by its name page	34
Figure 18 Search recipes by ingredients page	35
Figure 19 BFS tree search algorithm	37

1. Introduction

This introductory part of the Software Design Description (SDD) provides an overview of the entire document.

1.1 Project Definition

The aim of Recipe Repository System project is to develop a recipe repository system which manages the recipes of all kinds. This tool makes it possible for the user to perform operations on the recipes for both editing and displaying purposes. The system will have two kinds of users as registered and unregistered users. Editing type of actions can only be performed by registered users.

The main functionalities, goals and objectives of the product can be found in the section that follows.

1.2 Goals and Scope

As stated before, the ultimate goal of the Recipe Repository System is to store recipes and other data elements entered by users and providing related operations to display, search and manage them. To be able to achieve this goal, the tool offers a lot of features to the users. The main objectives of the system can be stated as follows:

- The tool lets the user to display recipe information (its name, description and ingredients)
- The tool makes it possible to create new recipes
- The tool provides a way to add new ingredients to a recipe and it also makes it possible to add a recipe as a ingredient to it
- The tool lets the user to add a new ingredient to the ingredients list
- The tool provides the facility for searching for a recipe by entering its name or some of its ingredients
- The tool makes it possible to display all of the ingredients independent of the recipes
- The tool lets the user to add unit of measurements to the corresponding list

1.3 Design Constraints

To be able to work efficiently, satisfying the requirements imposed, Recipe Repository project should be carefully designed. However, there are some design constraints which should be taken into account while designing the system.

Recipe Repository project is a web-based system and that adds an overhead because of the possible problems with the Internet connection. To be able to minimize the effect of this overhead, the communication within the system modules should be minimized avoiding the unnecessary interactions that can further delay processing.

Recipe Repository system heavily interacts with the database behind it. Nearly all the necessary information for processing is maintained in the database. There is a heavy load of fetching/storing data from/to the database. This makes the efficiency of the DBMS an important constraint that must be taken into account seriously. An efficient DBMS should be used and the database should be carefully designed, preventing any unnecessary burden put on the DBMS. Also the queries should be designed efficiently to minimize the cost of database operations. By this way, the overhead caused by the DBMS can be minimized.

2. Architectural Design

2.1 Deployment Diagram

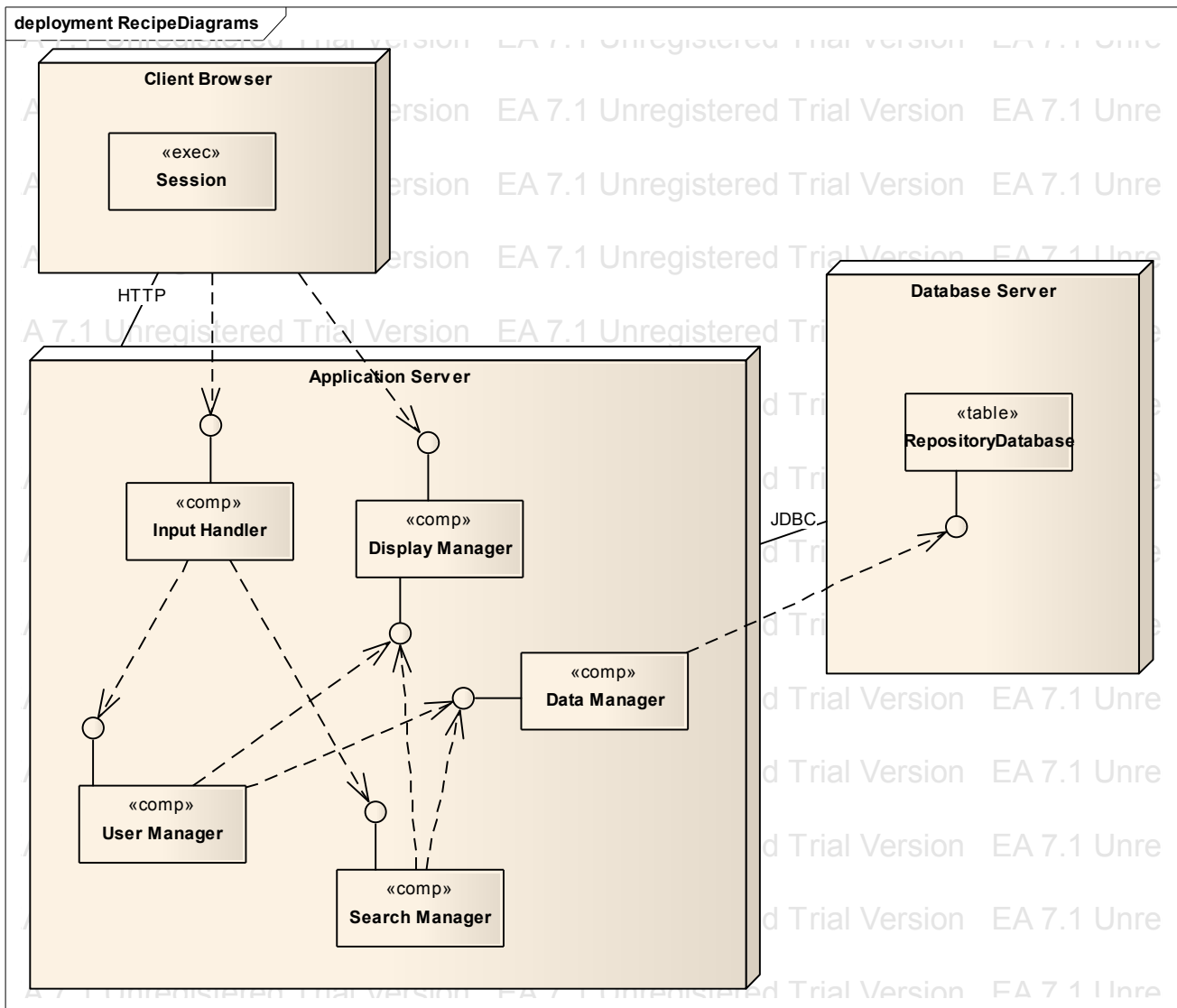


Figure 1 Deployment Diagram

Recipe Repository system consists of 3 main nodes, namely the browser, the application server, and the database server. Each node consists of a number of components that declares interfaces and the components communicate among themselves by means of the methods in the interfaces.

Browser works on the user side. It communicates with the application server through HTTP protocol. It consists of the Session component, which connects to the other components working on the application server through the interfaces they declare.

In the application server, the main components that constitute the functionality of Recipe Repository sit. Firstly, Input Handler component receives user inputs and forwards them to other related components. User Manager component manages the registered users, their sessions and the records owned by them. Search Manager component handles the queries created by the user and finds the relevant data effectively by using the search algorithm.

Display Manager component prepares the appropriate pages with the data provided resulted from user inputs. Finally, Data Manager component provides access to the DBMS and provides required data from database or commit user entered data to it.

The relationships between components are described as follows. Input Handler component forwards user inputs to User Manager and Search Manager. Searches are handled by Search Manager; however user sessions and other repository operations are handled by User Manager. These latter two components use Data Manager for database operations and Display Manager component to send their results to be displayed.

Finally, there is the database server node. There is a JDBC connection between the database server and the application server. The database server consists of the database portion of Recipe Repository system. There is one database component as a respective of the Data Manager component in the application server. There is the Repository Database, which contains all records. This component again declares interfaces, and the appropriate component in the application server communicates with these components through the interfaces they declare.

3. Use Case Design

This section provides a usage scenario for the software. It organized information collected during requirements elicitation into use-cases.

3.1 User profiles

The system must have two different user profiles that distinguish the objectives and rights of users. We can classify these users as:

- Registered User
- Unregistered User

A registered user can perform all operations that an unregistered user can perform. The difference between two user types can be summarized as follows: An unregistered user can perform only search and display type of actions; on the other hand a registered user can perform editing type of actions in addition to these.

In the use case diagrams, actors are specified as in the following diagram:

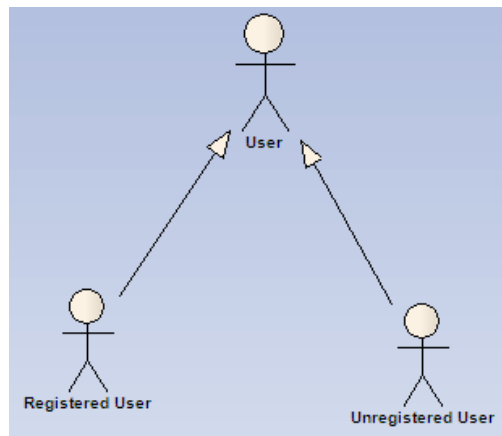


Figure 2 Actors

In use case diagrams, registered user specific actions are displayed by using "Registered User" actor; unregistered user specific actions are displayed by using "Unregistered User" actor and common actions are displayed by using "User" actor.

3.2 Use-cases

3.2.1 Unregistered User

3.2.1.1 Register to the system

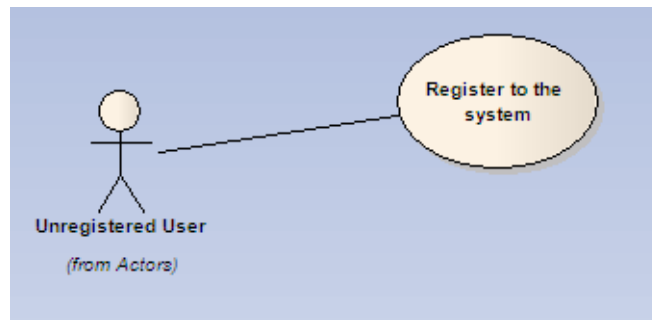


Figure 3 Register to the system use case

Use context: This use case is activated on demand if an unregistered user requests to register to Recipe Repository System.

