**Web-based Ordering & Ingredient Estimating for Bakery Manufacturer**

Software Design Document

By

**Miss. NontraMahachanont 542115026**

**Mr. ParinyaPanyanak 542115034**

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Mr.PhudinanSingkhamfu**

|  |  |  |
| --- | --- | --- |
| **Software Requirement Specification** | | |
| Cross Ref. VSE-29110 | **Coverage Level:** | **Version:** |
|  | V.3.0 | V.3.0 |

|  |  |
| --- | --- |
| **Process Ownership** | **Approving Authority** |
| Web-based Ordering & Ingredient Estimating for Bakery Manufacturer | Aj.Phudinan Singkhamfu |
| **Scope** | **Approved Date** |
| Senior Project | 16 January 2015 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version Number** | **Record Date** | **Prepared/**  **Modified By** | **Reviewed By** | **Change Details** |
| **Documents History** | | | | |
| V.0.1 | 06-05-2014 | NM, PP | NM, PP | - Add Chapter 1  - Add Chapter 2 |
| V.0.2 | 07-05-2014 | NM, PP | NM, PP | - Add Chapter 3  - Fix inappropriate words |
| V.0.4 | 09-05-2014 | NM, PP | NM, PP | - Add Chapter 4  - Fix inappropriate words |
| V.0.6 | 12-05-2014 | NM, PP | NM, PP | - Adjust Chapter 2  - Fix inappropriate words |
| V.1.0 | 24-06-2014 | NM, PP | NM, PP | - Adjust Chapter 2  - Adjust Chapter 3  - Adjust Chapter 4  - Fix inappropriate words |
| V.1.1 | 16-09-2014 | NM, PP | NM, PP | - Add Chapter 1  - Add Chapter 2  - Add Chapter 4 |
| V.1.2 | 17-09-2014 | NM, PP | NM, PP | - Add Chapter 3  - Fix inappropriate words |
| V.2.0 | 22-10-2014 | NM, PP | NM, PP | - Adjust Chapter 2  - Adjust Chapter 3  - Adjust Chapter 4  - Fix inappropriate words |
| V.2.1 | 16-11-2014 | NM, PP | NM, PP | - Add Chapter 1  - Add Chapter 2  - Add Chapter 3  - Add Chapter 4 |
| V.3.0 | 16-01-2015 | NM, PP | NM, PP | - Adjust Chapter 1  - Adjust Chapter 2  - Adjust Chapter 3  - Adjust Chapter 4 |

**\*NM = NontraMahachanont**

**\*PP = ParinyaPanyanak**

**\*PS = PhudinanSingkhamfu**

**Table of contents**

[**Chapter One | System Architecture** 5](#_Toc409365348)

[**Chapter Two |Detailed Design** 8](#_Toc409365349)

[2.1 Class Diagram 8](#_Toc409365350)

[2.2 Class Diagram Description 9](#_Toc409365351)

[2.2.1 Controller Class 9](#_Toc409365352)

[2.2.2 Model Class 17](#_Toc409365353)

[2.2.3 Service Class 25](#_Toc409365354)

[2.3 Sequence Diagram 26](#_Toc409365355)

[2.3.1 Admin can activate member 26](#_Toc409365356)

[2.3.2 Member can login 27](#_Toc409365357)

[2.3.3 Member can logout 28](#_Toc409365358)

[2.3.4 Admin can edit member information 29](#_Toc409365359)

[2.3.5 Admin can delete member 30](#_Toc409365360)

[2.3.6 Admin can view member information 31](#_Toc409365361)

[2.3.7 Admin can create bakery product 32](#_Toc409365362)

[2.3.8 Admin can edit bakery product information 33](#_Toc409365363)

[2.3.9 Admin can delete bakery product 34](#_Toc409365364)

[2.3.10 Member can view bakery product list 35](#_Toc409365365)

[2.3.11 Wholesaler order bakery product 36](#_Toc409365366)

[2.3.12 Wholesaler send bakery product to product cart 37](#_Toc409365367)

[2.3.13 Wholesaler view their order information 38](#_Toc409365368)

[2.3.14 Employee view all ordered information 39](#_Toc409365369)

[2.3.15 Admin create ingredient information 40](#_Toc409365370)

[2.3.16 Admin edit ingredient information 41](#_Toc409365371)

[2.3.17 Admin delete ingredient information 42](#_Toc409365372)

[2.3.18 Admin view ingredient list 43](#_Toc409365373)

[2.3.19 Admin view ingredient information 44](#_Toc409365374)

[2.3.20 Admin create menu recipe information 45](#_Toc409365375)

[2.3.21 Admin edit menu recipe information 46](#_Toc409365376)

[2.3.22 Admin delete menu recipe information 47](#_Toc409365377)

[2.3.23 Employee view current ingredient estimation 48](#_Toc409365378)

[2.3.24 Employee view previous ingredient estimation 49](#_Toc409365379)

[2.3.25 Employee and Wholesaler make a report 50](#_Toc409365380)

[2.3.26 Employee and Wholesaler make a graph 51](#_Toc409365381)

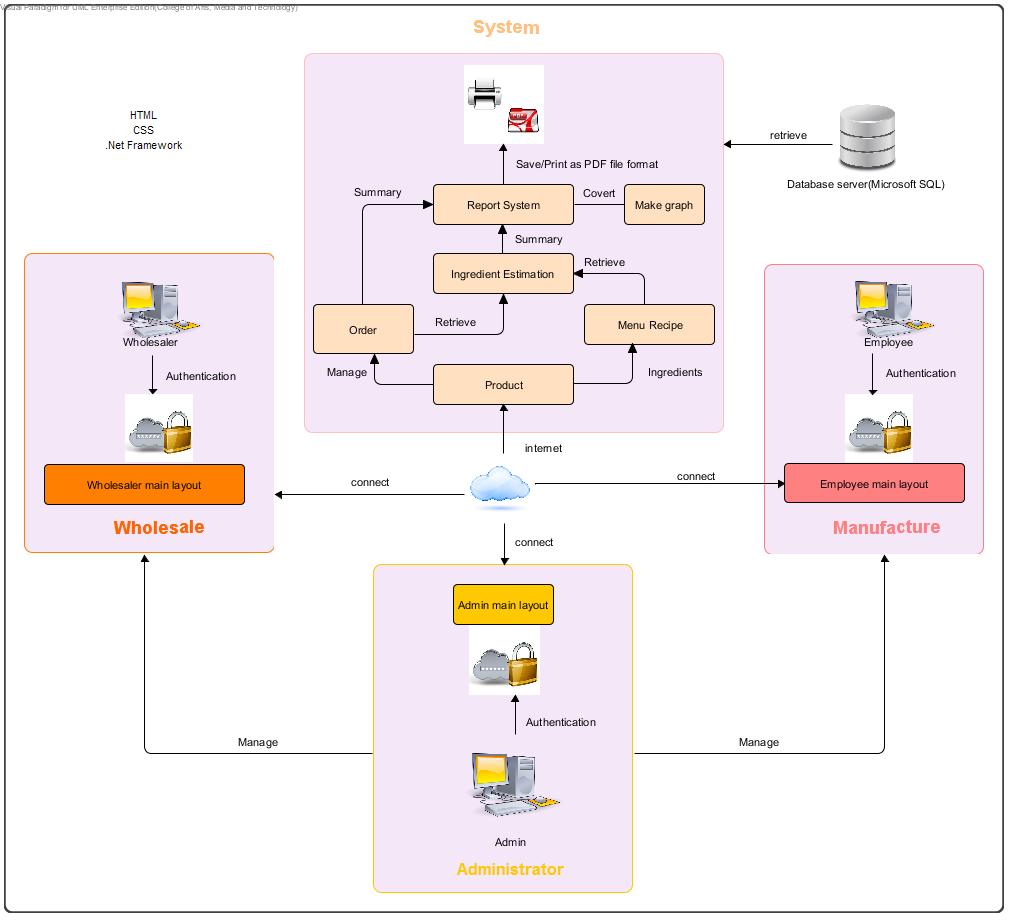
[**Chapter Three | Data Architecture** 52](#_Toc409365382)

[3.1 Entity Relationship Diagram 52](#_Toc409365383)

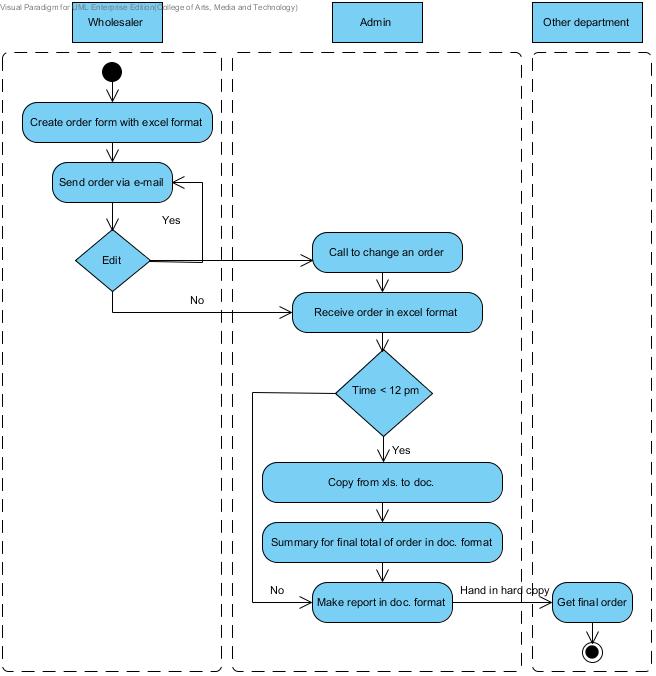
[3.2 ER Diagram Description 53](#_Toc409365384)

[**Chapter Four | User Interface** 62](#_Toc409365385)

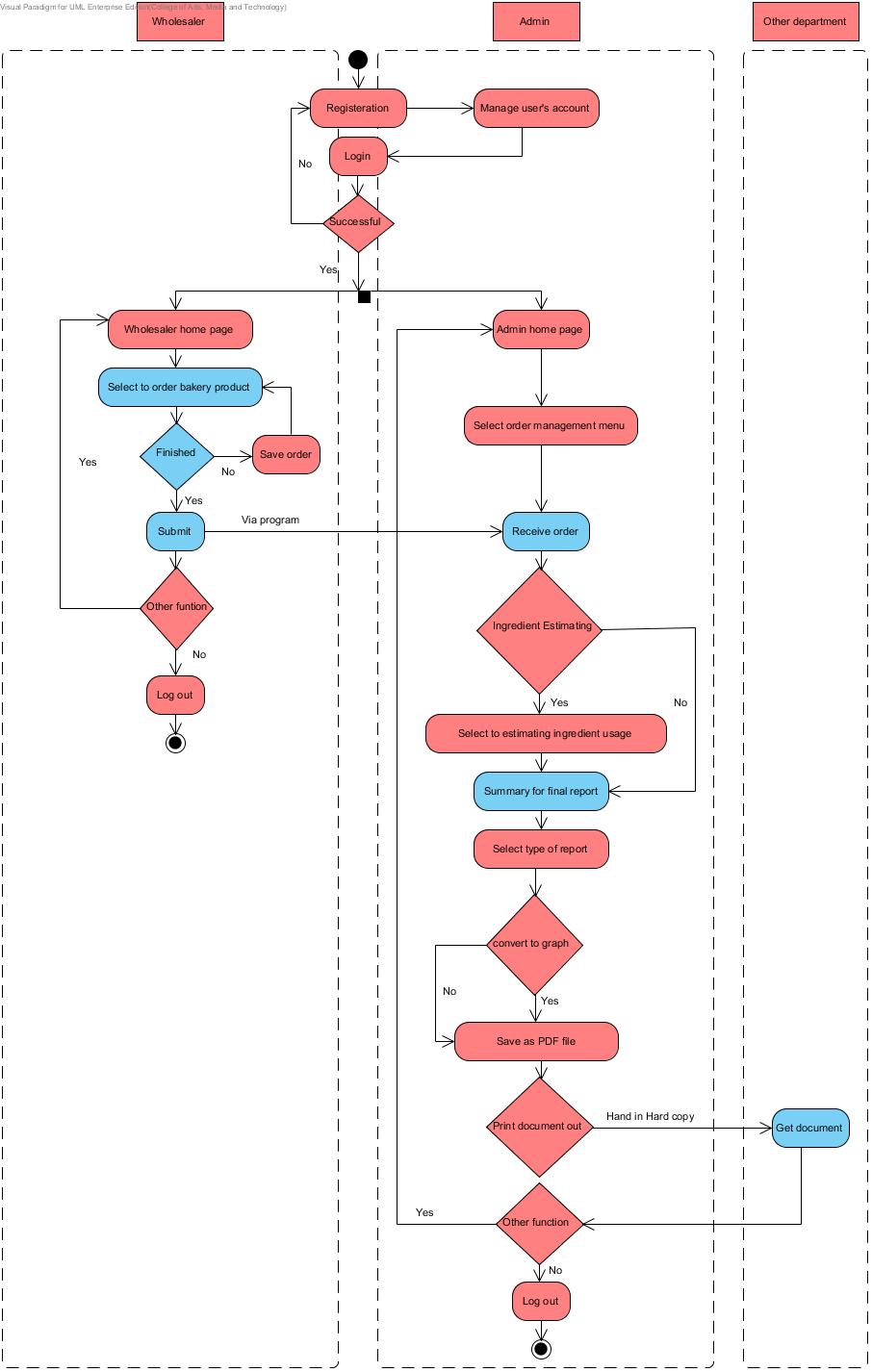
# **Chapter One |** **System Architecture**



**Figure 1: Web-based Ordering & Ingredient Estimating for Bakery Manufacture**



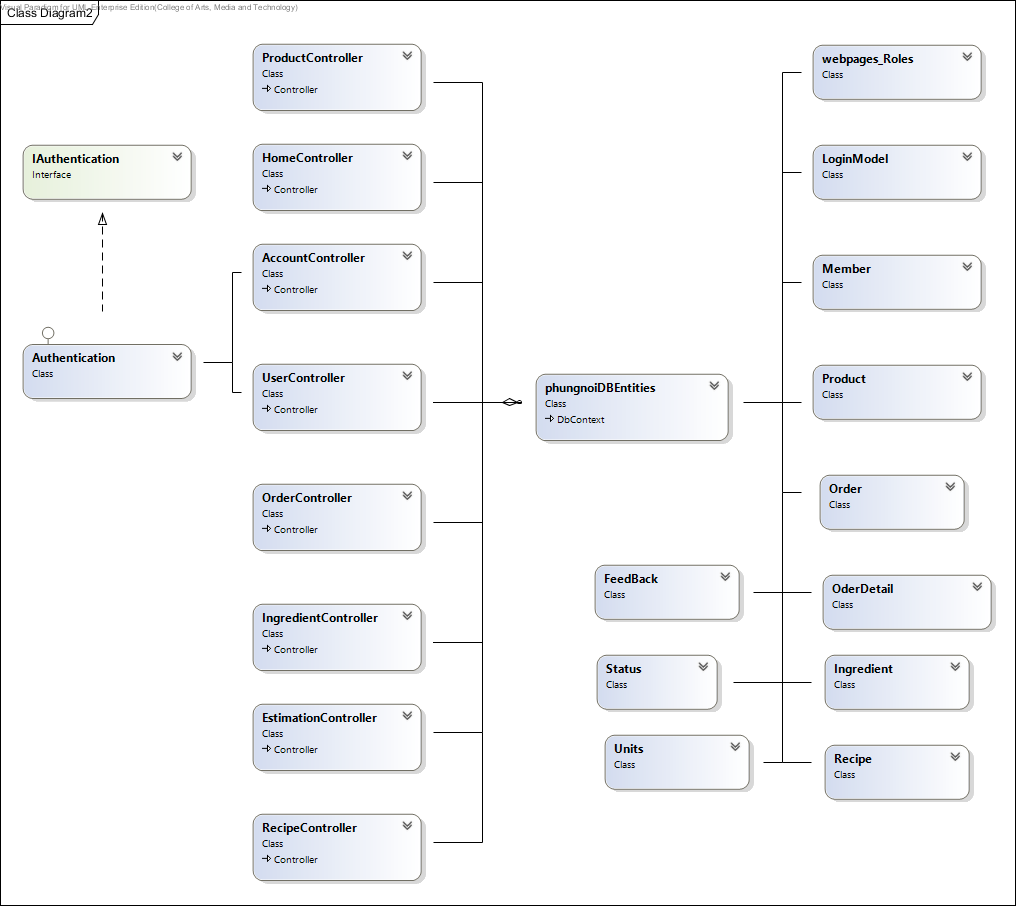
**Figure 2: Flow chart of the original workflow**



**Figure 3: Flow chart of the Web-based Ordering & Ingredient Estimating for Bakery Manufacture**

# **Chapter Two |Detailed Design**

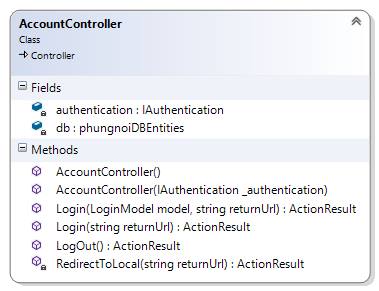
## Class Diagram



**Figure 4: Class Diagram**

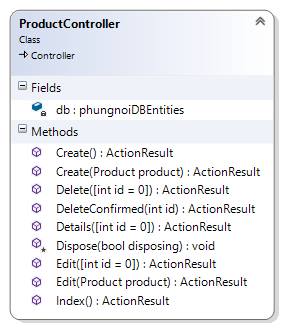
## Class Diagram Description

### 2.2.1 Controller Class



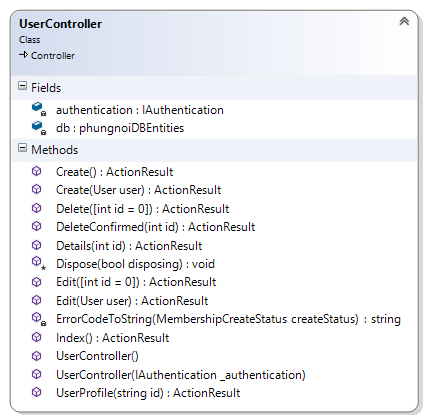
**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Login() | This method uses for logging in to the system. | **Return Type**: ActionResult  **Parameters:** model; LoginModel, returnUrl; String |
| 2 | Logout() | This method uses for logging out from the system. | **Return Type**: ActionResult |



**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Create() | This method uses to call the Create page represent. | **Return Type**: ActionResult |
| 2 | Create() | This method used to add the product into the database. | **Return Type**: ActionResult  **Parameters:** product; Product |
| 3 | Delete() | This method uses to call the Delete page represent. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 4 | DeleteConfirmed() | This method uses to delete the product from the database**.** | **Return Type**: ActionResult  **Parameters:** id; int |
| 5 | Edit() | This method uses to call the Edit page represent. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 6 | Edit() | This method used to update the product information into the database. | **Return Type**: ActionResult  **Parameters:** product; Product |
| 7 | Details() | This method uses to call the details of the product. | **Return Type**: ActionResult  **Parameters:** int id = 0 |
| 8 | Index() | This method uses to call the index page represent. | **Return Type**: ActionResult |



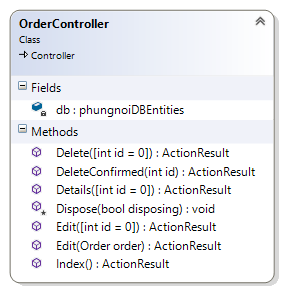
**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Create() | This method uses to call the Create page represent. | **Return Type**: ActionResult |
| 2 | Create() | This method used to add the product into the database. | **Return Type**: ActionResult  **Parameters:** user; User |
| 3 | Delete() | This method uses to call the Delete page represent. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 4 | DeleteConfirmed() | This method uses to delete the product from the database**.** | **Return Type**: ActionResult  **Parameters:** id; int |
| 5 | Edit() | This method uses to call the Edit page represent. | **Return Type**: ActionResult  **Parameters:** id = 0;int |
| 6 | Edit() | This method used to update the product information into the database. | **Return Type**: ActionResult  **Parameters:** user; User |
| 7 | Details() | This method uses to call the details of the product. | **Return Type**: ActionResult  **Parameters:** id; int |
| 8 | Index() | This method uses to call the index page represent. | **Return Type**: ActionResult |
| 9 | UserProfile() | This method uses to call the Member Profile page represent. | **Return Type**: ActionResult  **Parameters:** id; string |



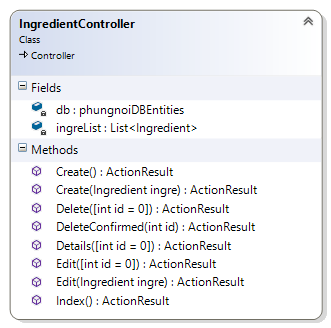
**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | About() | This method uses to call the About page represent. | **Return Type**: ActionResult |
| 2 | Cart() | This method uses to call the Cart page represent. | **Return Type**: ActionResult |
| 3 | Contact() | This method uses to call the Contact page represent. | **Return Type**: ActionResult |
| 4 | CopyOrder() | This method uses to load the existing ordered from the database**.** | **Return Type**: ActionResult  **Parameters:** orderId; int |
| 5 | History() | This method uses to call the History page represent. | **Return Type**: ActionResult |
| 6 | Product() | This method uses to call the Product page represent. | **Return Type**: ActionResult |
| 7 | Product() | This method uses to add a set of products to the product cart. | **Return Type**: ActionResult  **Parameters:** products; IList<product> |
| 8 | SendCart() | This method uses to save or send the product cart to the database. | **Return Type**: ActionResult  **Parameters:** ord; Order, submit; string |
| 9 | Index() | This method uses to call the index page represent. | **Return Type**: ActionResult |



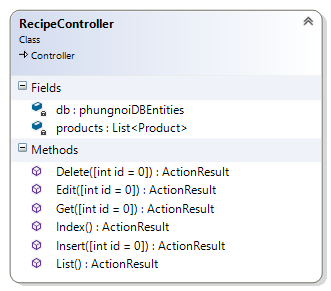
**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Delete() | This method uses to call the Delete page represent. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 2 | DeleteConfirmed() | This method uses to delete the ordered from the database**.** | **Return Type**: ActionResult  **Parameters:** id; int |
| 3 | Edit() | This method uses to call the Edit page represent. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 4 | Edit() | This method used to update the ordered information into the database. | **Return Type**: ActionResult  **Parameters:** order; Order |
| 5 | Details() | This method uses to call the details of the ordered. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 6 | Index() | This method uses to call the index page represent. | **Return Type**: ActionResult |



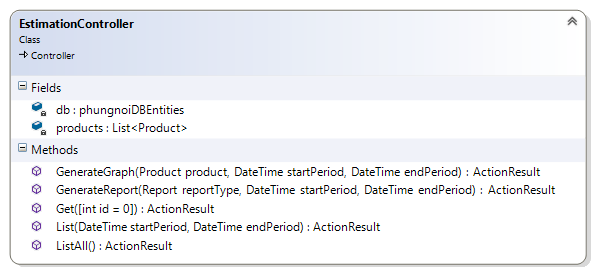
**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Create() | This method uses to call the Create page. | **Return Type**: ActionResult |
| 2 | Create() | This method uses to add the product into the database. | **Return Type**: ActionResult  **Parameters:** ingre; Ingredient |
| 3 | Delete() | This method uses to call the Delete page. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 4 | DeleteConfirmed() | This method uses to delete the ingredient from the database**.** | **Return Type**: ActionResult  **Parameters:** id; int |
| 5 | Details() | This method uses to call the ingredient details. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 6 | Edit() | This method uses to call the Edit page. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 7 | Edit() | This method used to update the ingredient information into the database. | **Return Type**: ActionResult  **Parameters:** ingre; Ingredient |
| 8 | Index() | This method uses to call the Index page. | **Return Type**: ActionResult |



**Method:**

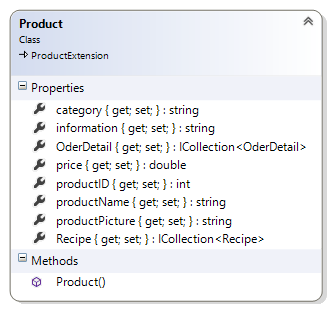
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Delete() | This method uses to delete ingredient from the menu recipe | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 2 | Edit() | This method uses to update ingredient quantity to the menu recipe. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 3 | Get() | This method uses to get a product. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 4 | Index() | This method uses to show all product. | **Return Type**: ActionResult |
| 5 | Insert() | This method uses to insert ingredient name and ingredient quantity to the menu recipe. | **Return Type**: ActionResult  **Parameters:** id = 0; int |
| 6 | List() | This method uses to show a product list according to the condition. | **Return Type**: ActionResult |



**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | GenerateGraph() | This method uses to generate a graph. | **Return Type**: ActionResult  **Parameters:** product; Product, startPeriod; DateTime, endPeriod; DateTime |
| 2 | GenerateReport() | This method uses to generate a report. | **Return Type**: ActionResult  **Parameters:** reportType; Report, startPeriod; DateTime, endPeriod; DateTime |
| 3 | Get () | This method uses to show a product that use to estimating ingredients. | **Return Type**: ActionResult |
| 4 | List () | This method uses to show a product list that estimation ingredient in the period. | **Return Type**: ActionResult  **Parameters:** startPeriod; DateTime, endPeriod; DateTime |
| 5 | ListAll() | This method uses to show all product list in the ingredient estimation. | **Return Type**: ActionResult |

### 2.2.2 Model Class

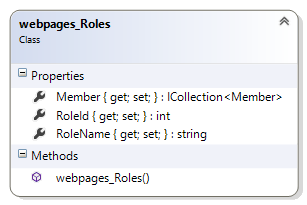


**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | category | This property uses for storing the product category into the database. | **Type**: string |
| 2 | information | This property uses for storing the product information into the database. | **Type**: string |
| 3 | OderDetail | This property uses for storing the order details into the database. | **Type**: ICollection<OrderDetail> |
| 4 | price | This property uses for storing the product price into the database | **Type**: double |
| 5 | productID | This property uses for storing the product identification into the database. | **Type**: int |
| 6 | productName | This property uses for storing the product name into the database. | **Type**: string |
| 7 | productPicture | This property uses for storing the product picture into the database. | **Type**: string |
| 8 | Recipe | This property uses for storing the product price into the database. | **Type**: ICollection<Recipe> |

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Product() | This method is a constructor of product. | **Return Type**: ActionResult |

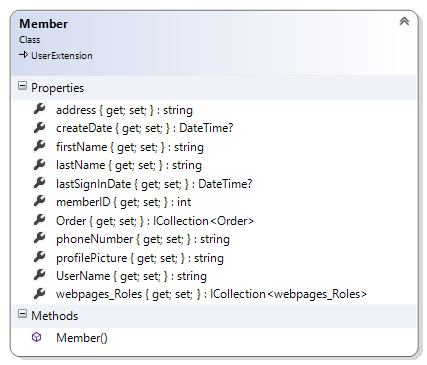


**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | RoleId | This property uses for separating the member identification from the database. | **Type**: int |
| 2 | RoleName | This property uses for separating the role name of all member into the database. | **Type**: string |
| 3 | Member | This property uses for storing the member’s information into the database. | **Type**: ICollection <Member> |

**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Webpage\_Roles() | This method is a constructor of Webpage\_Roles. | **Return Type**: ActionResult |



**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | address | This property uses for storing the address of a different member into the database. | **Type**: string |
| 2 | createDate | This property uses for storing the date of an activated member into the database. | **Type**: DateTime |
| 3 | firstName | This property uses for storing the first name of a member into the database. | **Type**: string |
| 4 | lastName | This property uses for storing the last name of a member into the database. | **Type**: string |
| 5 | lastSignInDate | This property uses for storing the last sign in date of a member into the database. | **Type**: DateTime |
| 6 | memberID | This property uses for storing the identification of a member into the database. | **Type**: int |
| 7 | Order | This property uses for storing the order into the database. | **Return Type**: Collection <Order> |
| 8 | phoneNumber | This property uses for storing the phone number of a different member into the database. | **Type**: string |
| **ID** | **Name** | **Description** | **Remark** |
| 9 | profilePicture | This property uses for storing the member’s picture into the database. | **Type**: string |
| 10 | UserName | This property uses for storing the email of a different member into the database. | **Return Type**: string |
| 11 | Webpage\_roles | This property uses for storing the role of each member into the database. | **Return Type**: Collection<Webpage\_roles> |

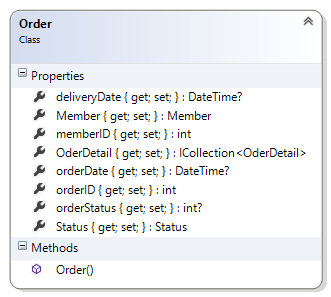
**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Member() | This method is a constructor of Member. | **Return Type**: ActionResult |



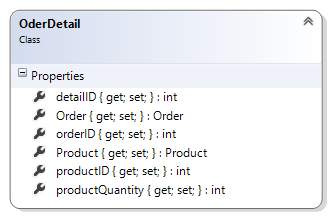
**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Password | This property uses for storing the password of the member into the database. | **Return Type**: string |
| 2 | UserName | This property uses for storing the username into the database. | **Return Type**: string |
| 3 | RememberMe | This property uses for storing the login status into the database. | **Return Type**: Boolean |



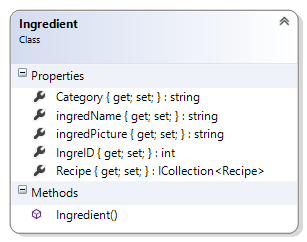
**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | deliveryDate | This property uses for storing the due date into the database. | **Type**: DateTime |
| 2 | orderID | This property uses for storing the order identification into the database. | **Type**: int |
| 3 | OderDetail | This property uses for storing the ordered information into the database. | **Type**: ICollection<OrderDetail> |
| 4 | orderDate | This property uses for storing the ordered date into the database. | **Type**: DateTime |
| 5 | orderStatus | This property uses for storing the order status of the member into the database. | **Type**: int |
| 6 | memberID | This property uses for storing the member identification of the member into the database. | **Type**: int |
| 7 | Member | This property uses for storing the member’s profile into the database. | **Type**: Member |
| 8 | Status | This property uses for storing the member’s status into the database. | **Type**: Status |