Actors: Unregistered Recipe Repository System user. This actor requests to register to Recipe Repository System.

Aim: The aim of this use case is to provide the unregistered users to register to the system.

Preconditions: None.

Description:

Step1) Recipe repository system shall allow unregistered user to request registration to the system.

Step 2) Recipe repository system shall check the information entered by unregistered user for completeness and correctness.

Exceptional case 2.1) If the entered user information is incomplete or incorrect, the Recipe Repository System shall prompt for reentering the user information. Then the flow continues from Step 2.

Step 3) The user becomes a registered user with the information he/she provided.

Step 4) Use case ends.

3.2.2 Registered User

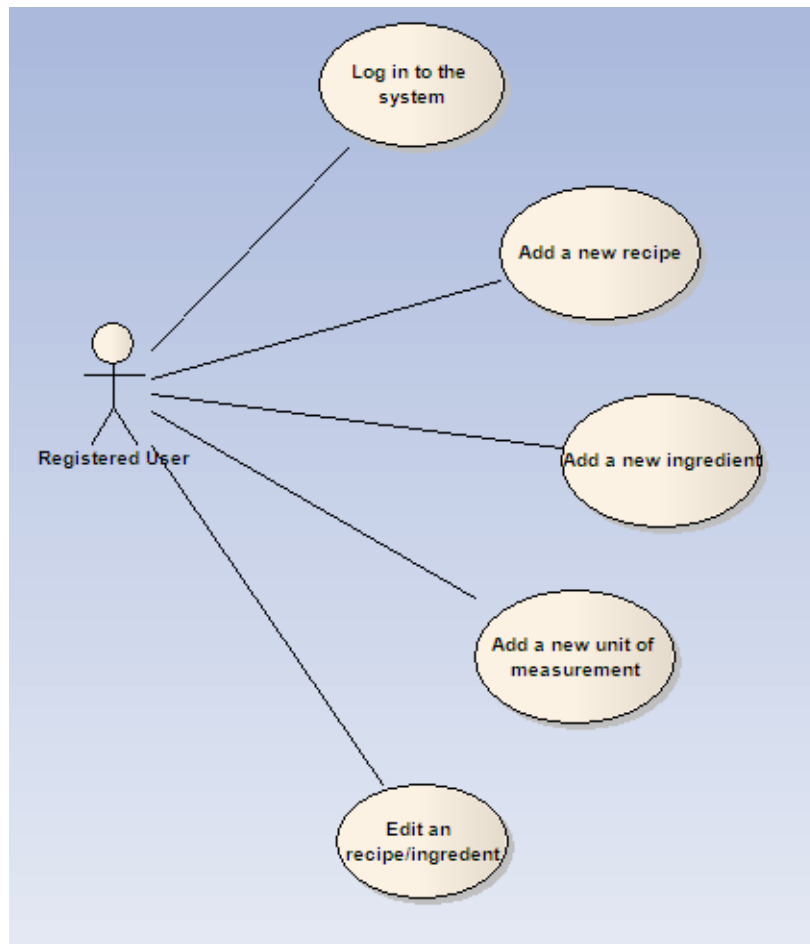


Figure 4 Registered user use cases

3.2.2.1 Log-in to the system

Use context: This use case is activated on demand if a user requests to log into the Recipe Repository System by entering his/her user information.

Actors: Recipe Repository System registered user. This actor requests to log into the system.

Aim: The aim of this use case is to provide the registered users to log into the system.

Preconditions: The user of this scenario is already registered to the system before.

Description:

Step1) Recipe repository system shall allow registered user to request system log in by entering his/her user information.

Step 2) Recipe repository system shall check the entered information to match the user that requesting login to a user in its database.

Exceptional case 2.1) If the entered user information does not match with any user information in the database, Recipe Repository System shall prompt for reentering his/her information. Then the flow continues from Step 2.

Step 3) The user accepted as registered user and has registered user access rights.

Step 4) The use case ends.

3.2.2.2 Add a new recipe

Use context: This use case is activated on demand if a user requests to add a new recipe.

Actors: Recipe Repository System registered user. This actor requests to add a new recipe addition.

Aim: The aim of this use case is to provide the registered users to add a new recipe to the system.

Preconditions: Registered user is logged in to the system.

Description:

Step1) Recipe repository system shall allow registered user to request a new recipe addition.

Step 2) Recipe repository system shall add the requested recipe to the recipe list if it is not a duplication of an existing recipe.

Exceptional case 2.1) If requested new recipe name is a duplication of an existing recipe, Recipe repository system shall prompt for reentering the recipe name. Then the flow continues from Step 2.

Step3) Recipe repository system shall allow registered user to add ingredients to the recipe with amount information. **Step 4)** Recipe is added to recipe database.

Step 5) Use case ends.

3.2.2.3 Add a new ingredient

Use context: This use case is activated on demand if a user requests to add a new ingredient.

Actors: Recipe Repository System registered user. This actor requests to add a new ingredient.

Aim: The aim of this use case is to provide the registered users to add a new ingredient to the system.

Preconditions: Registered user is logged in to the system.

Description:

Step1) Recipe repository system shall allow registered user to request a new ingredient addition with its unit of measurement.

Exceptional case 1.1) If requested ingredient unit of measurement does not exist in the ingredient units of measurement database, Recipe Repository System shall prevent ingredient addition (User shall perform ingredient unit of measurement addition first).

Exceptional case 1.2) Use case ends.

Step 2) Recipe repository system shall add the requested ingredient with its unit of measurement to the ingredient database if it is not a duplication of an existing ingredient.

Exceptional case 2.1) If requested new ingredient name is a duplication of an existing ingredient, Recipe Repository System shall prompt for reentering the ingredient name. Then the flow continues from Step 2.

Step 3) Ingredient is added to ingredient database.

Step 4) Use case ends.

3.2.2.4 Add a new unit of measurement

Use context: This use case is activated on demand if a user requests to add a new unit of measurement.

Actors: Recipe Repository System registered user. This actor requests to add a new unit of measurement.

Aim: The aim of this use case is to provide the registered users to add a new unit of measurement to the system.

Preconditions: Registered user is logged in to the system.

Description:

Step1) Recipe repository system shall allow registered user to request a new unit of measurement addition.

Step 2) Recipe repository system shall add the requested unit of measurement to the units of measurement database, if its name is not a duplication of an existing unit of measurement.

Exceptional case 2.1) If requested new unit of measurement name is a duplication of an existing unit of measurement, Recipe Repository System shall prompt for reentering the unit of measurement name. Then the flow continues from Step 2.

Step 3) Unit of measurement is added to ingredient database.

Step 4) Use case ends.

3.2.2.5 Edit an ingredient/recipe

Use context: This use case is activated on demand if a user requests to edit an

ingredient/recipe.

Actors: Recipe Repository System registered user. This actor requests to edit an ingredient/recipe.

Aim: The aim of this use case is to provide the registered users to edit an ingredient/recipe.

Preconditions: Registered user is logged in to the system.

Description:

Step1) Recipe repository system shall allow registered user to request to edit an ingredient/recipe.

Step 2) Recipe repository system shall update the existing ingredient/recipe, if its name is not a duplication of an existing ingredient/recipe name.

Exceptional case 2.1) If requested new ingredient/recipe name is a duplication of an existing ingredient/recipe name, Recipe Repository System shall prompt for reentering the ingredient/recipe name. Then the flow continues from Step 2.

Step 3) Ingredient/recipe is updated in the database.

Step 4) Use case ends.

3.2.3 All Users

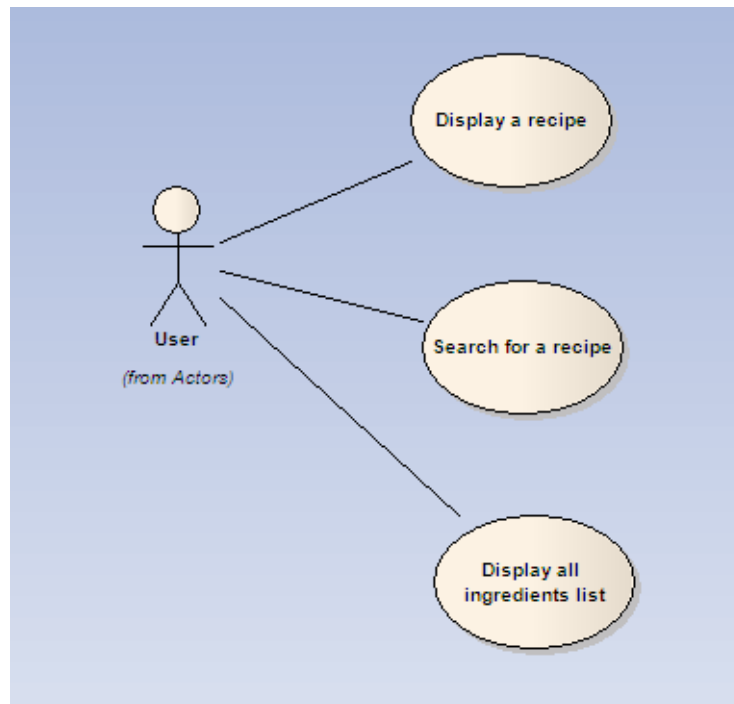


Figure 5 All users use cases

3.2.3.1 Display a recipe

Use context: This use case is activated on demand if a user requests to display a recipe with its content.

Actors: Recipe Repository System user (registered or unregistered). This actor requests to display a recipe with its content.

Aim: The aim of this use case is to provide the user (registered or unregistered) to display a recipe with its content.

Preconditions: None.

Description:

Step1) Recipe repository system shall allow registered user to request to display a recipe with its content by double clicking the recipe name.

Step 2) Recipe repository system shall display the recipe with its ingredients and description to the user.

Step 3) Use case ends.

3.2.3.2 Display all ingredients list

Use context: This use case is activated on demand if a user requests to display all ingredients independent of the recipes.

Actors: Recipe Repository System user (registered or unregistered). This actor requests to display all ingredients independent of the recipes.

Aim: The aim of this use case is to provide the user (registered or unregistered) to display all ingredients independent of the recipes.

Preconditions: None.

Description:

Step1) Recipe repository system shall allow registered user to request to display all ingredients independent of the recipes.

Step 2) Recipe repository system shall display the ingredients list on the screen.

Step 3) Use case ends.

3.2.3.3 Search for a recipe

Use context: This use case is activated on demand if a user requests to search for a recipe with entering some of its ingredients as keyword.

Actors: Recipe Repository System user (registered or unregistered). This actor requests to search for a recipe with entering some of its ingredients as keyword.

Aim: The aim of this use case is to provide the user (registered or unregistered) to search for a recipe.

Preconditions: None.

Description:

Step1) Recipe repository system shall allow registered user to request search for a recipe with entering some of its ingredients as keyword.

Step 2) Recipe repository system shall display the search results as a list of the found recipes in the order of their relevancy.

Step 3) Use case ends.

4. Data Design

4.1 Entity Relationship Diagram

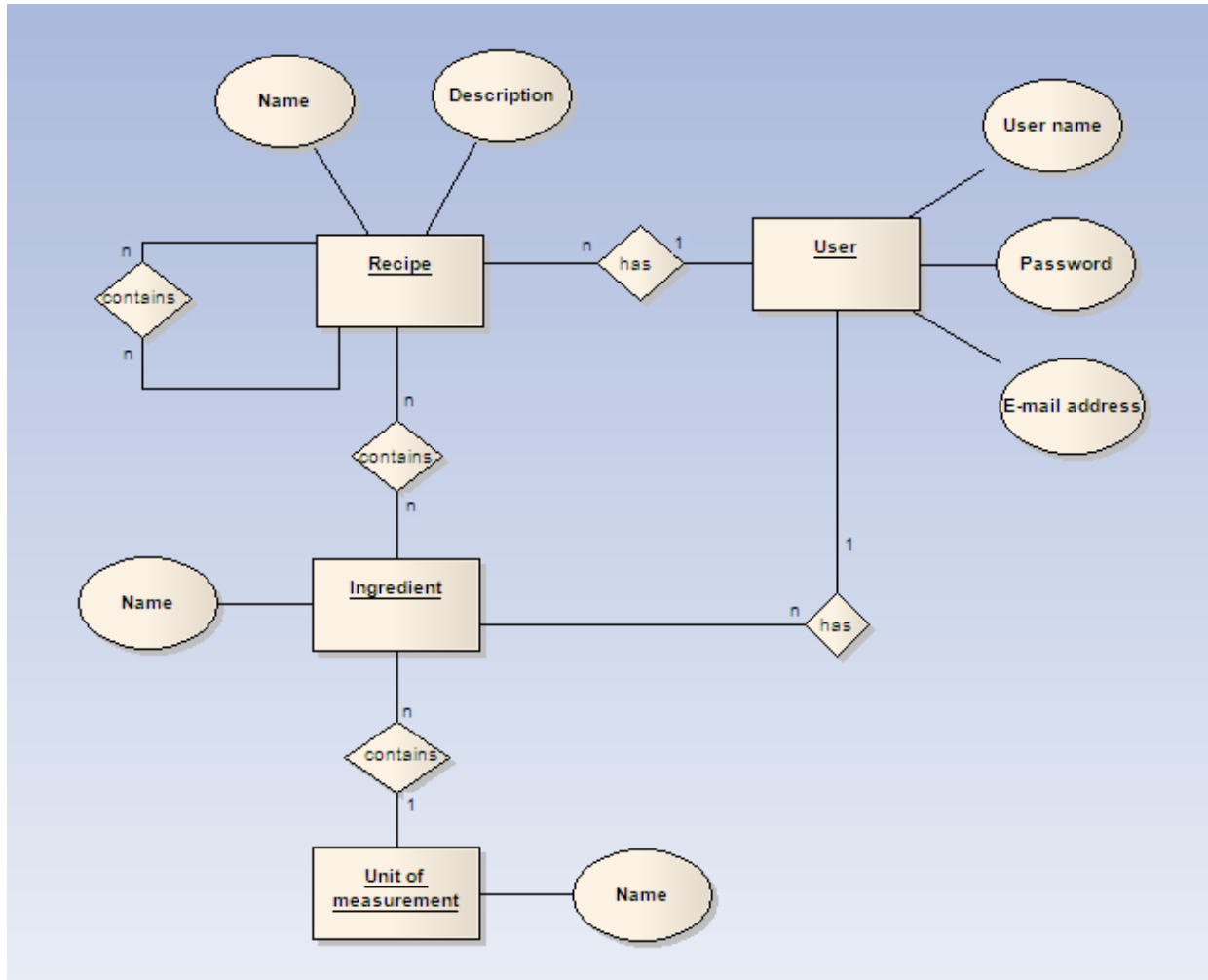


Figure 6 ER Diagram

4.1.1 Entities

a) User: 'User' is the entity that constitutes the internal system representation of a person using Recipe Repository System. It interacts with the system to accomplish some tasks, create a recipe, and a new ingredient, search for a recipe, edit a recipe/ingredient etc. It has a user name, which uniquely determines it, a password that secures its data, an e-mail address.

b) Recipe: 'Recipe' is an entity which is the main constituting part of the Recipe Repository System. A Recipe represents a real life recipe which has some number of ingredients and description and it is uniquely identified by its name. A recipe can have another recipe as its ingredient as well. Each recipe has an owner of type User, which only has right to edit its name.

c) Ingredient: An 'Ingredient' is the constituting part of the recipe. It has a unique name and a unit of measurement. Each ingredient has an owner of type User, which only has right to edit its name.

d) Unit of measurement: 'Unit of measurement' is used to define the quantity of an ingredient. It has a unique name. If an ingredient is defined as a type of recipe, it will not have a unit of measurement. Instead, 'number of count' will be used to define its quantity.

4.1.2 Relationships

a) has: The relationship 'has' is between recipe and user and ingredient and user. It means that, each recipe is owned by one user and one user can have more than one receipt. Similarly, each ingredient is owned by one user and one user can have more than one ingredient.

b) contains: The relationship 'contains' is between recipe and ingredient and ingredient and unit of measurement. It means that, each recipe can made of some number of ingredients and one ingredient can be used by more than one recipe. On the other hand, each ingredient has one unit of measurement.

4.2 Data Dictionary

User	
UserId (Primary key)	Unique id to identify the user
User Name	User entered name for each user
E-mail	User entered email information
Password	User entered password information

Recipe	
RecipeId (Primary key)	Unique id to identify each recipe
Name	Name for the recipe
Description	Description for the recipe
Contained ingredients table id (Foreign key: Contained Ingredients table - Cid)	Id for the table that holds the ingredients for the recipe with its type (if it is also a recipe or not) and amount

Ingredient	
Ingredient Id (Primary key)	Unique id to identify each ingredient
Name	Name for the ingredient
Unit id (Foreign key: Unit Of Measurements table - Unit id)	Id for the table that holds the unit name

Unit Of Measurement	
Unit Id (Primary key)	Unique id to identify each unit of measurement
Name	Name for the unit

Contained Ingredients	
Cid (Primary key)	Unique id to identify the contained ingredients table for a recipe
Ingredient id (Foreign key: Ingredients table - Ingredient id)	Id for the table that holds the ingredient specific information
Ingredient type	An integer value to specify whether the ingredient is a recipe or not (If 0, it is not a recipe, only an ingredient. If 1, it is a recipe as well)
Ingredient amount	The amount for the ingredient. This value is associated with the unit of measurement of the ingredient (i.e. if unit is gr and the amount value is 100, this means 100 gr of that ingredient). If ingredient is a recipe, this amount value is thought as "times of recipe" (i.e. if ingredient is of type recipe1 and the amount value is 2, this means 2 times the recipe1).

User Ingredients	
User Id (Foreign key: User table-User id)	Id that specified the user for which the table holds the owned ingredients.
Ingredient id (Foreign key: Ingredient table - Ingredient id)	The id for the ingredient that is entered to the system and by the user which is specified by above user id (This ingredient is editable only by this user).

User Recipes	
User Id (Foreign key: User table-User id)	Id that specified the user for which the table holds the owned recipes.
Recipe id (Foreign key: Recipe table - Recipe id)	The id for the recipe that is entered to the system and by the user which is specified by above user id (This recipe is editable only by this user).

User Unit Of Measurements	
User Id (Foreign key: User table-User id)	Id that specified the user for which the table holds the owned unit of measurements.
Ingredient id (Foreign key: Unit Of Measurements table - Unit id)	The id for the unit of measurement that is entered to the system and by the user which is specified by above user id (This unit of measurement is editable only by this user).