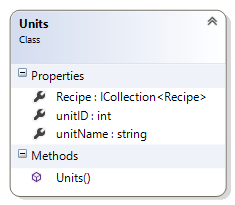
**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | detailID | This property uses for storing the order’s detail identification into the database. | **Type**: int |
| 2 | Order | This property uses for storing the order into the database. | **Type**: Order |
| 3 | orderID | This property uses for storing the order identification into the database. | **Type**: int |
| 4 | product | This property uses for storing the Product into the database. | **Type**: Product |
| 5 | productID | This property uses for storing the Product identification into the database. | **Type**: int |
| 6 | productQuantity | This property uses for storing the product quantity into the database. | **Type**: int |



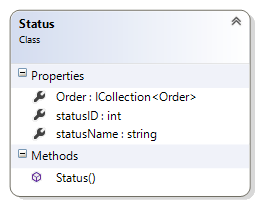
**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Category | This property uses for storing the ingredient category into the database. | **Type**: string |
| 2 | IngreID | This property uses for storing the ingredient identification into the database. | **Type**: int |
| 3 | IngredName | This property uses for storing the ingredient name into the database. | **Type**: string |
| 4 | IngredPicture | This property uses for storing the ingredient picture into the database. | **Type**: string |
| 5 | Recipe | This property uses for storing the menu recipe into the database. | **Type**: ICollection<Recipe> |



**Property:**

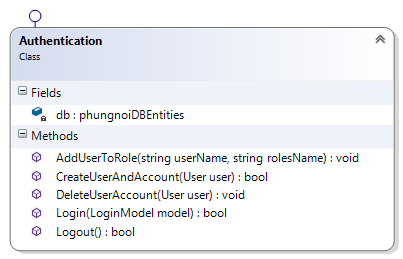
|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Recipe | This property uses for storing the menu recipe into the database. | **Type**: ICollection<Recipe> |
| 2 | unitID | This property uses for storing the units identification into the database. | **Type**: int |
| 3 | unitName | This property uses for storing the units name into the database. | **Type**: string |



**Property:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | Order | This property uses for storing the order into the database. | **Type**: ICollection<Order> |
| 2 | StatusID | This property uses for storing the status identification into the database. | **Type**: int |
| 3 | StatusName | This property uses for storing the status name into the database. | **Type**: string |

### 2.2.3 Service Class



**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Name** | **Description** | **Remark** |
| 1 | AddUserToRole() | This method uses to add a role of a member. | **Return Type**: void  **Parameters:** username ;string, rolesName; string |
| 2 | CreateUserAndAccount() | This method uses to create a new member. | **Return Type**: Boolean  **Parameters:** user; User |
| 3 | DeleteUserAccount() | This method uses to delete a member from the database. | **Return Type**: void  **Parameters:** user; User |
| 4 | Login() | This method uses to login before using all functional in the website. | **Return Type**: Boolean  **Parameters:** model; LoginModel |
| 5 | Logout() | This method uses to logout member exit from the website. | **Return Type**: Boolean |

## 2.3 Sequence Diagram

### 2.3.1 Admin can activate member



**Figure 5: SD-01 Admin can activate user**

First, the admin creates a member by input the member’s information and submits to the controller in UI-05. Then, the controller will validate the existing information of the member and the model state availability. If the information passed those validations. It will assign the role for a member. Then, it will redirect to the member profile page.

### 2.3.2 Member can login



**Figure 6: SD-02 Member can login**

All of the members type have to login before using the system. First, all of them must inputs username and password in UI-01. Then selects login button, AccountController will check the model. If the authentication valid with the existing information, the system will call action RedirectTolocal. Else the member no permission, the controller will redirect to the login page.

### 2.3.3 Member can logout

****

**Figure 7: SD-03 member can logout**

All of the members type can logout from the system. First, all of them select a menu list on the top right of the homepage. Then selects logout button to the system in UI-15. When the member confirms for logging out, the system will call action Logout from AccountController to the Authentication. Then the system returns logout to AccoutController in UI-01.

### 2.3.4 Admin can edit member information



**Figure 08: SD-04 Admin can edit member information**

First, the admin edits a member’s information by select edit link and inputs new update information in UI-06. Then the system calls action edit from UserController and calls FirstOrDefault from, Dbset<user> to find a username of the member in the database. If the identification of the username is equal to founded identification, the system will update new member information to the database.

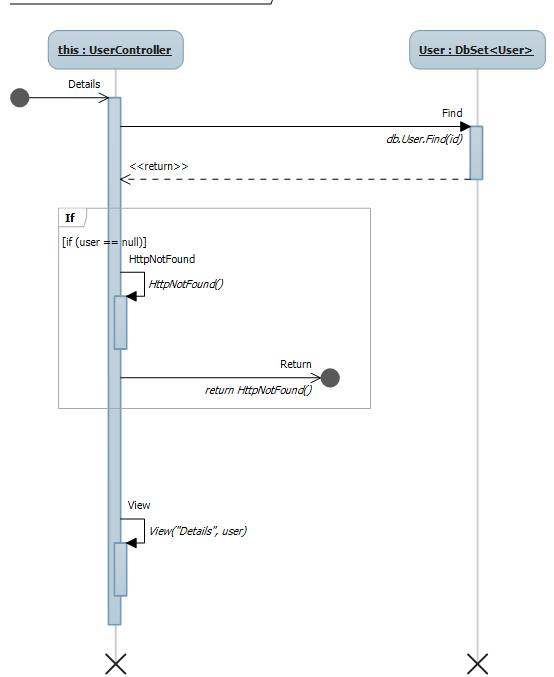
### 2.3.5 Admin can delete member



**Figure 09: SD-05 Admin can delete member**

First, the admin deletes a member by select delete link and submits to the controller in UI-07. Then, the controller will find identification of a member and remove it from the database. Then, it will redirect to the home page.

### 2.3.6 Admin can view member information



**Figure 10: SD-06 Admin can view member information**

First, the admin view member information by select view detail link. Then, the controller will find identification of the member and return member information to display on the screen in UI-09. Then, it will redirect to member profile page.

### 2.3.7 Admin can create bakery product



**Figure 11: SD-07 Admin can create bakery product**

The admin can create new bakery product information. First, the admin selects a product management link on the main layout page of the admin. Then selects create new bakery product and input all bakery product information in UI-10. When the admin confirms adding new bakery product information, the system will call create action to ProductController and check the model. If valid model, ProductController will call add action to Dbset<Product> in the database. After save changed, the system will return to the main layout page of the admin.

### 2.3.8 Admin can edit bakery product information



**Figure 12: SD-08 Admin can edit bakery product information**

First, the admin edits bakery product by inputs update bakery product information in UI-11 and submits to the controller. Then, the system will call edit action to ProductController and check the model. If the model is valid, ProductController will call Entry<product> action to updates new bakery product information and then save a change in the database. Then, controller will call a list of all bakery product before return to the home page.

### 2.3.9 Admin can delete bakery product

****

**Figure 13: SD-09 Admin can delete bakery product**

First, the admin deletes bakery product by select a delete link and submits to the controller in UI-12. The system will call DeleteConfirmed action to ProductController and find the product identification that match to the required product. Then return the product to ProductController and call remove action to remove the product from the database. After finished controller will remove that product from a list of all bakery product and return to the main layout page of the admin.

### 2.3.10 Member can view bakery product list



**Figure 14: SD-10 Member can view bakery product list**

First, on the main menu of the home page. All of the members can view all bakery product information by select the product page. The system will call product action to HomeController then check a filter keyword from a search box. If have a filter keyword from a member, HomeController will call LINQ action to find a list of products from Dbset<product> and return a list of products in UI-13. Else, HomeController displays a list of

all bakery product from the database.

### 2.3.11 Wholesaler order bakery product

**Figure 15: SD - 11 Wholesaler order bakery product**

The wholesaler has to access the HomeController before creating a new order in the system. First, the wholesaler must inputs new product name, product quantity, and delivery date into the product cart in UI-22. Then HomeController will check the bakery product number in the product cart. If the input information is more than zero, it will return a product to HomeController.

### 2.3.12 Wholesaler send bakery product to product cart



**Figure 16: SD - 12 Wholesaler send bakery product to own cart**

The wholesaler has to access the HomeController before sending a new order to the system. First, the wholesaler must selects an add button to add a bakery product into the product cart. Then HomeController will add a bakery product into the product cart. If the input information is valid, it will return the product cart page to the wholesaler.

### 2.3.13 Wholesaler view their order information



**Figure 17: SD - 13 Wholesaler view their order information**

The wholesaler has to access the HomeController before viewing own ordered information in the system. First, the wholesaler must login to the system and selects order bar to view current ordered information in UI-23. Then HomeController will check the member identification and retrieve their ordered information to display at HomeController.

### 2.3.14 Employee view all ordered information



**Figure 18: SD - 14 Employee view all ordered information**

The employee has to access the OrderController before viewing all ordered information in the system. First, the employee must login to the system and selects order management link to access the order management page. Then OrderController will check the ordered information and retrieve all ordered information to display at OrderController in UI-25.

### 2.3.15 Admin create ingredient information

****

**Figure 19: SD - 15 Admin create ingredient information**

The admin has to access the IngredientController before creating new ingredient information in the system. First, the admin must inputs new ingredient information include ingredient name, category, and ingredient picture to the ingredient form. Then select save button, IngredientController will check the ingredient form. If the input information is valid, it will save new ingredient information into the database.

### 2.3.16 Admin edit ingredient information

****

**Figure 20: SD - 16 Admin edit ingredient information**

The admin has to access the IngredientController before editing ingredient information in the system. First, the admin must inputs updated ingredient information include ingredient name, category, and ingredient picture to the ingredient form. Then selects save button, IngredientController will check the ingredient form. The system will update ingredient information into the database. Then the system returns updated ingredient information to IngredientController.

### 2.3.17 Admin delete ingredient information

****

**Figure 21: SD - 17 Admin delete ingredient information**

The admin has to access the IngredientController before deleting ingredient information in the system. First, the admin deletes ingredient by select to delete the existing information and confirm to delete. Then the system calls ingredient information from the database. If the admin saves the action, the system shall clear the existing ingredient

information from the database.

### 2.3.18 Admin view ingredient list

****

**Figure 22: SD-18 Admin view ingredient list**

First, the admin has to access the ingredient page. Then the admin searches the ingredient information by input the filter keyword and submits to the controller. The controller will validate the existing information with the filter keyword and displays the related results. All ingredient information are retrieved from the phungnoi database, ingredient table. Then, it will return the required data to the admin in the ingredient information page.

### 2.3.19 Admin view ingredient information

****

**Figure 23: SD-19 Admin view ingredient information**

First, the admin has to access the ingredient information page and request the require data to the IngredientController. Then, the controller will find ingredient identification are matched with the existing information and return it to the admin. The admin can view the ingredient information from the ingredient information page.

### 2.3.20 Admin create menu recipe information

****

**Figure 24: SD-20 Admin create menu recipe information**

First, the admin has to access the menu recipe page and request to add the ingredients to the menu recipe in UI-26. Then, the controller will send a request and return ingredient information that inserted to the admin. The admin can view the menu recipe information from the menu recipe page.

### 2.3.21 Admin edit menu recipe information



**Figure 25: SD-21 Admin edit menu recipe information**

First, the admin has to access the menu recipe page and request to edit the ingredient information; ingredient name, ingredient quantity and units of the menu recipe. Then, the controller will send the updated information and return new ingredient information to the admin. The admin can view the menu recipe information from the menu recipe page.

### 2.3.22 Admin delete menu recipe information



**Figure 26: SD-22 Admin delete menu recipe information**

First, the admin has to access the menu recipe page and request to delete menu recipe information from the system. Then, the controller will send a request and return the empty information to the admin. The admin can view the empty menu recipe information from the menu recipe page.

### 2.3.23 Employee view current ingredient estimation



**Figure 27: SD-23 Employee view current ingredient estimation**

First, the employee has to access the ingredient estimation page and request to Get the ingredient estimation result from the EstimationController. Then, the controller wills estimating ingredient usage and return a result to the employee. The employee can view the estimation result from the ingredient estimation page.

### 2.3.24 Employee view previous ingredient estimation



**Figure 28: SD-24 Employee view previous ingredient estimation**

First, the employee has to access the ingredient estimation page and request a list of all bakery product to the EstimationController. Then, the controller wills estimating ingredient usage with the period and return a result to the employee. The employee can view the estimation result from the ingredient estimation page.

### 2.3.25 Employee and Wholesaler make a report



**Figure 29: SD-25 Employee and Wholesaler make a report**

First, both of the members type has to access the report page and request to generate a report to the EstimationController. Then, the controller will create a report from the existing information with the period and return the report result to the screen in UI-27. The both of members type can view a report from the report page.

### 2.3.26 Employee and Wholesaler make a graph

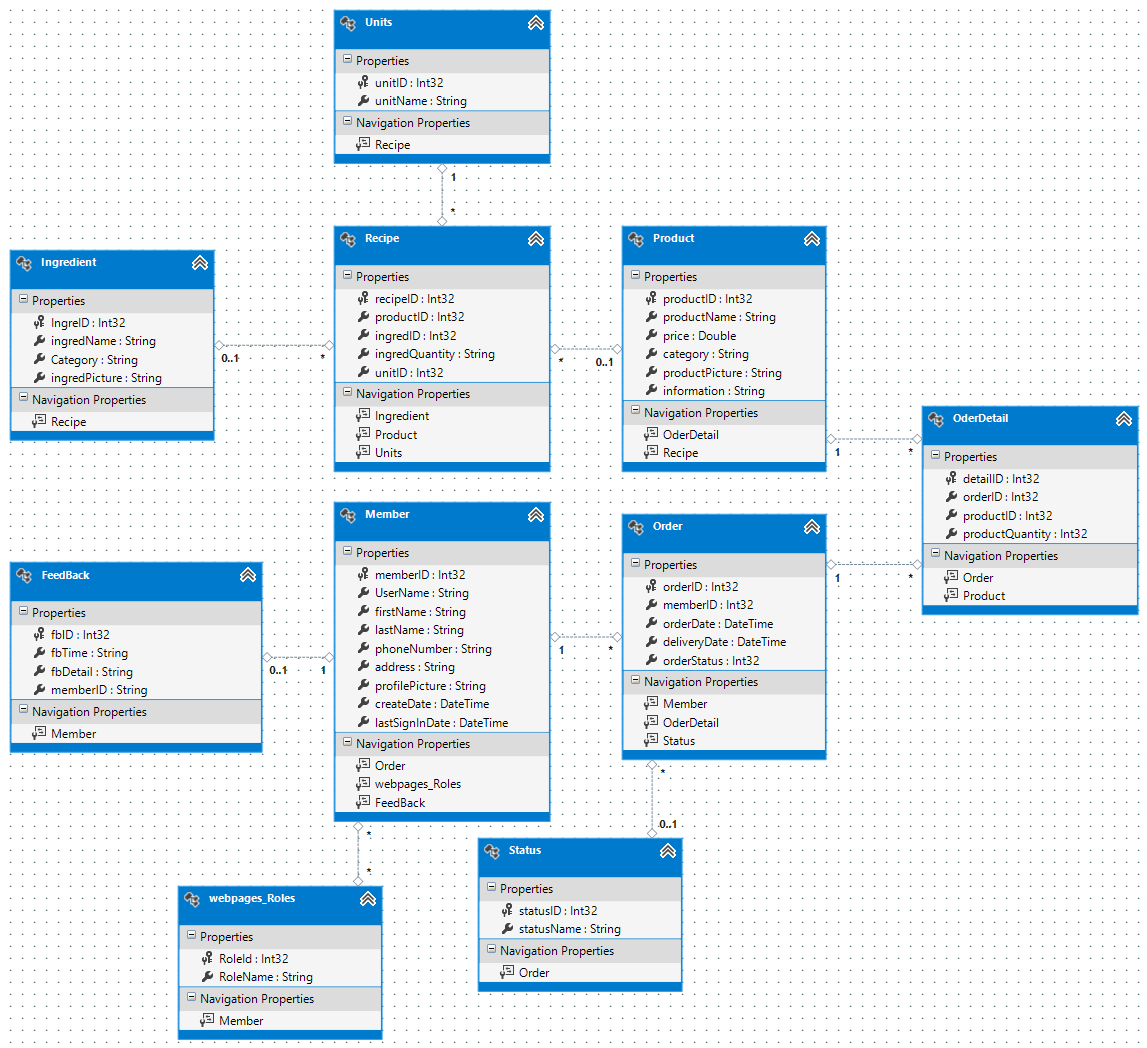


**Figure 30: SD-26 Employee and Wholesaler make a graph**

First, both of the members type has to access the graph page and request to generate a graph to the EstimationController. Then, the controller will create a graph from the statistical data with the period and return the graph image to the screen in UI-28. The both of members can view a graph picture from the graph page.