5. Modeling

5.1 Class Diagram

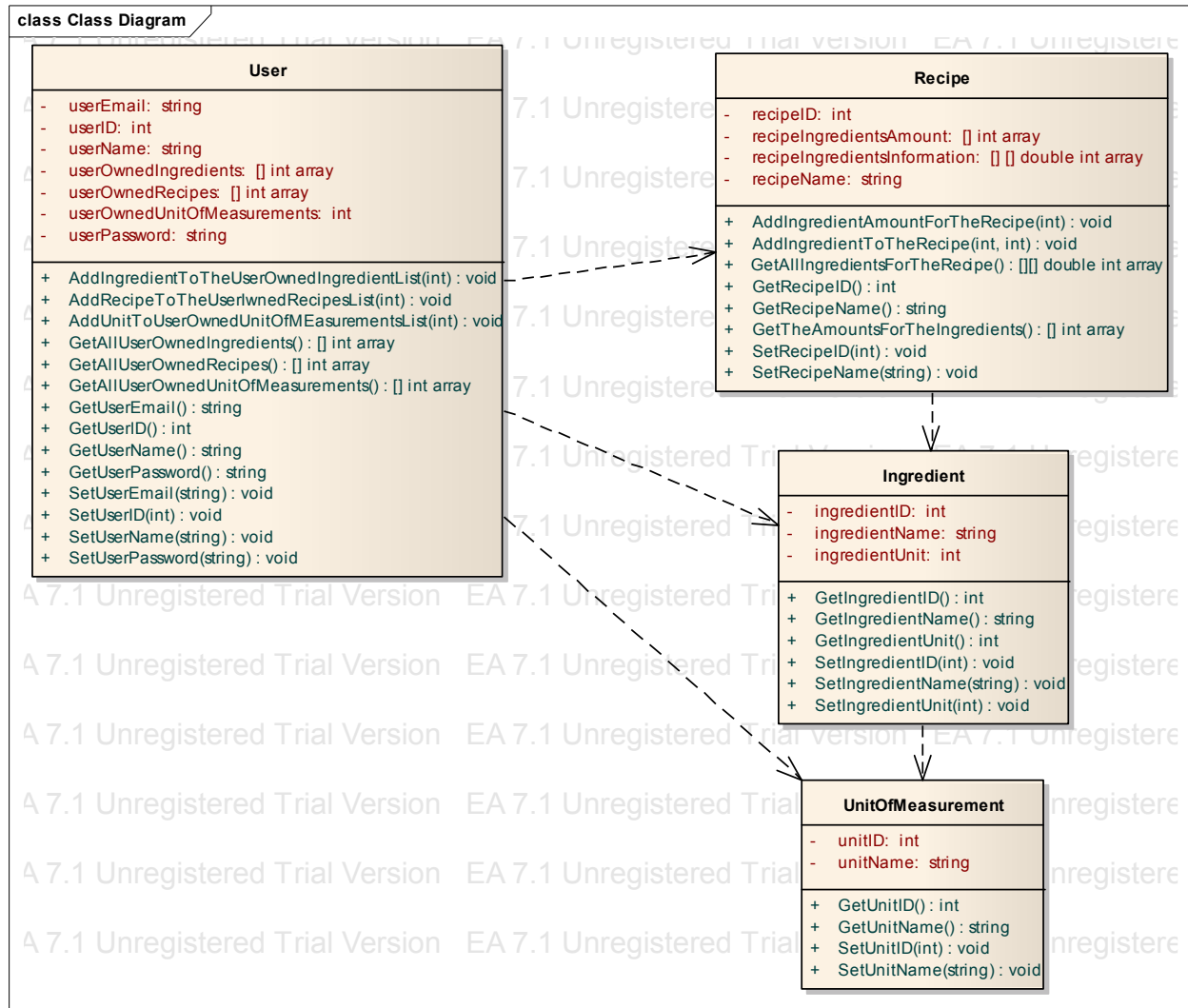


Figure 7 Class Diagram

Ingredient

This class holds the necessary information to define an ingredient (such as the unique id for an ingredient, name for the ingredient and the unit of measurement for the ingredient).

Attributes

Attribute	Notes
ingredientID int Private	Unique id to identify each ingredient

Attribute	Notes
ingredientName string Private	Name for the ingredient
ingredientUnit int Private	Id for the corresponding unit name

Operations

Method	Notes	Parameters
GetIngredientID() int Public	Getter function for the ingredient id	
GetIngredientName() string Public	Getter function for the ingredient name	
GetIngredientUnit() int Public	Getter function for the ingredient unit	
SetIngredientID() void Public	Setter function for the ingredient ID	<u>int</u> [in] <u>ingredientID</u>
SetIngredientName() void Public	Setter function for the ingredient name	<u>string</u> [in] <u>ingredientName</u>
SetIngredientUnit() void Public	Setter function for the ingredient unit	<u>int</u> [in] <u>ingredientUnit</u>

Recipe

This class holds the necessary information to define a recipe (such as the unique id for a recipe, name for the recipe, ingredient list for the recipe).

Attributes

Attribute	Notes
recipeID int Private	Unique id to identify each recipe
recipeIngredientsAmount [] int array Private	Array to hold ingredient amount that are hold in recipeIngredientsInformation array
recipeIngredientsInformation [][] double int array Private	Array that holds the ingredients for the recipe with its type. First dimension for ingredient id and second dimension for ingredient type (if it is also a recipe or not).
recipeName string Private	Name for the recipe

Operations

Method	Notes	Parameters
AddIngredientAmountForTheRecipe() void Public	Function to add amount information for the ingredient in the recipeIngredientsInformation array at the same array position.	<u>int</u> [in] <u>amount</u>
AddIngredientToTheRecipe() void Public	Function to add an ingredient to the recipe with an indicator to show it is also a recipe or not	<u>int</u> [in] <u>ingredientType</u> <u>int</u> [in] <u>ingredientID</u>
GetAllIngredientsForTheRecipe() [][] double int array Public	Function to obtain the ingredients list for the recipe	
GetRecipeID() int Public	Getter function for recipe ID	

Method	Notes	Parameters
GetRecipeName() string Public	Getter function for recipe name	
GetTheAmountsForTheIngredients() [] int array Public	Function to obtain a list of the ingredien amounts	
SetRecipeID() void Public	Setter function for recipe ID	<u>int</u> [in] <u>recipeID</u>
SetRecipeName() void Public	Setter function for recipe name	<u>string</u> [in] <u>recipeName</u>

UnitOfMeasurement

This class holds the necessary information to define a unit of measurement (such as the unique id for a unit of measurement and name for the unit of measurement).

Attributes

Attribute	Notes
unitID int Private	Unique id to identify each unit of measurement
unitName string Private	Name for the unit of measurement

Operations

Method	Notes	Parameters
GetUnitID() int Public	Getter function for unit of measurement ID	
GetUnitName() string Public	Getter function for unit of measurement name	
SetUnitID() void	Setter function for unit of measurement ID	<u>int</u> [in] <u>unitID</u>

Method	Notes	Parameters
Public		
SetUnitName() void Public	Setter function for unit of measurement name	<u>string</u> [in] <u>unitName</u>

User

This class holds the user specific information in the system.

Attributes

Attribute	Notes
userEmail string Private	User entered email information
userID int Private	Used to uniquely specify the user in the system.
userName string Private	User entered name for each user
userOwnedIngredients [] int array Private	Array to hold the list of ingredient ids that are added to the system by the user
userOwnedRecipes [] int array Private	Array to hold the list of recipe ids that are added to the system by the user

Attribute	Notes
userOwnedUnitOfMeasurements int Private	Array to hold the list of unit of measurement ids that are added to the system by the user
userPassword string Private	User entered password information

Operations

Method	Notes	Parameters
AddIngredientToTheUserOwnedIngredientList() void Public	Function to add a new ingredient to the list of user owned ingredient list to the system	int [in] <u>ingredientID</u>
AddRecipeToTheUserOwnedRecipesList() void Public	Function to add a new recipe to the list of user owned recipes list	int [in] <u>recipeID</u>
AddUnitToUserOwnedUnitOfMeasurementsList() void Public	Function to add a new unit of measurement to the user defined unit of measurements in the system	int [in] <u>unitID</u>
GetAllUserOwnedIngredients() [] int array Public	Function to obtain all of the user defined ingredients in the system.	
GetAllUserOwnedRecipes() [] int array Public	Function to obtain all of the user owned recipes list	
GetAllUserOwnedUnitOfMeasurements() [] int array Public	Function to obtain all of the user defined unit of measurements in the system.	
GetUserEmail() string Public	Getter function for userEmail.	
GetUserID() int Public	Getter function for userID	
GetUserName() string Public	Getter function for userName	

Method	Notes	Parameters
GetUserPassword() string Public	Getter function for userPassword	
SetUserEmail() void Public	Setter function for userEmail	<u>string</u> [in] <u>userEmail</u>
SetUserID() void Public	Setter function for userID	<u>int</u> [in] <u>userID</u>
SetUserName() void Public	Setter function for userName	<u>string</u> [in] <u>userName</u>
SetUserPassword() void Public	Setter function for userPassword	<u>string</u> [in] <u>userPassword</u>