# **Chapter Three | Data Architecture**

## 3.1 Entity Relationship Diagram



**Figure 31: Entity Relationship Diagram**

## 

## 3.2 ER Diagram Description

**DB-01: Webpages\_Roles**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| RoleId | Identify different role of a member type | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| RoleName | Name of each role in the system | String | - Not Null | - |

**DB-02: Member**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| memberID | Identify different from the member | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| UserName | Username or Email of a member in the system | nvarchar(15) | - Not Null | - |
| firstName | First name of a member | nchar(50) | - Not Null | - |
| lastName | Last name of a member | nchar(50) | - Not Null | - |
| phoneNumber | Telephone number to contact a member | nchar(20) | - Not Null | - |
| address | Home address of a member | nchar(400) | - Not Null | - |
| profilePicture | Path of member’s profile picture | nchar(200) | - Not Null | - |
| lastSignInDate | Last date of member login to the system | DateTime | - Not Null | - |
| createDate | Date of member allows to reach the system | DateTime | - Not Null | - |

**DB-03: Product**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| productID | Number to identify the a bakery product | Integer(10) | -Not Null | - Primary Key  - Auto Incremental  - Unique |
| productName | Name of a bakery product | nvarchar(50) | - Not Null | - |
| productPicture | Path of a bakery’s picture | nvarchar(200) | - Not Null | - |
| category | Category to classify a bakery product | nvarchar(50) | - Not Null | - |
| information | Description of a bakery product | nvarchar(100) | - | - |
| price | Value of a bakery product | float | - Not Null | - |

**DB-04: Recipe**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| recipeID | Number to identify a menu recipe | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| productID | Number to identify a bakery product | Integer(10) | - Not Null | - |
| ingredID | Number to identify an ingredient | Integer(10) | - Not Null | - |
| ingredQuantity | Quantity of each ingredient usage in a bakery product | nvarchar(50) | - Not Null | - |
| unitID | Number to identify the units | Integer(10) | - Not Null | - |

**DB-05: Ingredient**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| IngreID | Number to identify an ingredient | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| ingredName | Ingredient name that content in the menu recipe | nvarchar(50) | - Not Null | - |
| Category | Category of ingredient type | nvarchar(50) | - Not Null | - |
| ingredPicture | Path of ingredient’s picture | nchar(200) | - Not Null | - |

**DB-06: Order**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| orderID | Number to identify an order | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| memberID | Number to identify a member | Integer(10) | - Not Null | - |
| orderDate | The date that ordering a bakery product | DateTime | - Not Null | - |
| deliveryDate | The date that delivery the order | DateTime | - Not Null | - |
| orderStatus | Number to identify the order status | Integer(10) | - Not Null | - |

**DB-07: Order Detail**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| detailID | Number to identify the order detail | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| orderID | Number to identify the order | Integer(10) | - Not Null | - |
| productID | Number to identify the bakery product | Integer(10) | - Not Null | - |
| productQuantity | The quantity of the bakery product | Integer(10) | - Not Null | - |

**DB-08: Units**



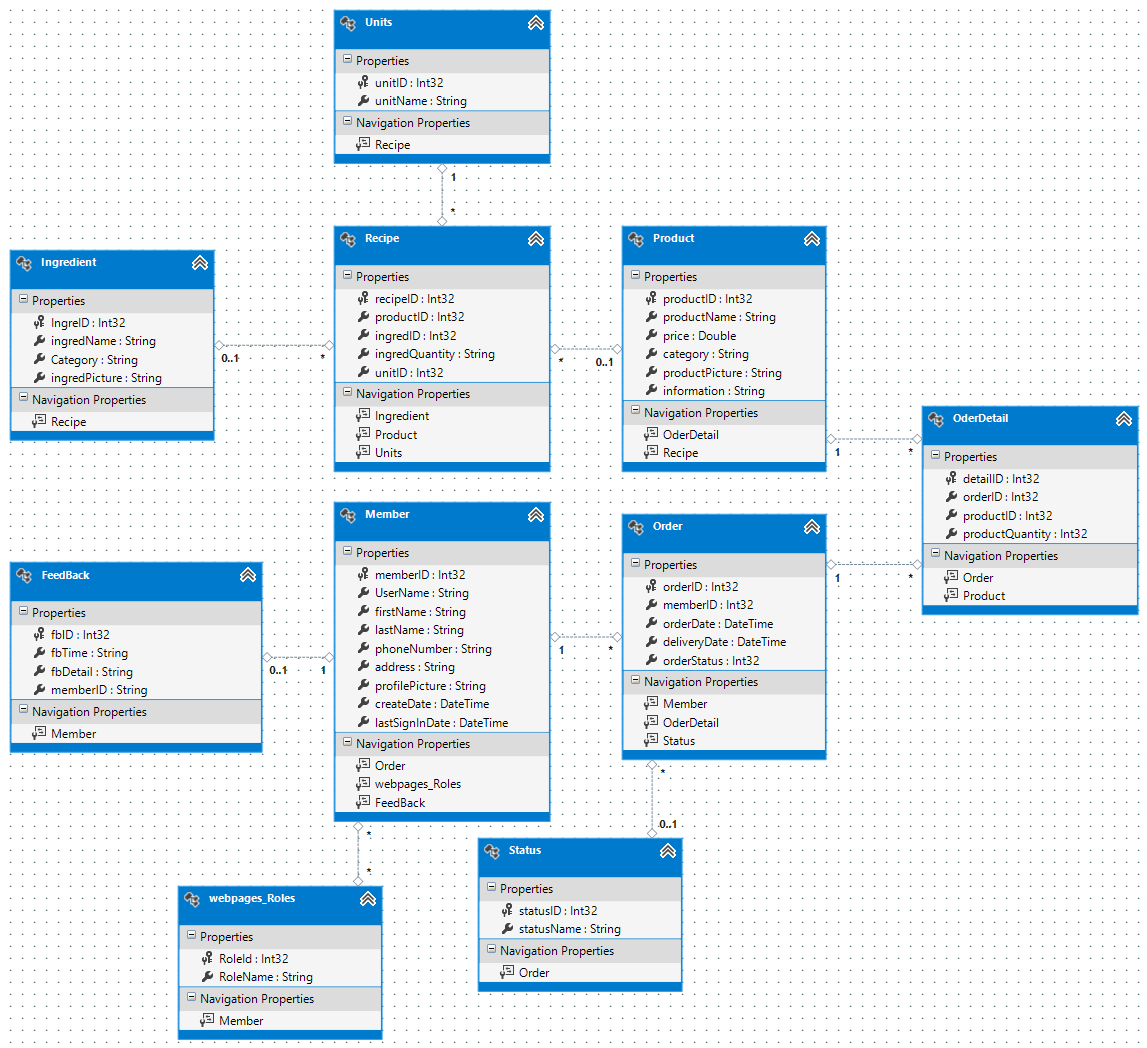
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| unitID | Number to identify the units | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| unitName | Name of the units | nvarchar(50) | - Not Null | - |

**DB-09: Status**



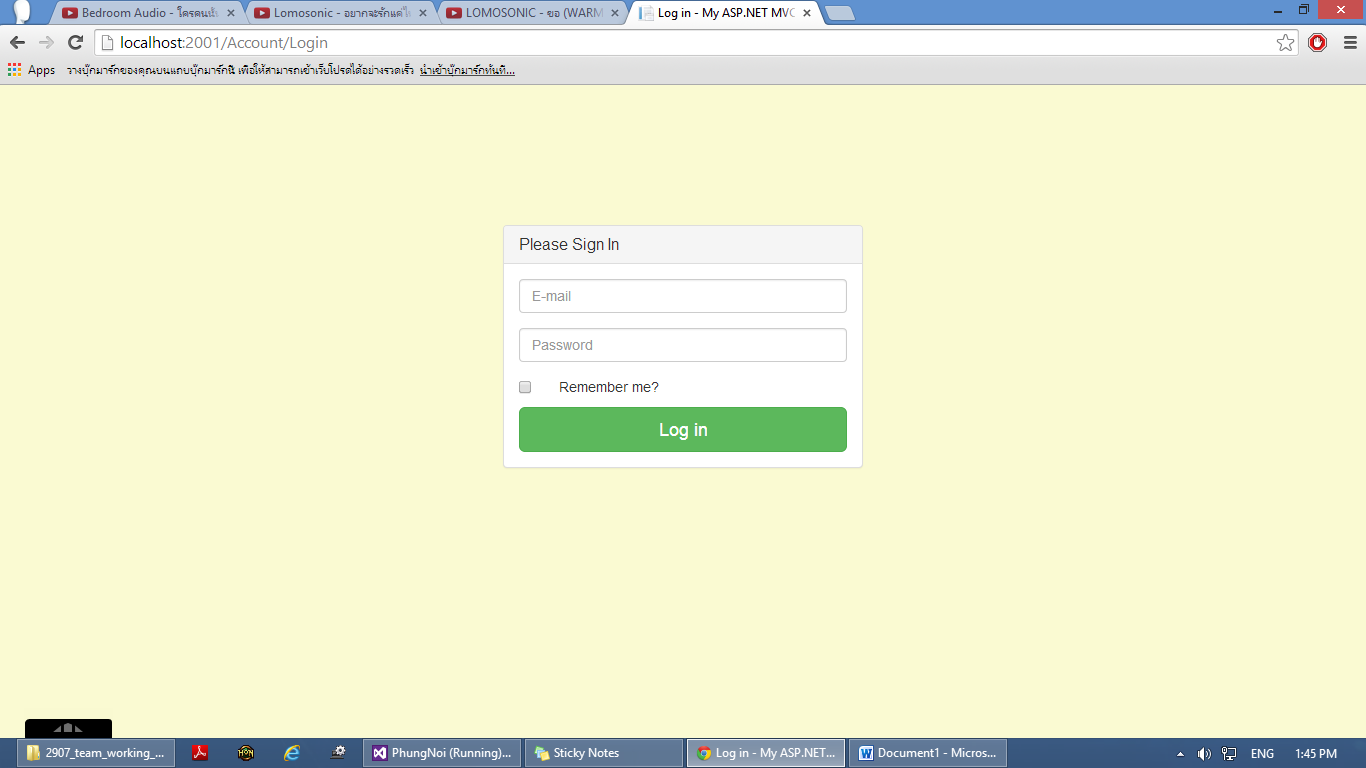
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| statusID | Number to identify the status | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| statusName | The name of the status | nvarchar(50) | - Not Null | - |

**DB-10: Feedback**

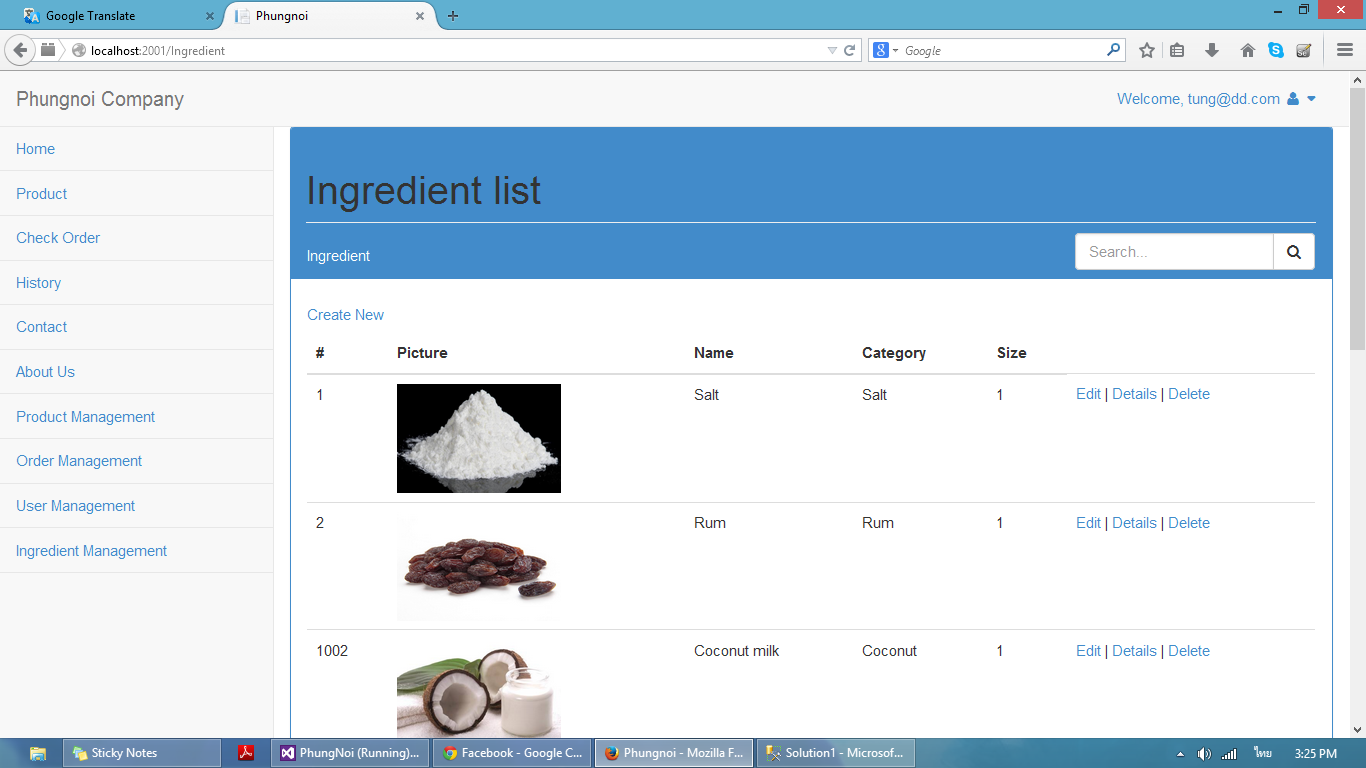


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Description** | **Type** | **Null able** | **Extra** |
| fbID | Number to identify the feedback comments | Integer(10) | - Not Null | - Primary Key  - Auto Incremental  - Unique |
| fbTime | Time of feedback’s comment | datetime | - Not Null | - |
| fbDetail | Information in the feedback box | nvarchar(100) | - Not Null | - |
| memberID | Number to identify a member | Integer(10) | - Not Null | - |