5.2 Sequence Diagram

5.2.1 Add New Recipe

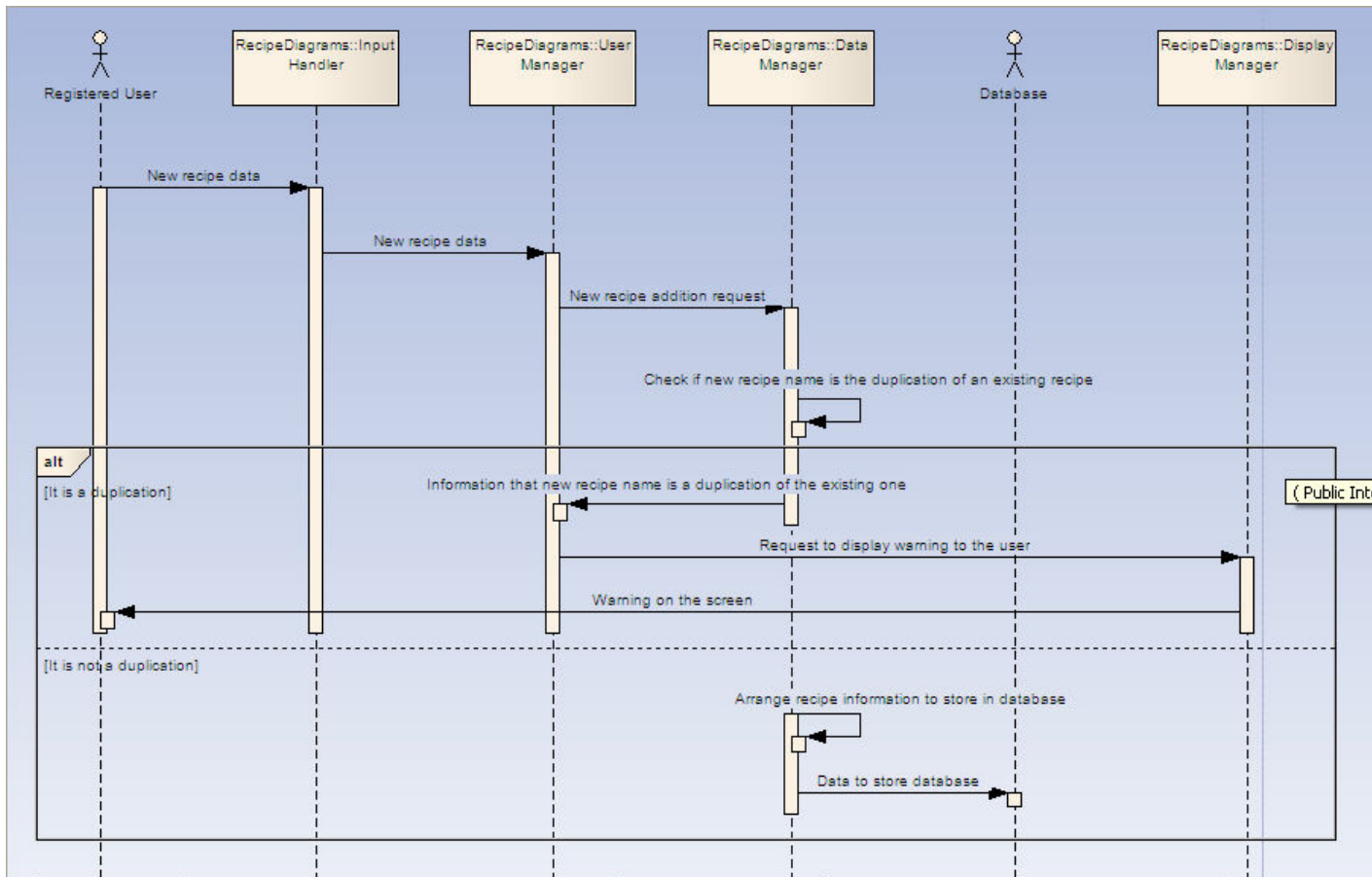


Figure 8 Add new recipe sequence diagram

5.2.2 Display All Recipes

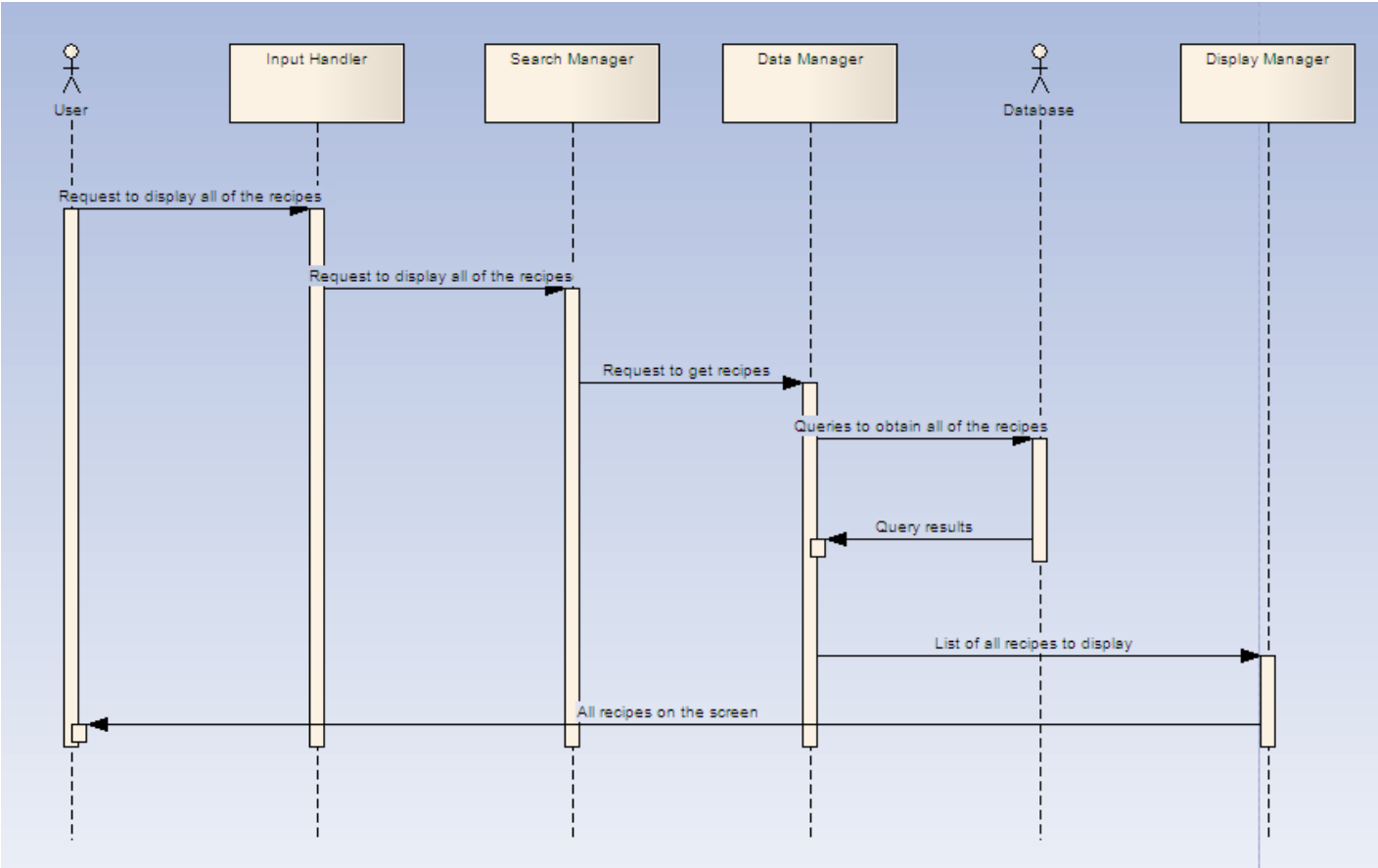


Figure 9 Display all recipes sequence diagram

5.2.3 Search for a Recipe

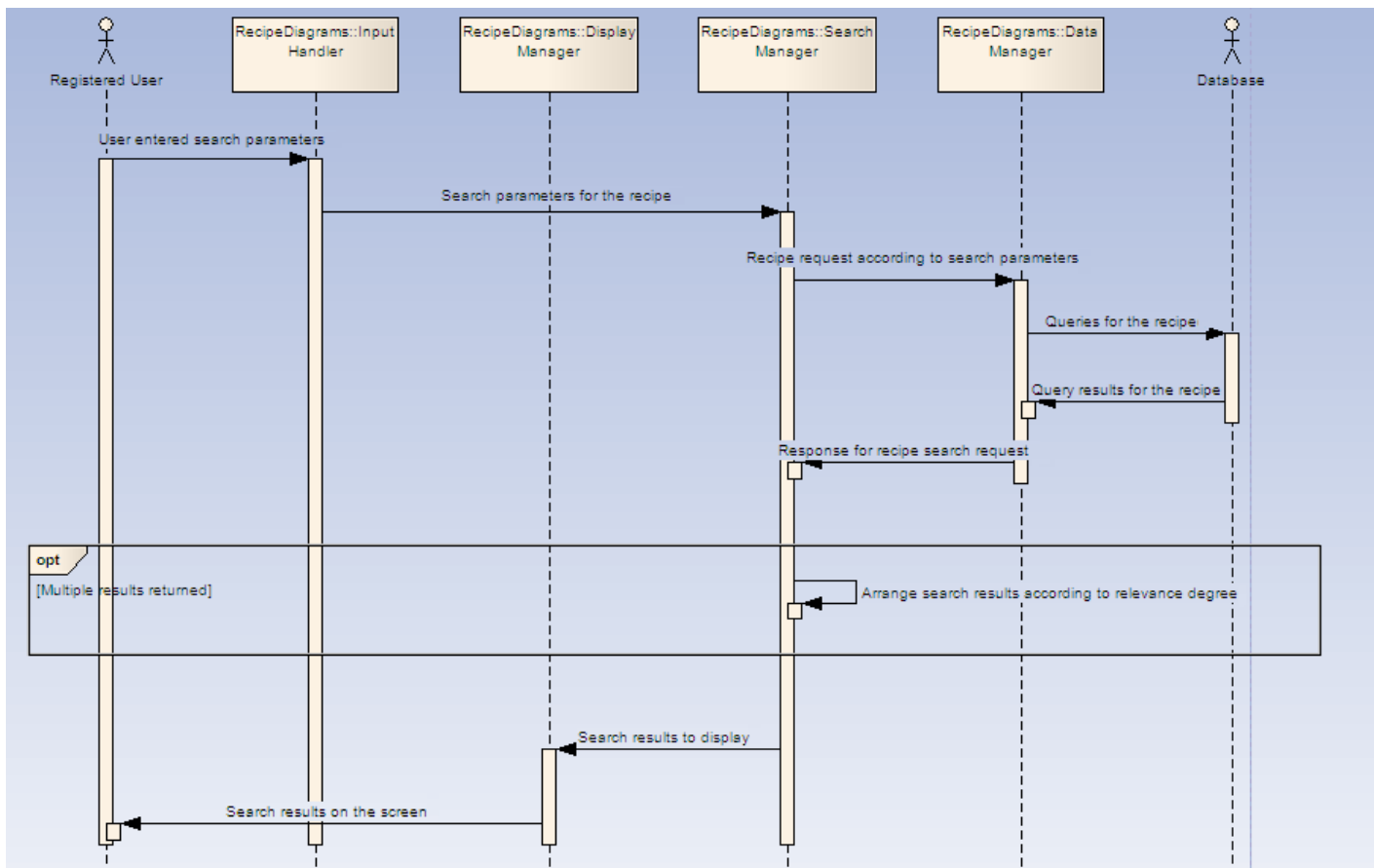


Figure 10 Search for a recipe sequence diagram

5.2.4 Edit an Ingredient

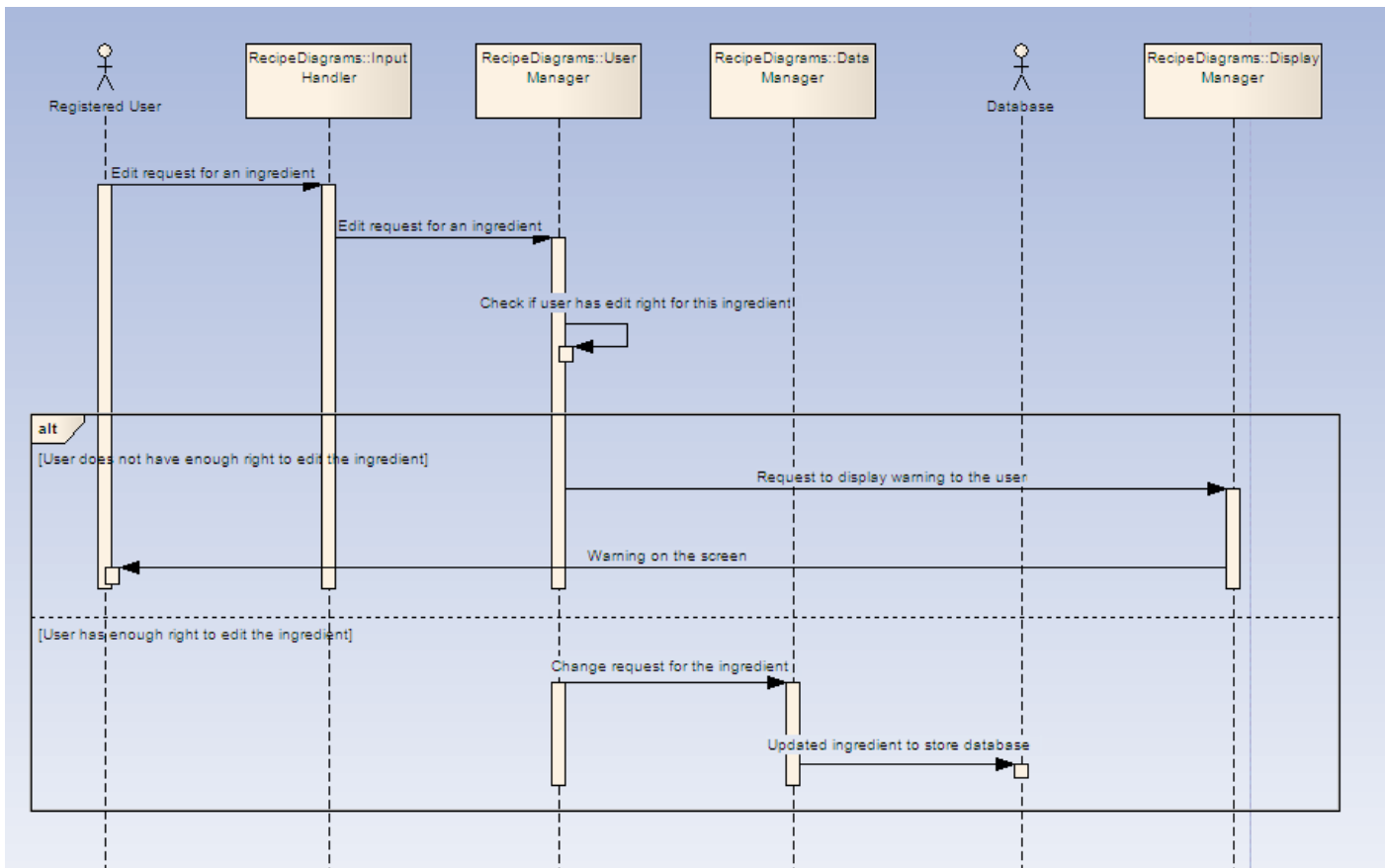


Figure 11 Edit an ingredient sequence diagram

6. User Interface Design

6.1 Home Menu

When you first enter to the Recipe Repository system, you will see the home page. However, you can navigate to this page from other pages by clicking the “Home” menu seen below.

Home

This is the home page for newcomer (or registered user not logged in yet).

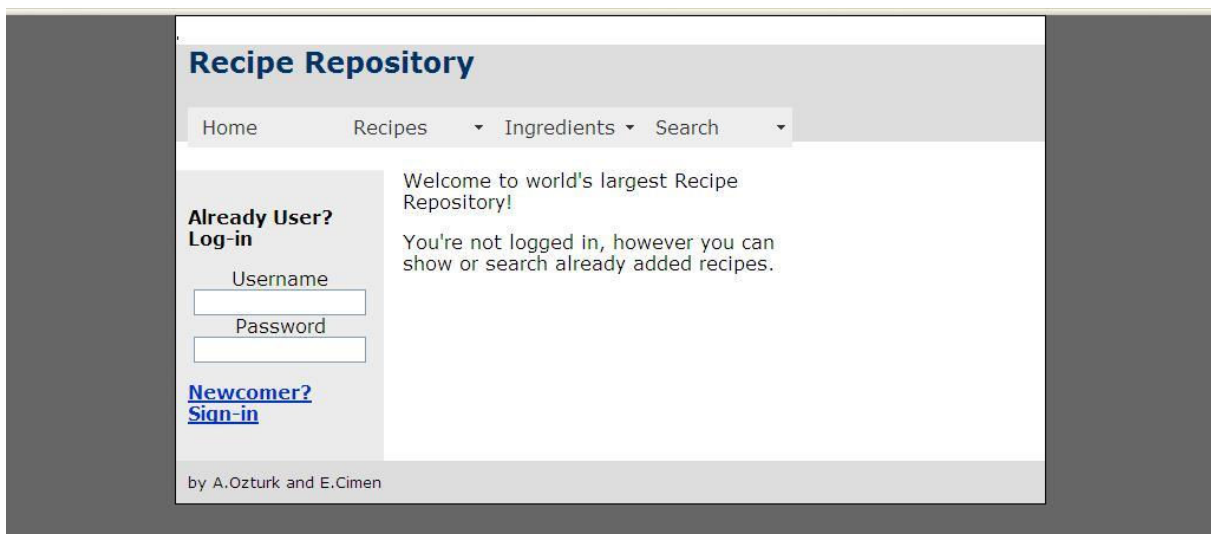
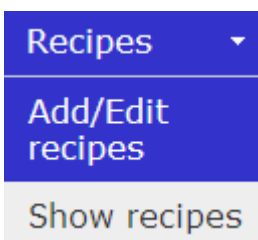


Figure 12 Home page

Unregistered user can register to the system by clicking “Newcomer? Sign-in” link shown in left pane and entering user information in the opened page. No screenshot for user registration shown here.

6.2 Recipes Menu

Only the user logged-in to the system can add or edit recipes of the repository, therefore a user can add or edit recipes after logging into the system by clicking the “Add/Edit recipes” submenu of the “Recipes” menu seen below.



A user can add a new recipe by entering its name and description, and submitting this information by the “Add” button.

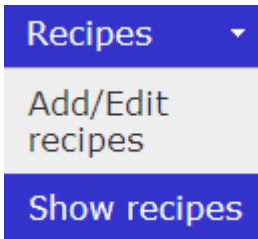
Already available recipe can be edited after selecting it from dropdown menu at the top. The name and description of the selected recipe can be changed and submitted by the "Add" button, or it can be deleted by the "Delete" button.

Moreover, the ingredients of the selected recipe can be managed on this page. Its already added ingredients can be removed after selecting them from 'Ingredients of the Recipe' list and submitting the "Remove Ingredient" button. Or, new ingredients can be added to the selected recipe by selecting an ingredient from the ingredients list, entering its amount according to its unit of measurement and submitting it by "Add Ingredient" button. A recipe can be added as an ingredient to the selected recipe by selecting a recipe from the recipes list, entering its amount as a count of measurement and submitting it by "Add Recipe as Ingredient" button.