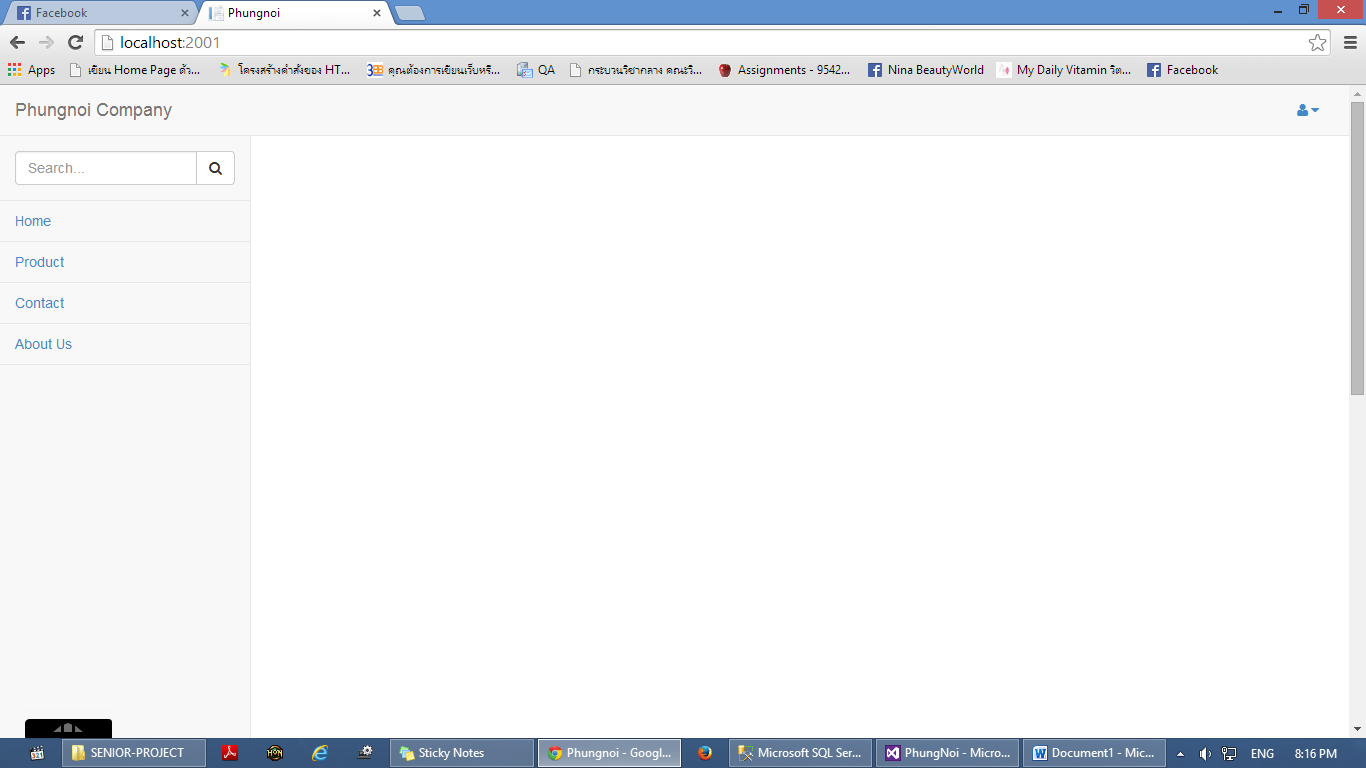
# **Chapter Four | User Interface**



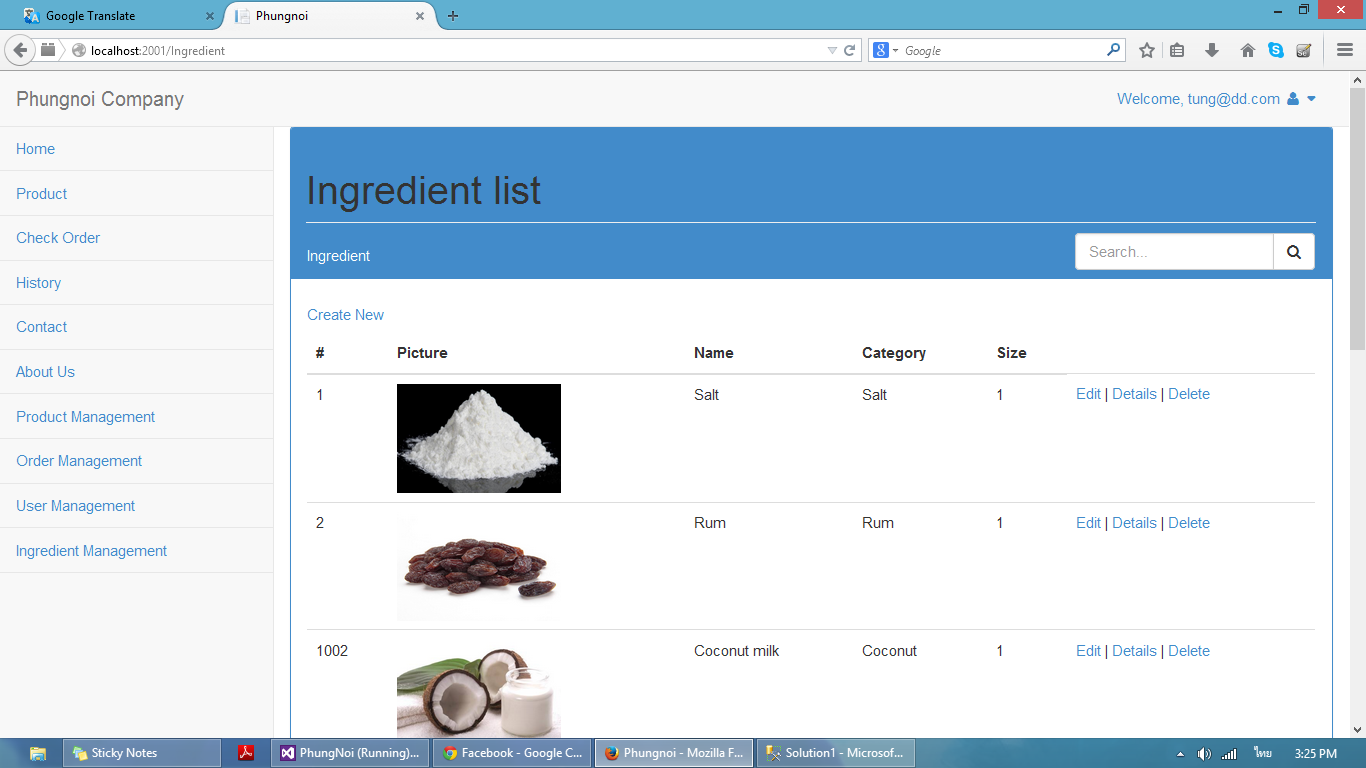
**Figure 32:** UI – 01 Login page



**Figure 33:** UI - 02 Admin main layout page



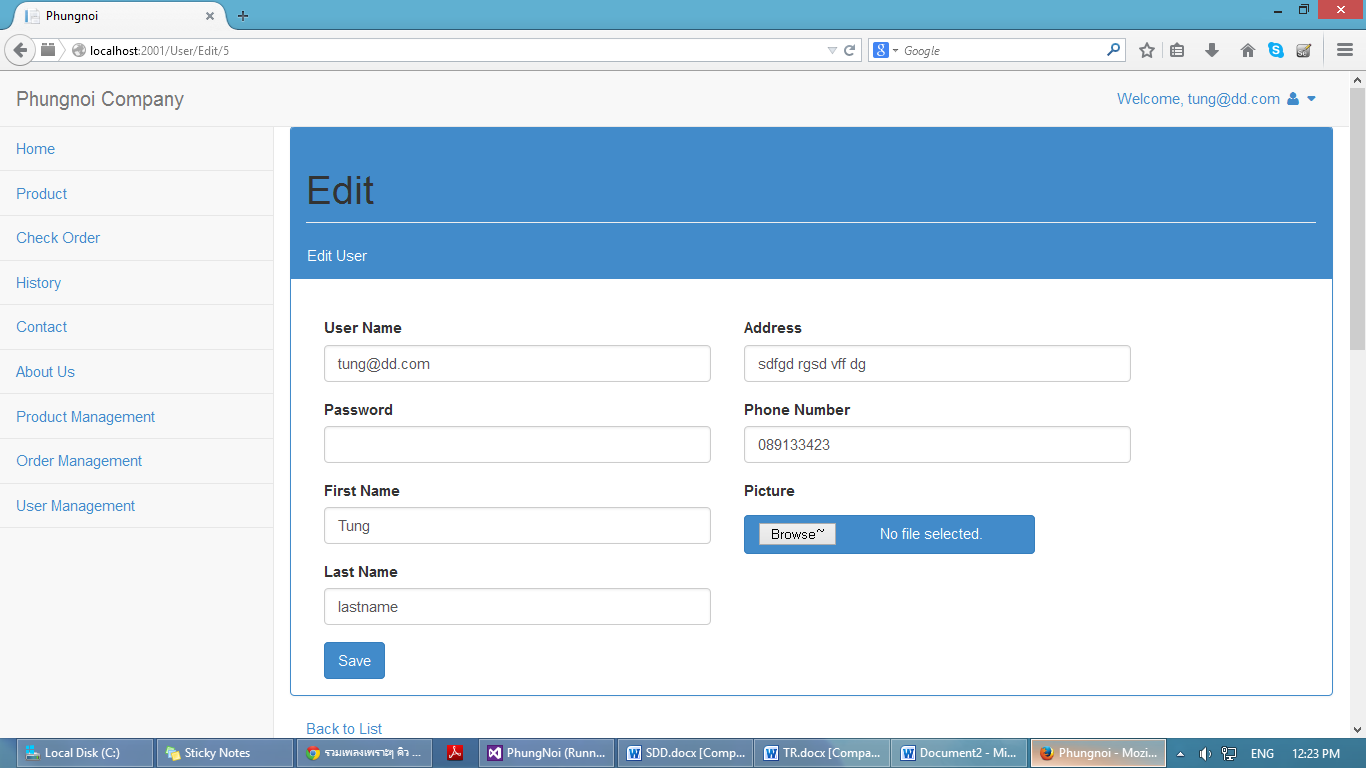
**Figure 34:** UI – 03 Wholesaler main layout page



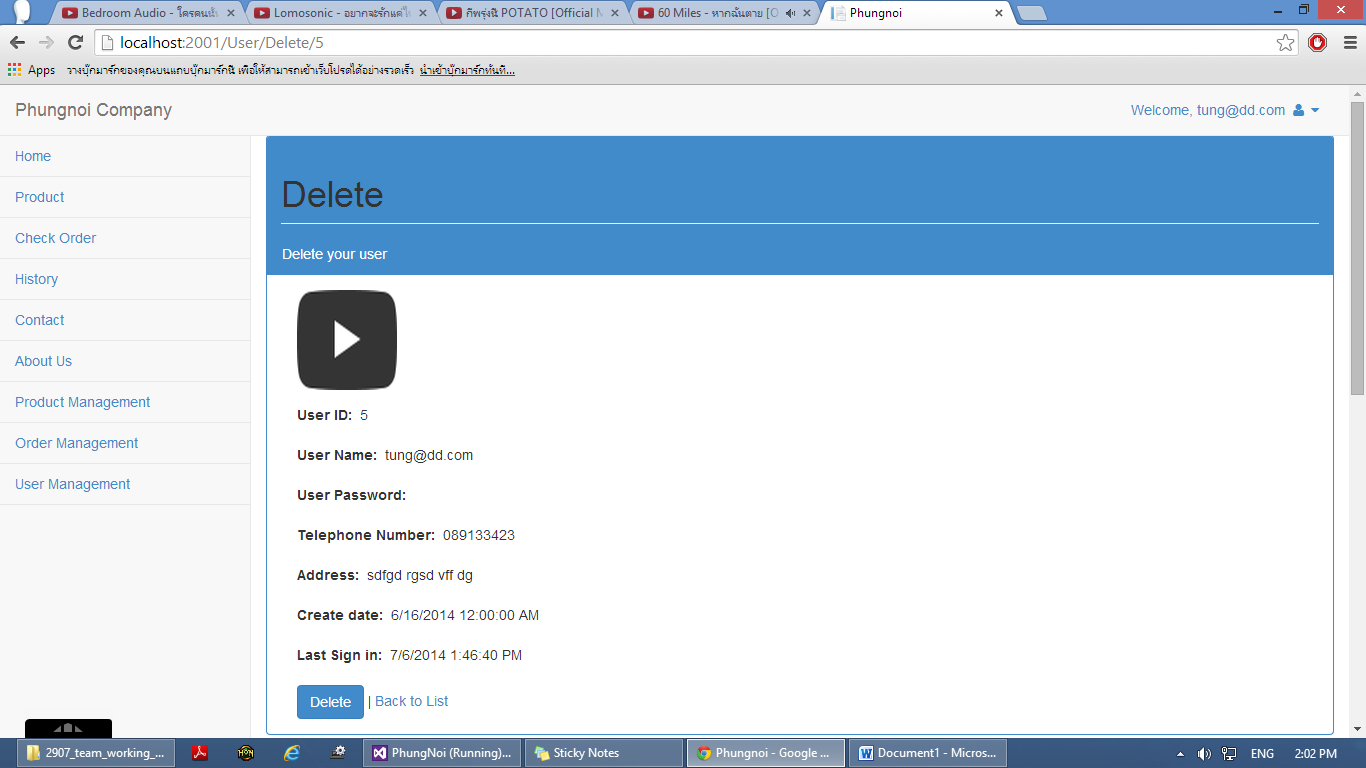
**Figure 35:** UI – 04 Employee main layout page

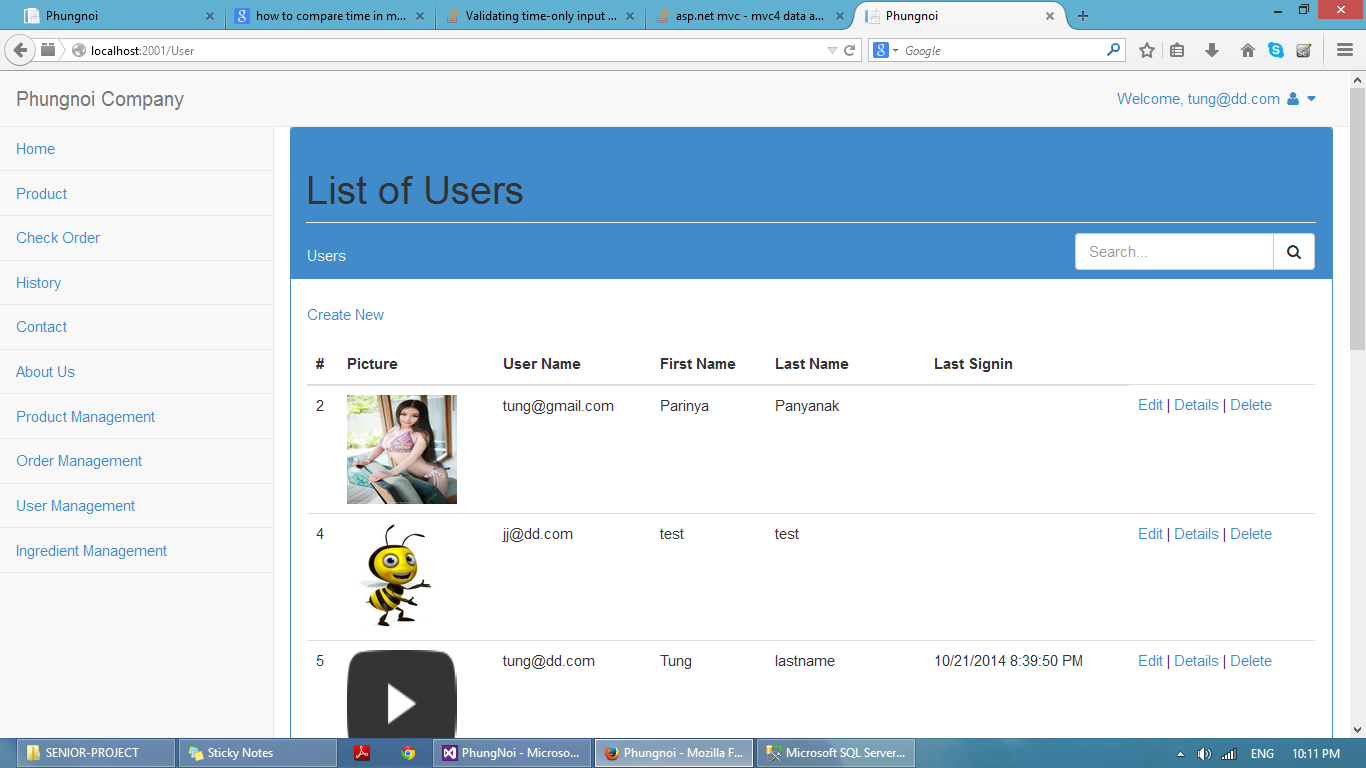


**Figure 36:** UI – 05 Create member information page

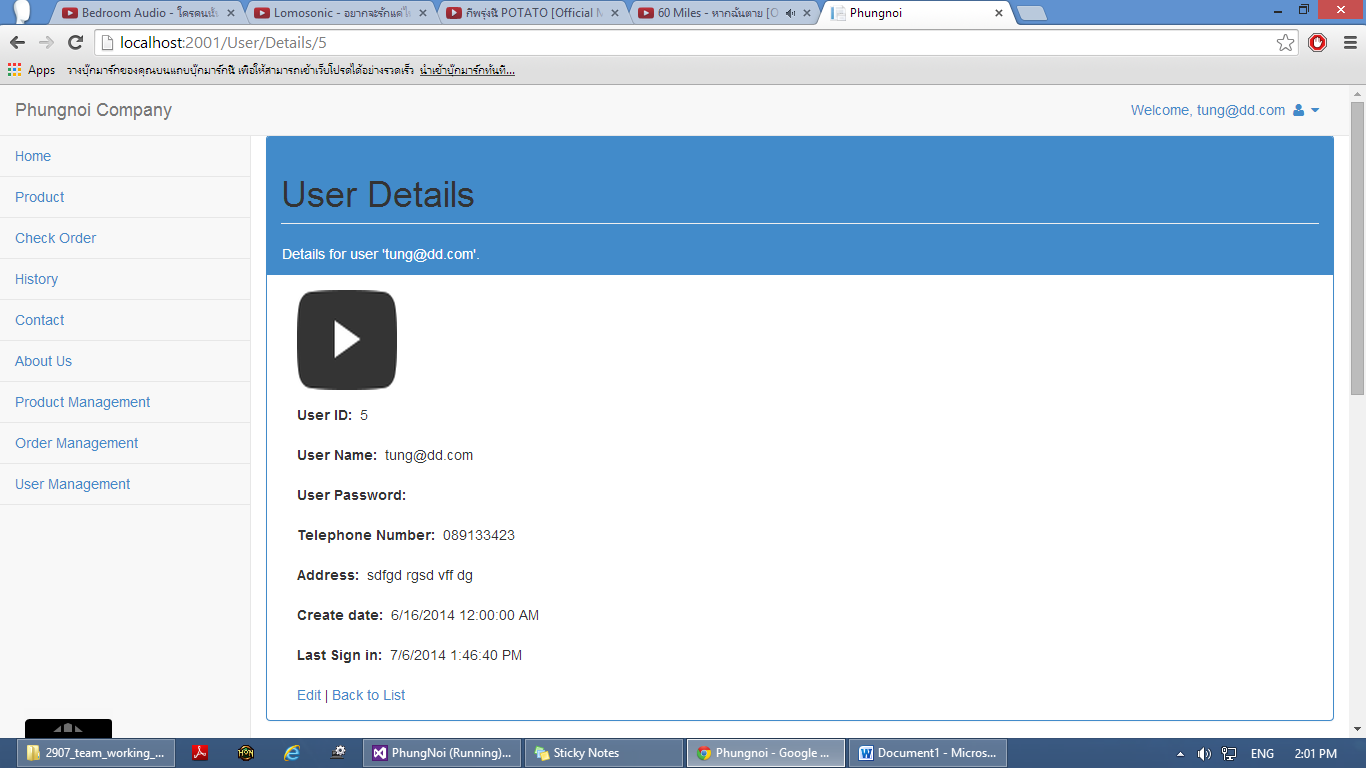


**Figure 37:** UI – 06 Edit member information page

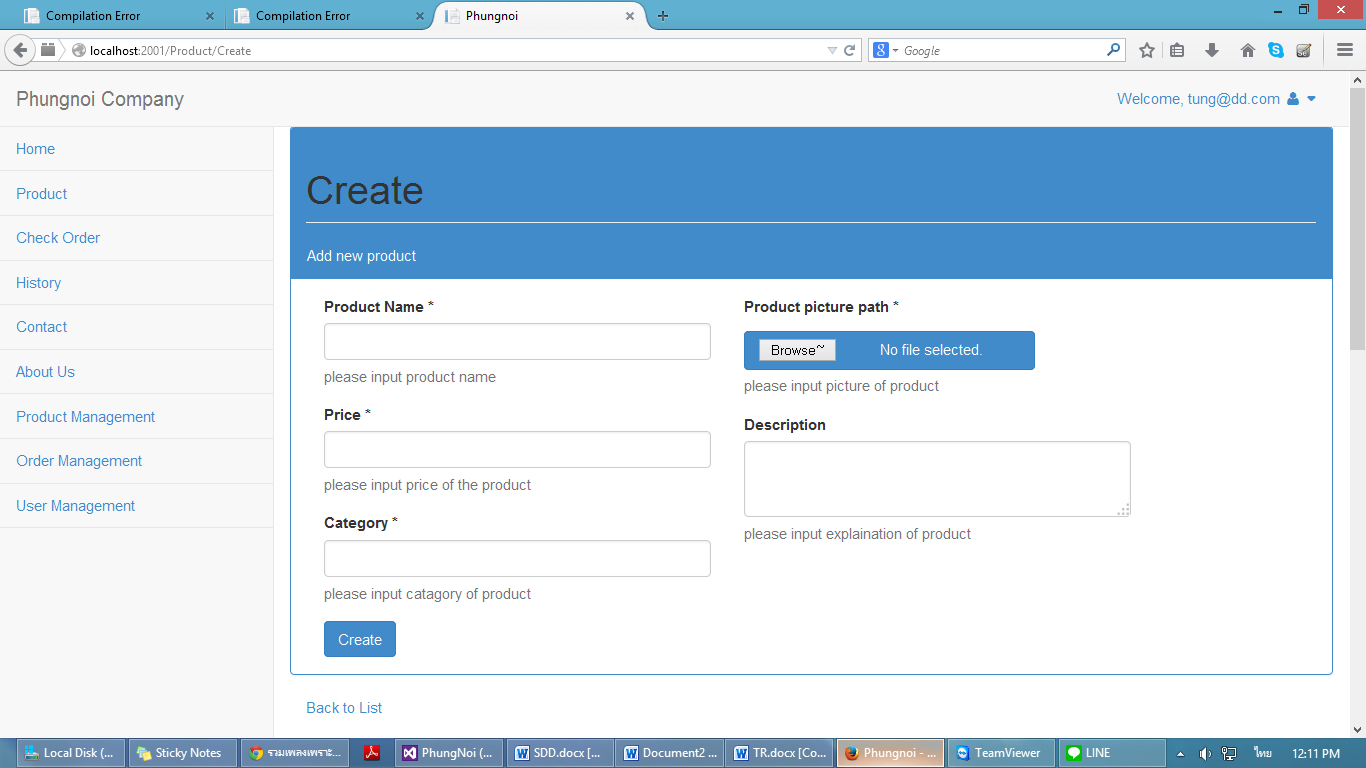


**Figure 38:** UI – 07 Delete member information page

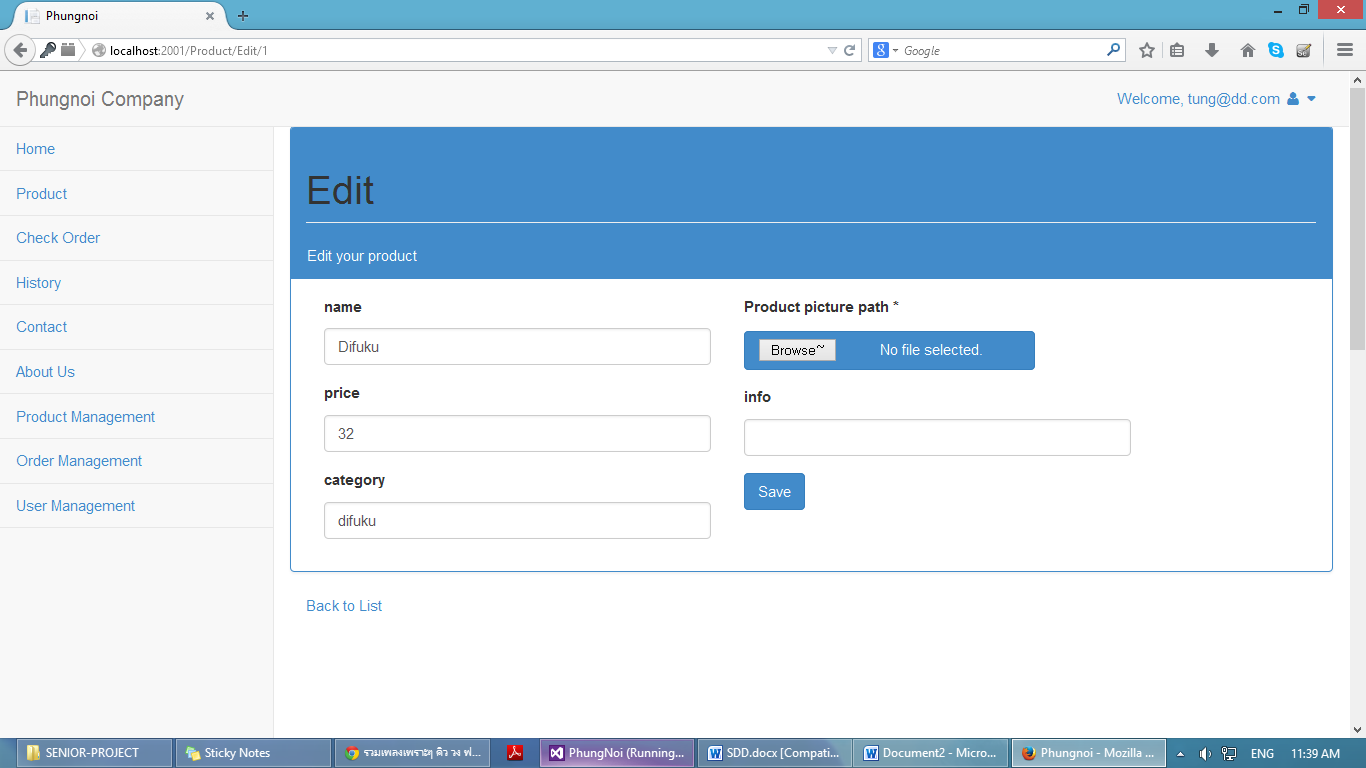
**Figure 39:** UI – 08 Member list page

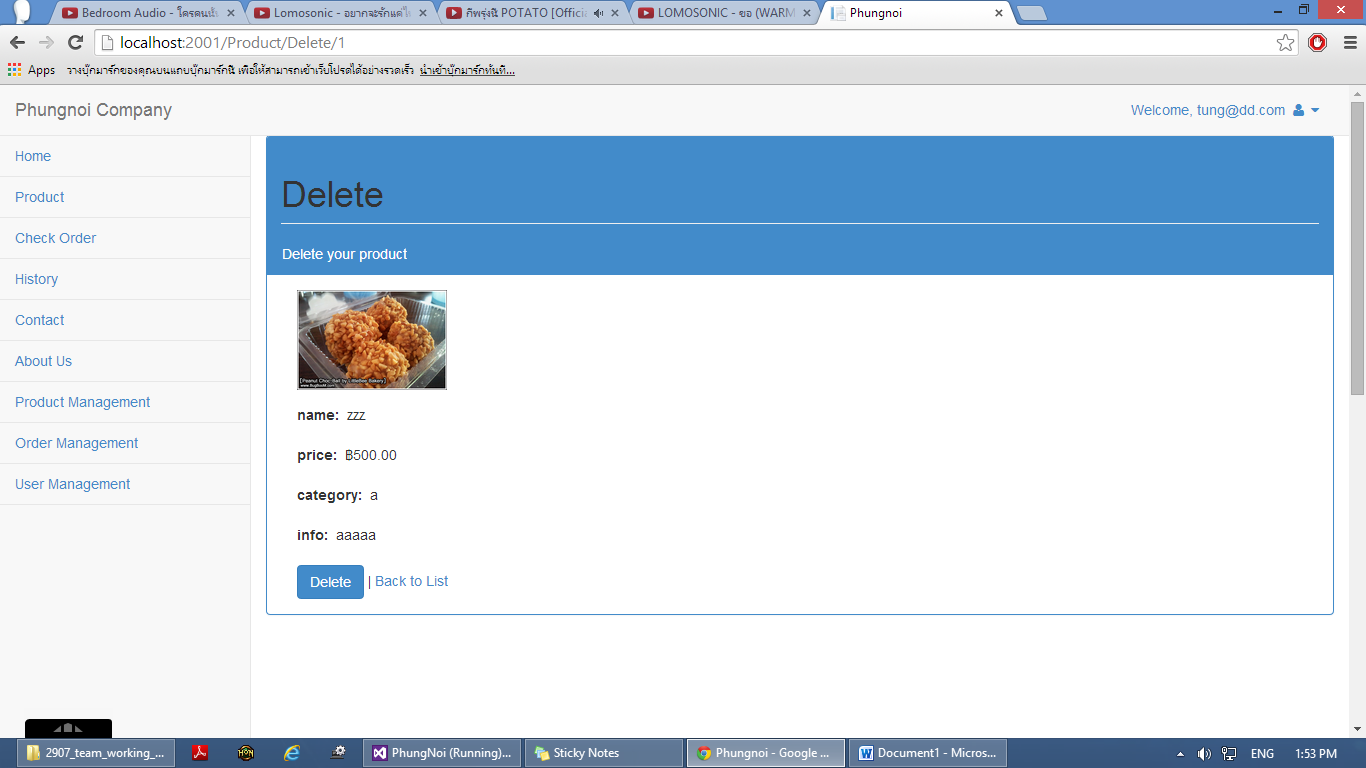