The screenshot shows a web application titled "Recipe Repository". At the top, there is a navigation bar with links: "Home", "Recipes", "Ingredients", and "Search". Below this, there is a "User Name" field. The main content area has a heading "Select a recipe from the repository here." followed by a dropdown menu showing "Sample Recipe 1". Below the dropdown is a "Delete the Recipe:" label and a "Delete" button. A paragraph follows: "Add a new recipe or edit the selected recipe by submitting its name and description." Below this are input fields for "Recipe Name:" and "Recipe Description:". An "Add" button is positioned below the description field. Underneath is a section titled "Ingredients of the Recipe:" with a list box containing "Ingredient 1 of Recipe", "Ingredient 2 of Recipe", "Ingredient 3 of Recipe", and "Ingredient 4 of Recipe". To the right of the list box is a "Remove Ingredient" button. A paragraph follows: "Add ingredients to the selected recipe below." Below this are two input sections. The first is for "Ingredients:" with a list box showing "Ingredient A", "Ingredient B", and "Ingredient C", an "Add Ingredient" button, an "Amount:" input field, and a "Unit:" dropdown menu. The second is for "Recipes:" with a list box showing "Recipe A", "Recipe B", and "Recipe C", an "Add Recipe as Ingredient" button, and an "Amount:" input field. At the bottom left, there is a footer text: "by A.Ozturk and E.Cimen".

Figure 13 Add/Edit recipes page

A user can display already available recipes of the repository (no need for logging into the system) by clicking the "Show recipes" submenu of the "Recipes" menu seen below.



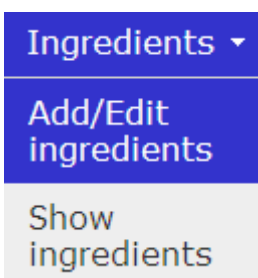
A recipe can be selected from the dropdown menu at the top. Its name, description and ingredients are shown as seen below. The amount and unit of each ingredient can be shown by selecting it from the recipe ingredients list.

The screenshot shows a web application titled 'Recipe Repository'. It has a navigation bar with 'Home', 'Recipes' (selected), 'Ingredients', and 'Search'. On the left is a 'User Name' field. The main content area has a 'Select a recipe:' dropdown menu with 'Sample Recipe 1' selected. Below this are fields for 'Recipe Name:', 'Recipe Description:', and 'Recipe Ingredients:'. The 'Recipe Ingredients:' field is a list box containing 'Ingredient 1', 'Ingredient 2', 'Ingredient 3', and 'Ingredient 4'. At the bottom of the form are 'Amount:' and 'Unit:' input fields. The footer text reads 'by A.Ozturk and E.Cimen'.

Figure 14 Show recipes page

6.3 Ingredients Menu

Only the user logged-in to the system can add or edit ingredients of the repository, therefore a user can add or edit ingredients after logging into the system by clicking the "Add/Edit ingredients" submenu of the "Recipes" menu seen below.



A user can add a new ingredient by entering its name and description, and submitting this information by the "Add" button.

Already available ingredient can be edited after selecting it from dropdown menu at the top. The name and description of the selected ingredient can be changed and submitted by the "Add" button, or it can be deleted by the "Delete" button. The unit of the ingredient can be set after selecting one from the units list and submitting it by the "Set Unit" button.

A new unit of measurement can be added to the repository system by entering its name and submitting it by the "Add Unit" button.

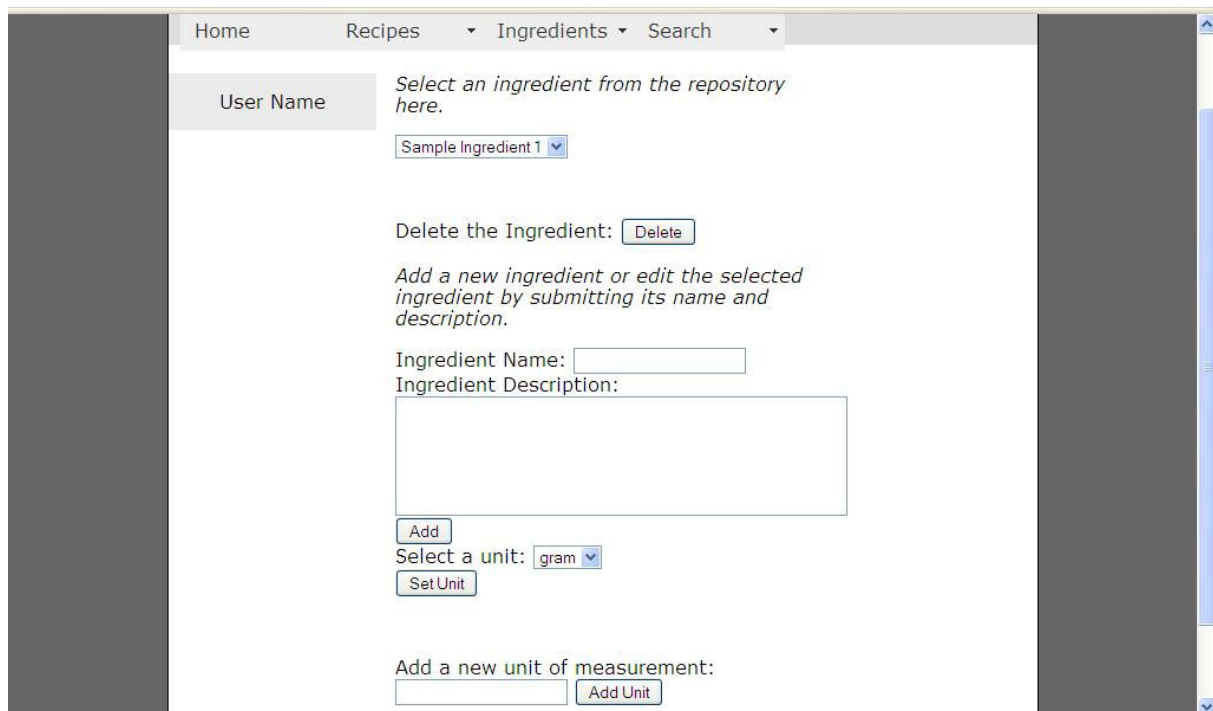


Figure 15 Add/Edit ingredients page

A user can display already available ingredients of the repository (no need for logging into the system) by clicking the "Show ingredients" submenu of the "Ingredients" menu seen below.

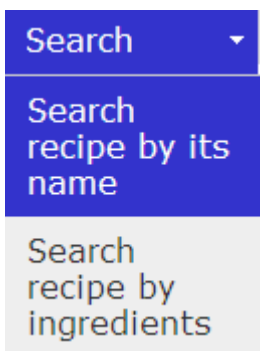


An ingredient can be selected from the dropdown menu at the top. Its name, description and unit are shown as seen below.

Figure 16 Show ingredients page

6.4 Search Menu

A user can search for already available recipes of the repository by entering its name (no need for logging into the system) on the page opened after clicking the "Search recipe by its name" submenu of the "Search" menu seen below.



The resulted recipes are ordered according to their decreasing relevancy in the list seen below. The detailed information of the recipe selected from this list can be shown by submitting the "Show" button. Then, a page shown in Figure 14 Show recipes page is opened.

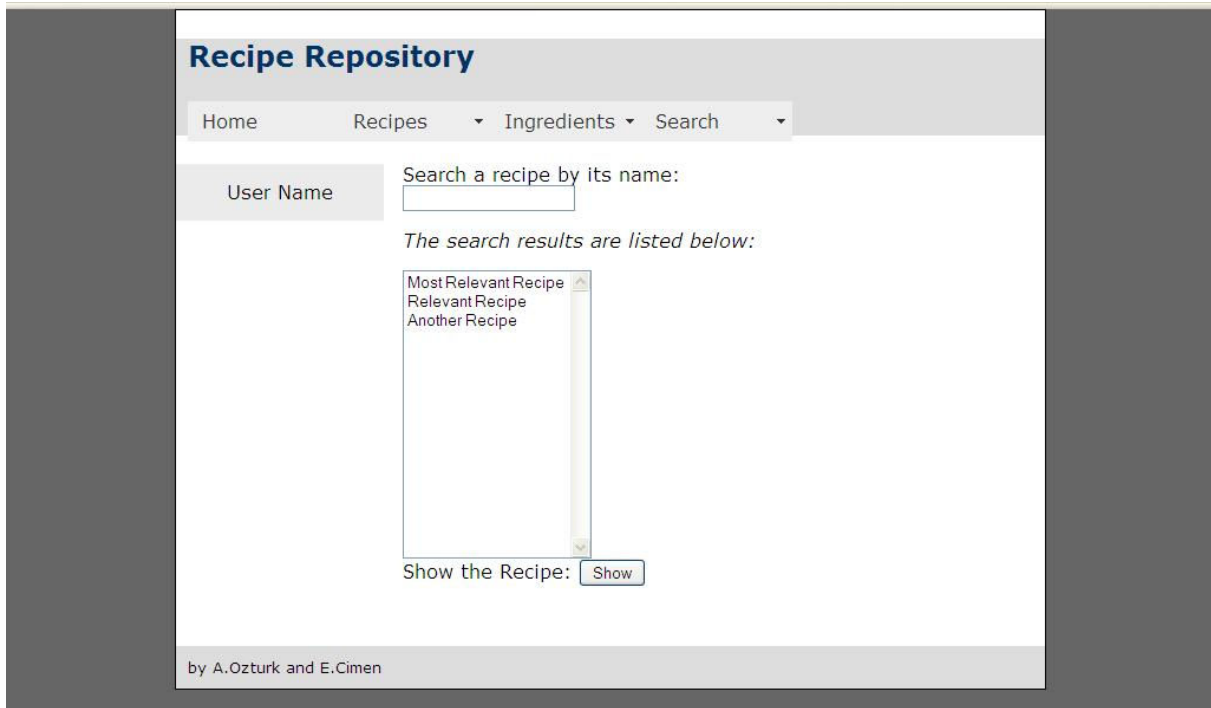
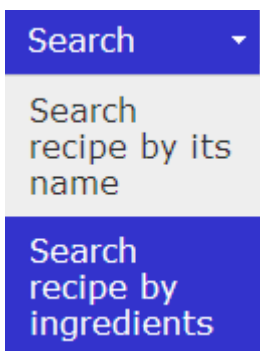


Figure 17 Search recipes by its name page

A user can search for already available recipes of the repository by entering one or more ingredient names separated by spaces (no need for logging into the system) on the page opened after clicking the "Search recipe by its name" submenu of the "Search" menu seen below.