**Figure 40:** UI – 09 Member details page

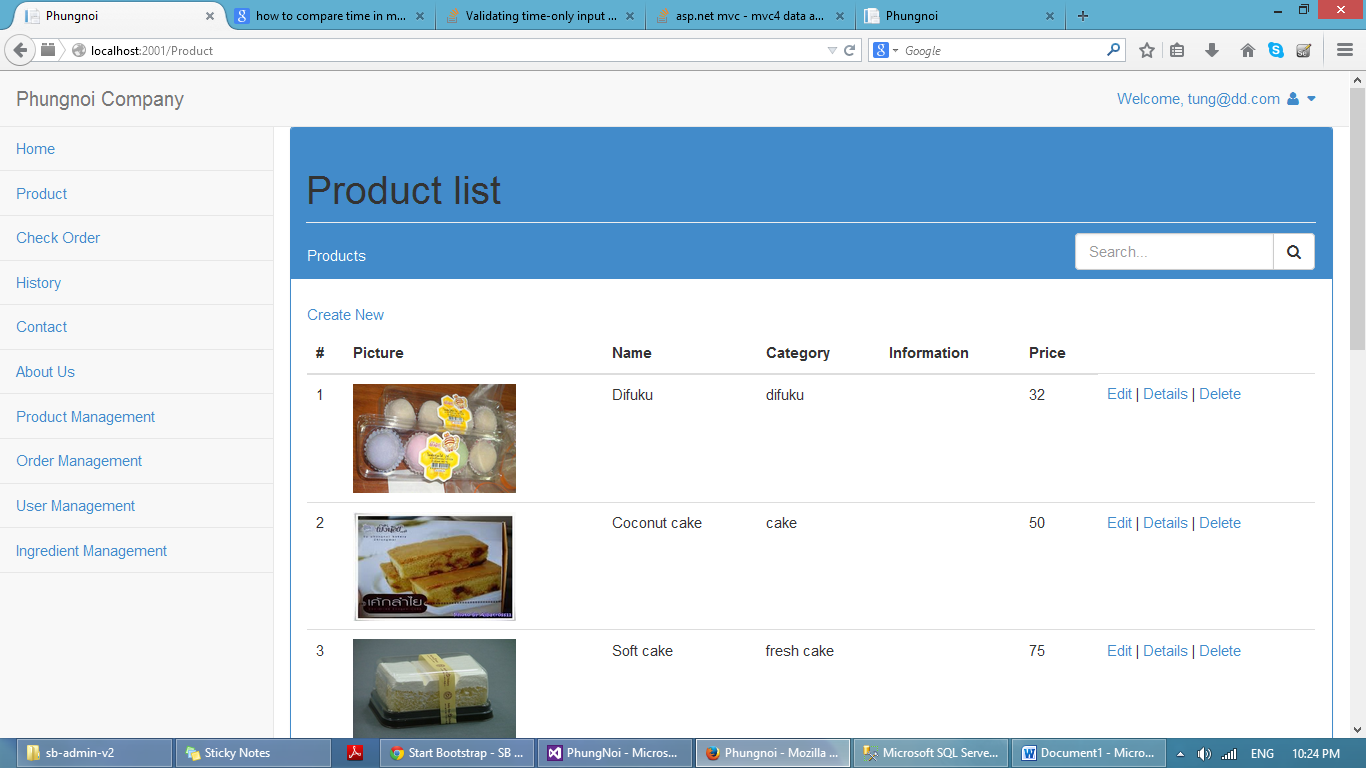


**Figure 41:** UI – 10 Create bakery product page

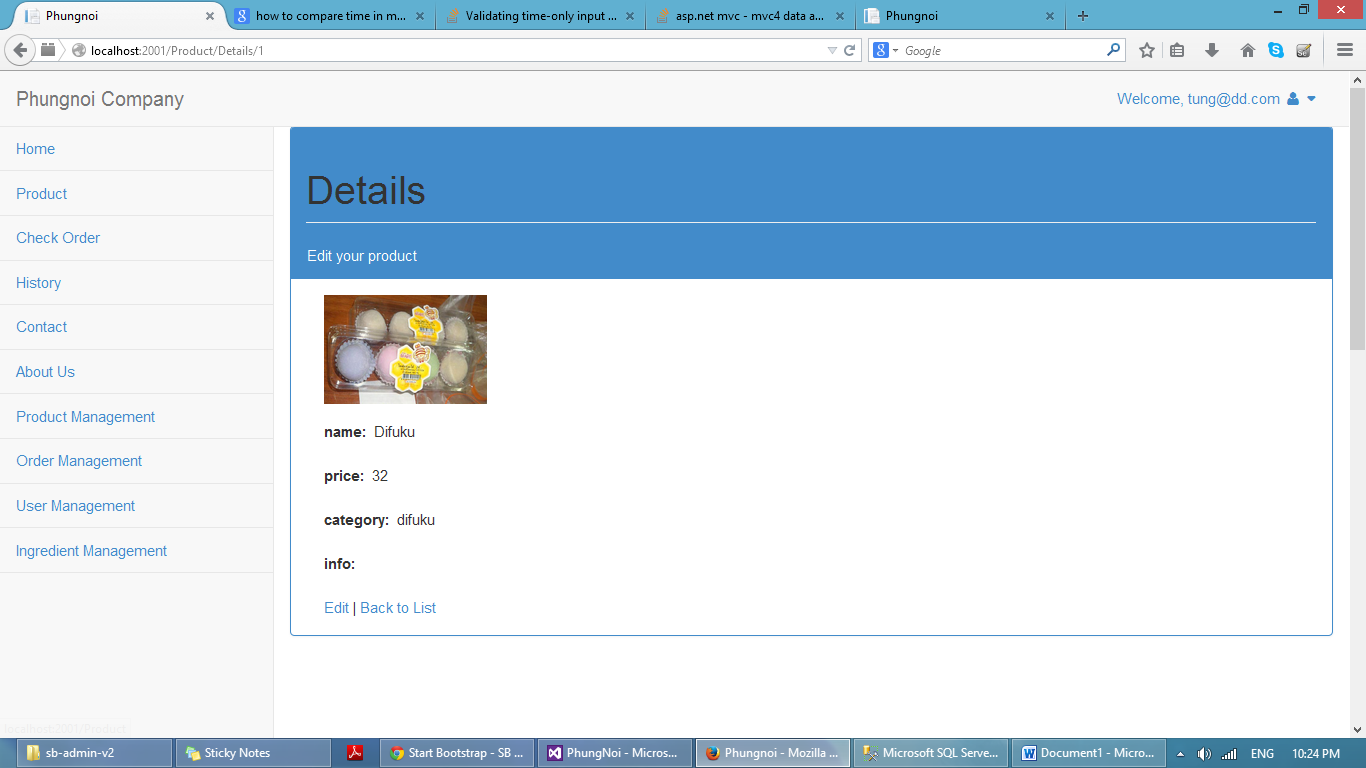
****

**Figure 42:** UI – 11 Edit bakery product page

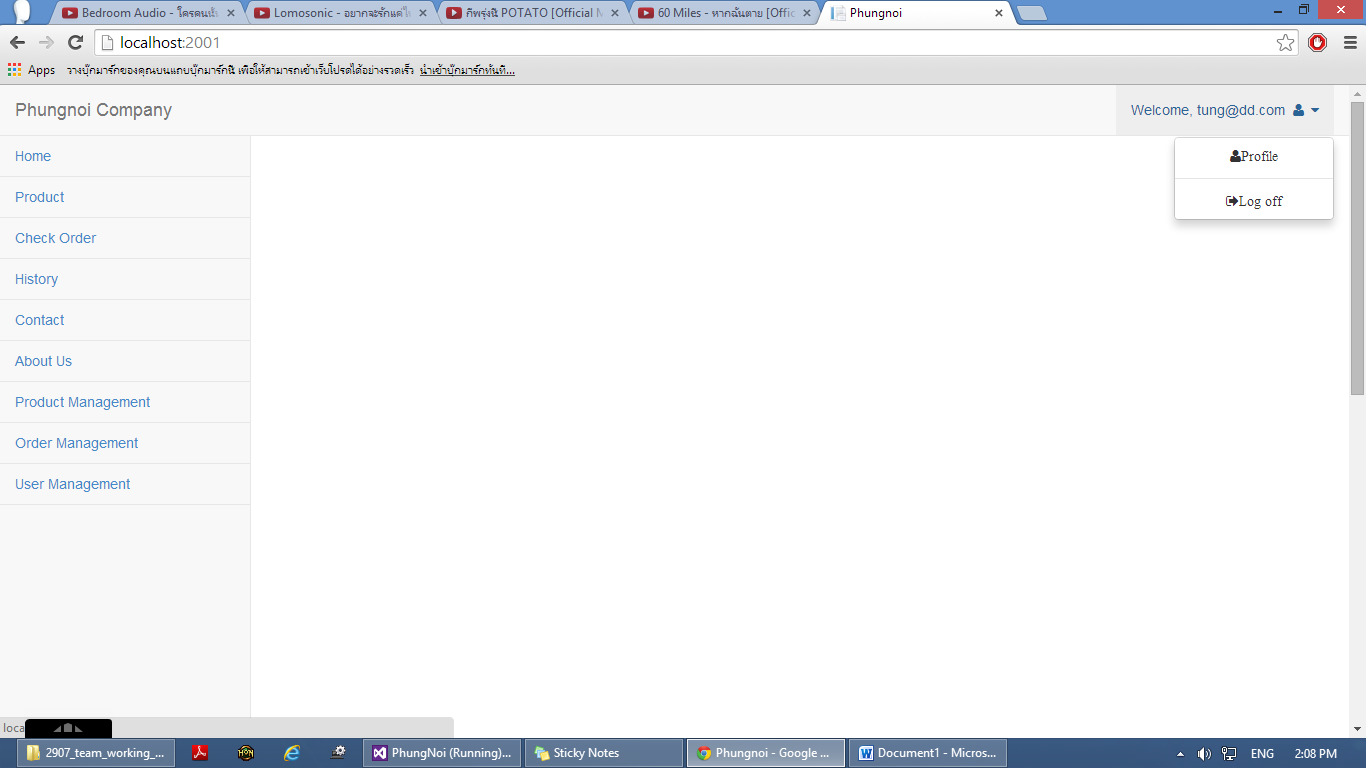
**Figure 43:** UI – 12 Delete bakery product page



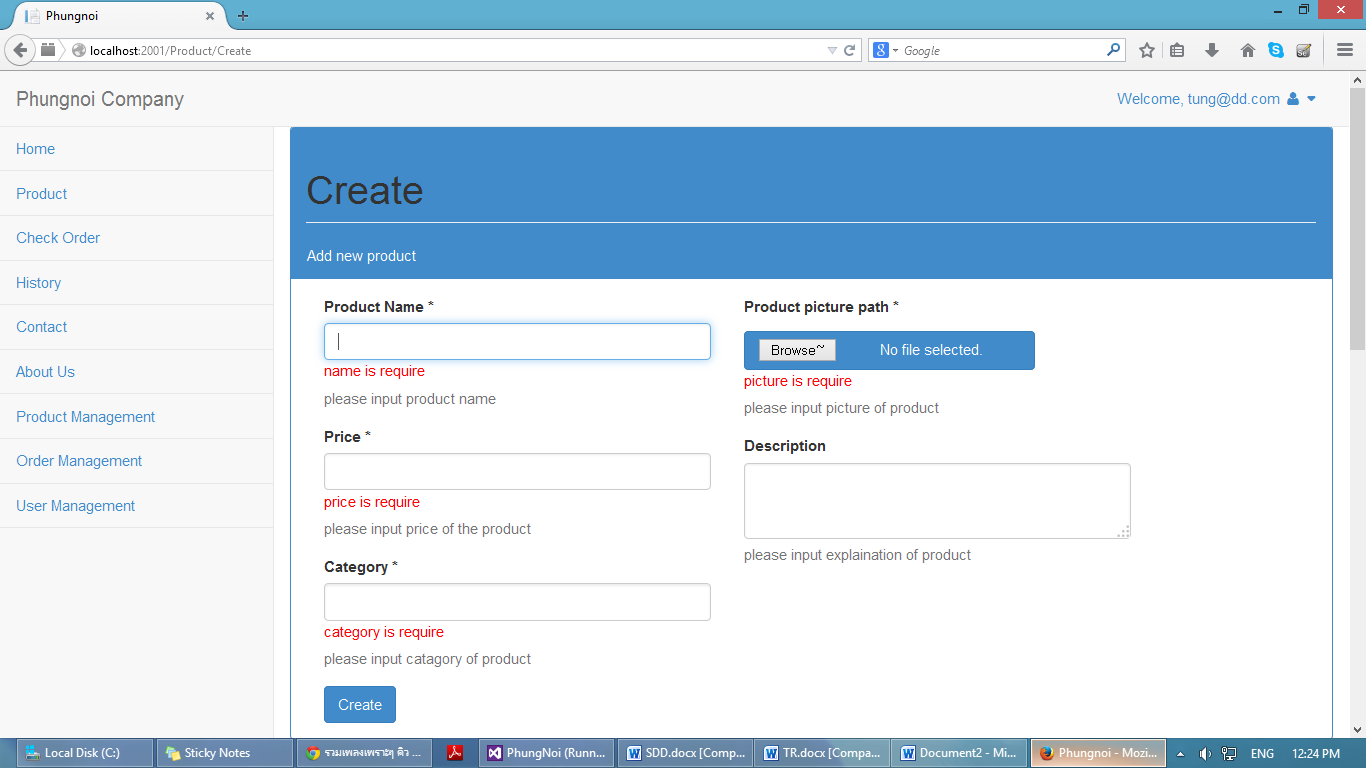
**Figure 44:** UI – 13 Bakery product list page