The resulted recipes are ordered according to their decreasing relevancy in the list seen below. The detailed information of the recipe selected from this list can be shown by submitting the "Show" button. Then, a page shown in Figure 14 Show recipes page is opened.

Recipe Repository

HomeRecipesIngredientsSearch

User Name

Search a recipe by its ingredients:

And

The search results are listed below:

Most Relevant Recipe

Relevant Recipe

Another Recipe

Show the Recipe: Show

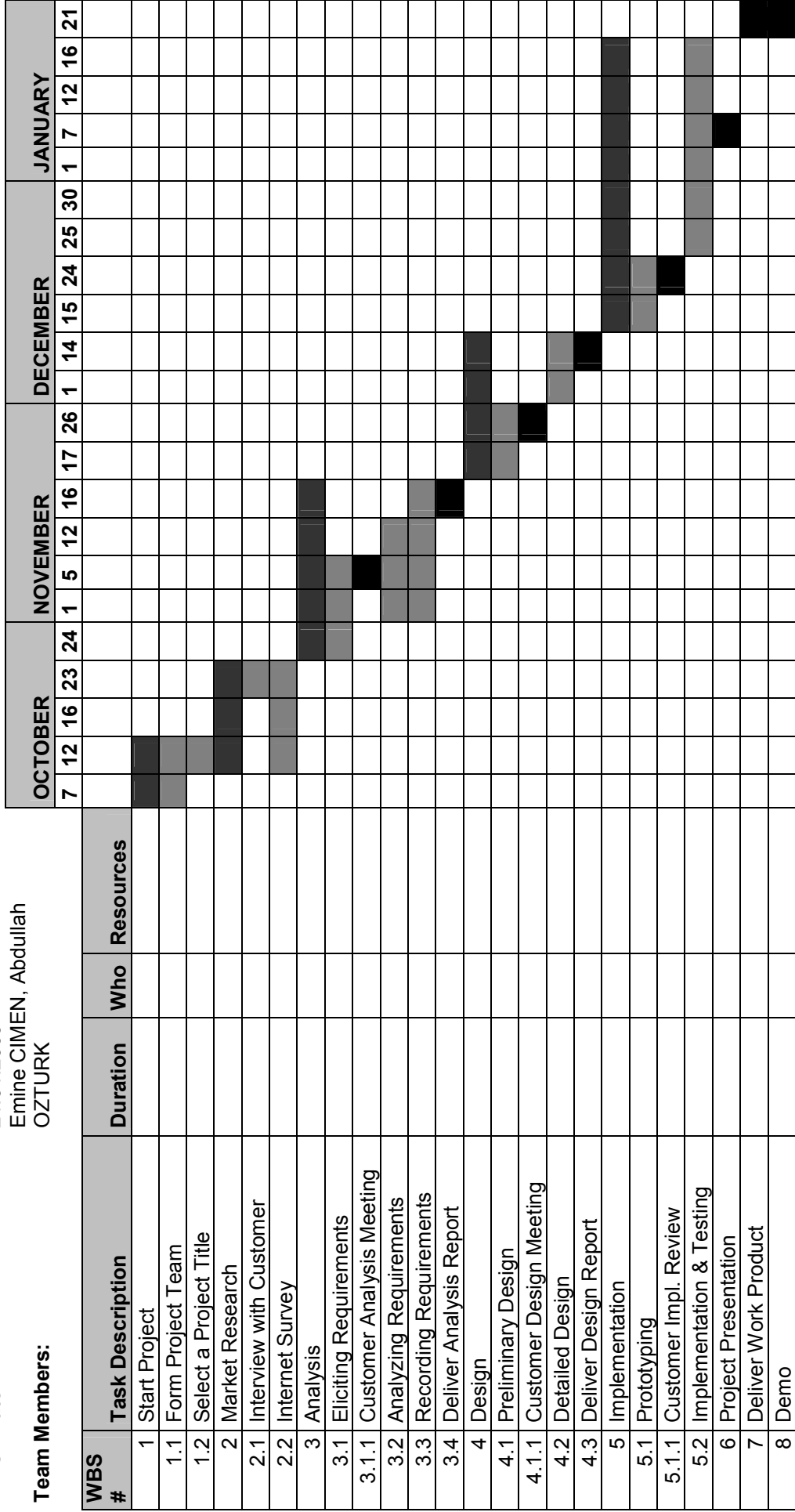
by A.Ozturk and E.Cimen

Figure 18 Search recipes by ingredients page

7. Project Plan

7.1 Gantt Chart

Project Name: Recipe Repository System
Start Date: 12.10.2008
Finish Date: 21.01.2009
Team Members: Emine CIMEN, Abdullah OZTURK



Appendixes

A.1 Search Algorithm

While performing searches for recipes in our tree-like data structure, we will prefer to use **Breadth-First Search (BFS) algorithm**. In graph theory, breadth-first search (BFS) is a graph search algorithm that begins at the root node and explores all the neighboring nodes. Then for each of those nearest nodes, it explores their unexplored neighbor nodes, and so on, until it finds the goal.

We have preferred this algorithm since we want to search upper layer nodes (i.e. ingredients or inner recipes) before looking deeper nodes of the inner recipes.

The figure below shows the order in which the nodes are expanded.

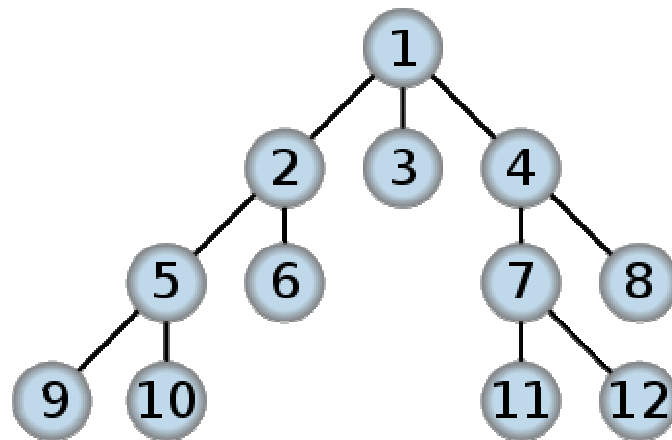


Figure 19 BFS tree search algorithm

BFS is an uninformed search method that aims to expand and examine all nodes of a graph or combinations of sequence by systematically searching through every solution. In other words, it exhaustively searches the entire graph or sequence without considering the goal until it finds it. It does not use a heuristic.

From the standpoint of the algorithm, all child nodes obtained by expanding a node are added to a FIFO queue. In typical implementations, nodes that have not yet been examined for their neighbors are placed in some container (such as a queue or linked list) called "open" and then once examined are placed in the container "closed".

Space Complexity

Since all of the nodes of a level must be saved until their child nodes in the next level have been generated, the space complexity is proportional to the number of nodes at the deepest level. Given a branching factor b and graph depth d the asymptotic space complexity is the number of nodes at the deepest level, $O(b^d)$. When the number of vertices and edges in the graph are known ahead of time, the space complexity can also be expressed as $O(|E| + |V|)$ where $|E|$ is the cardinality of the set of edges (the number of edges), and $|V|$ is the cardinality of the set of vertices. In the worst case the graph has a depth of 1 and all vertices must be stored. Since it is exponential in the depth of the graph, breadth-first search is often impractical for large problems on systems with bounded space.

Time Complexity

Since in the worst case breadth-first search has to consider all paths to all possible nodes the

time complexity of breadth-first search is $b + b^2 + b^3 + \dots + b^d$ which asymptotically approaches $O(b^d)$. The time complexity can also be expressed as $O(|E| + |V|)$ since every vertex and every edge will be explored in the worst case.

Completeness

Breadth-first search is complete. This means that if there is a solution breadth-first search will find it regardless of the kind of graph. However, if the graph is infinite and there is no solution breadth-first search will diverge.

Optimality

For unit-step cost, breadth-first search is optimal. In general breadth-first search is not optimal since it always returns the result with the fewest edges between the start node and the goal node. If the graph is a weighted graph, and therefore has costs associated with each step, a goal next to the start does not have to be the cheapest goal available. This problem is solved by improving breadth-first search to uniform-cost search which considers the path costs. Nevertheless, if the graph is not weighted, and therefore all step costs are equal, breadth-first search will find the nearest and the best solution.