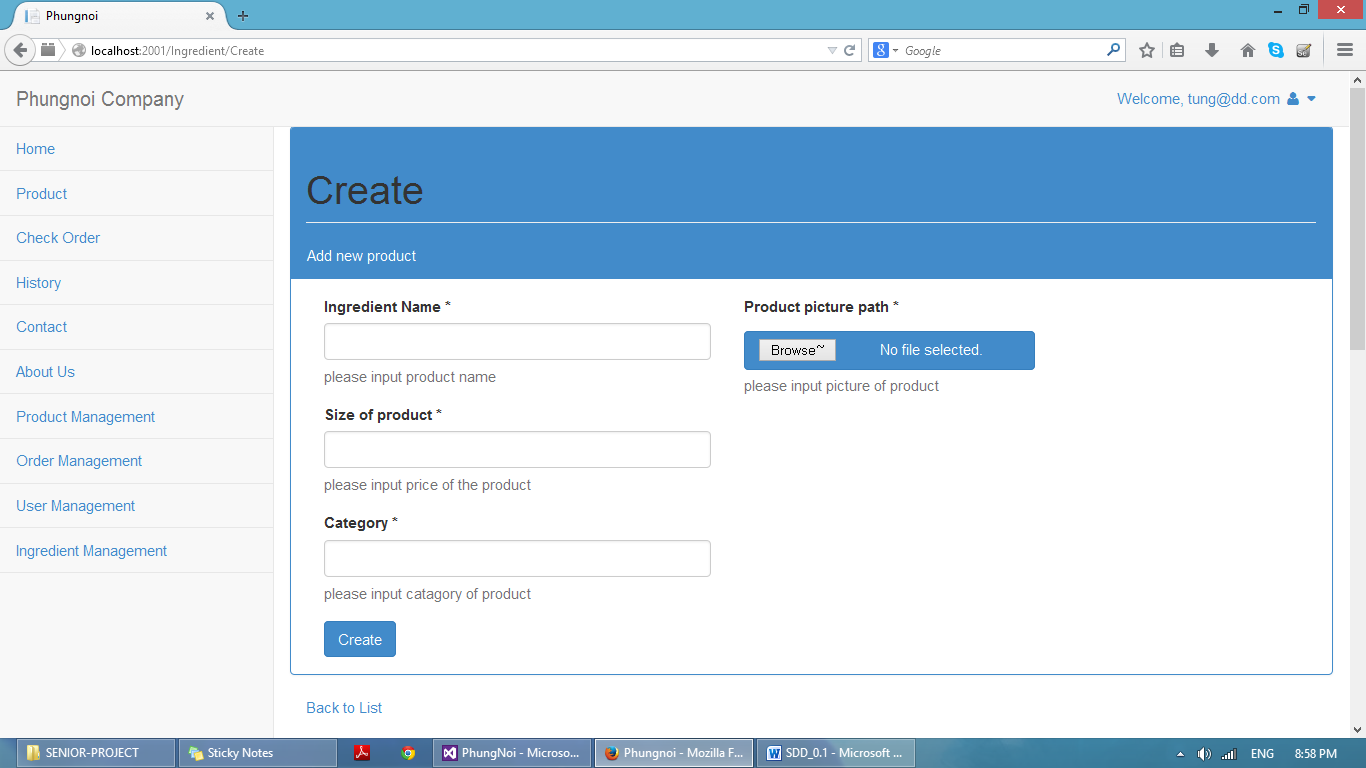
**Figure 45:** UI – 14 Bakery product details page



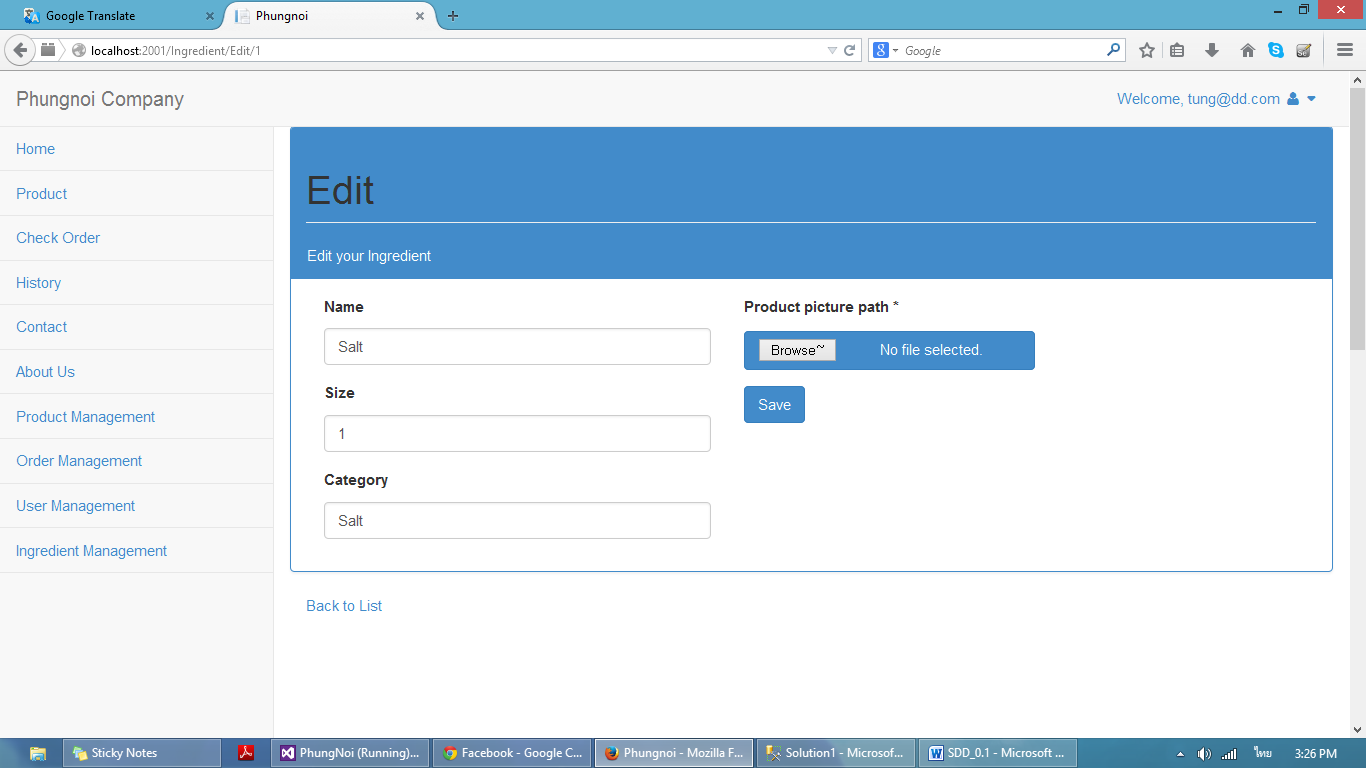
**Figure 46:** UI – 15 Logout page



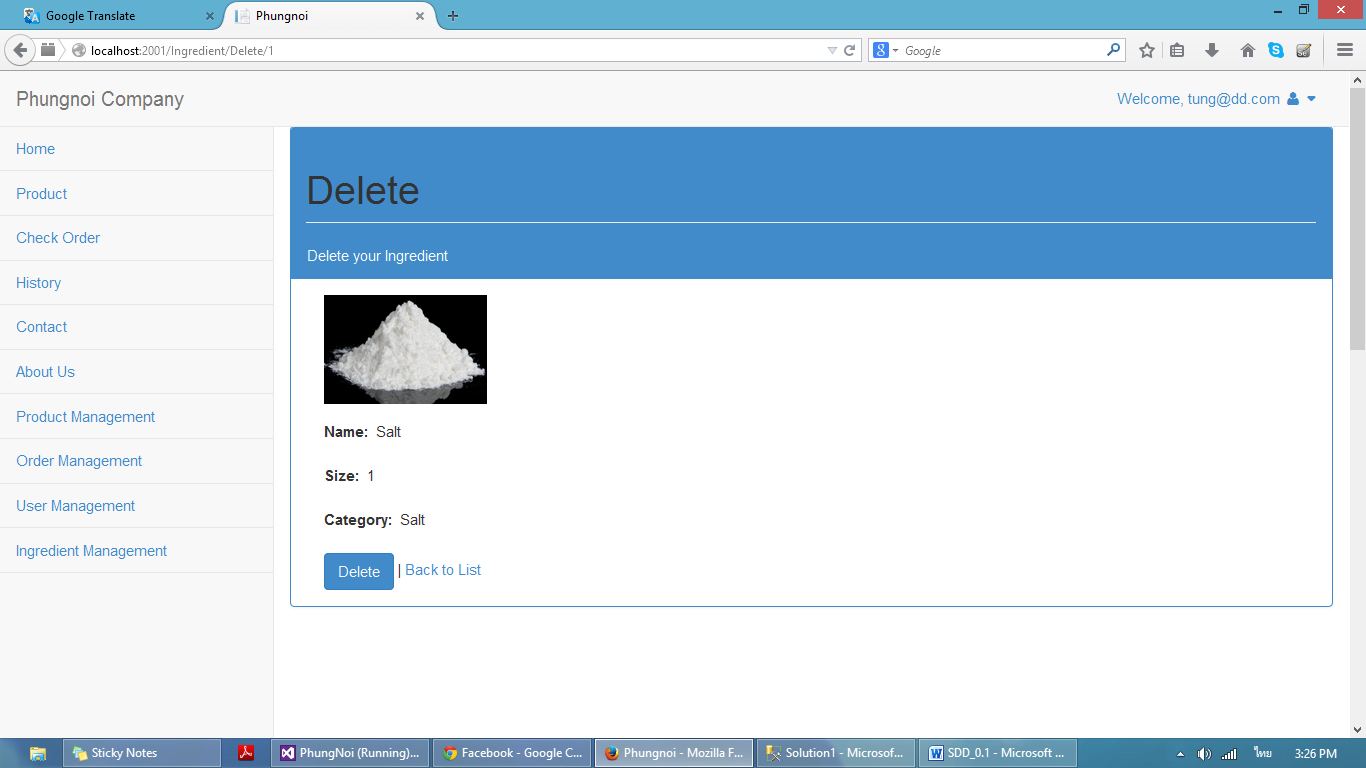
**Figure 47:** UI – 16 Validation form page



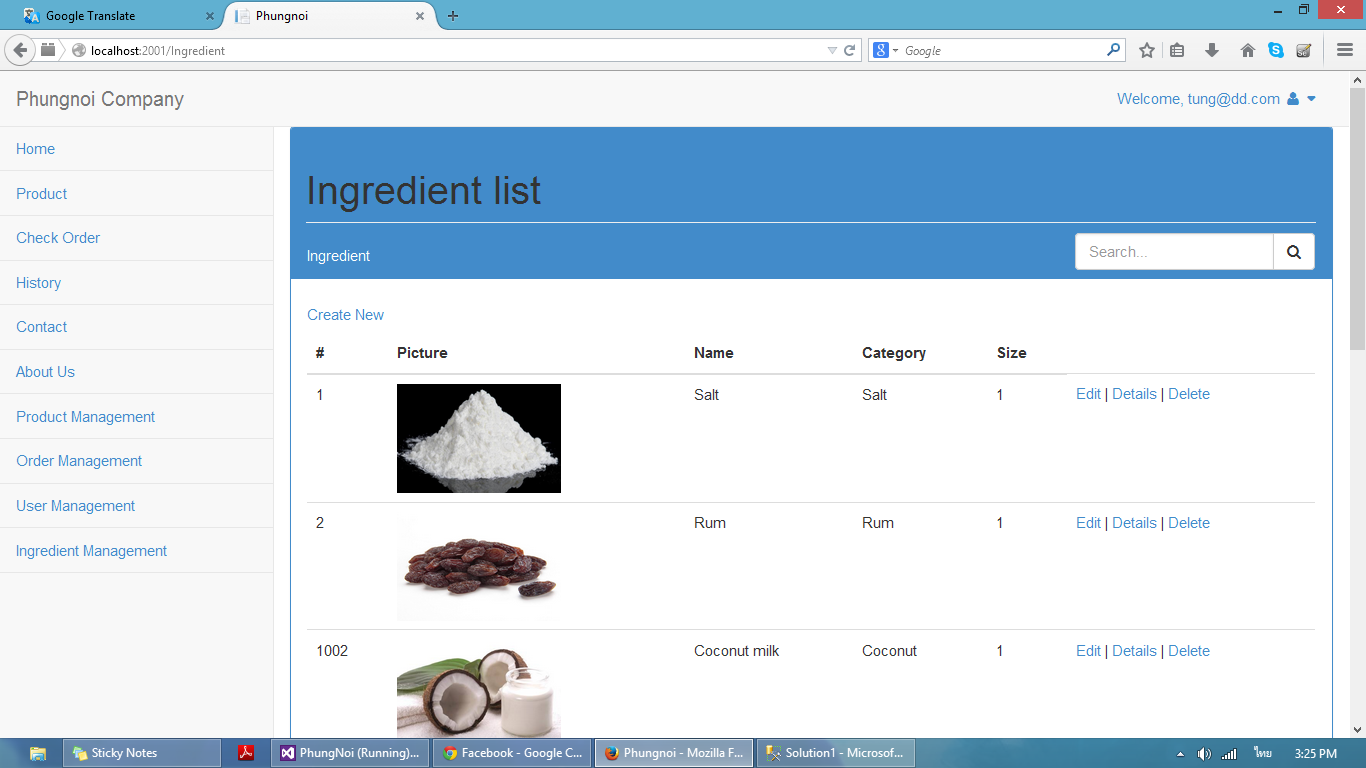
**Figure 48:** UI – 17 Create ingredient information page



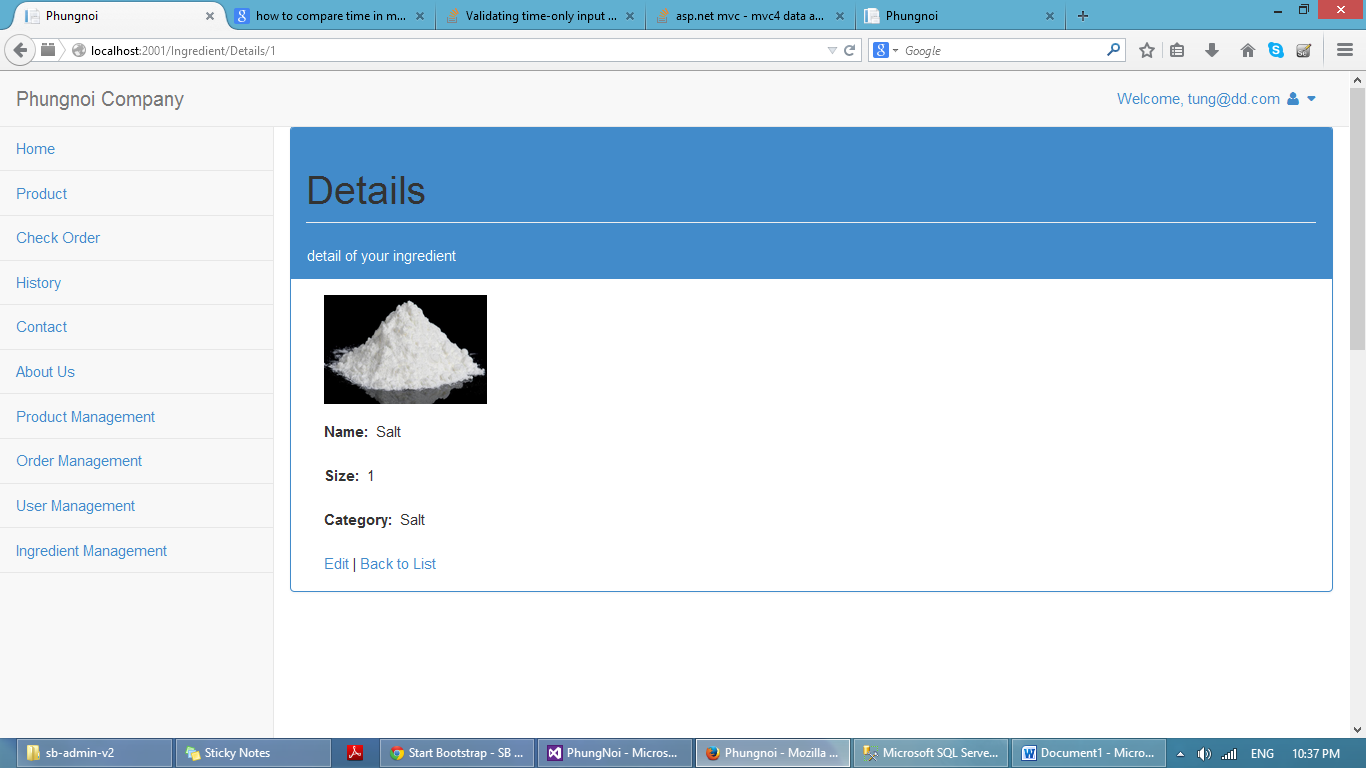
**Figure 49:** UI – 18 Edit ingredient information page



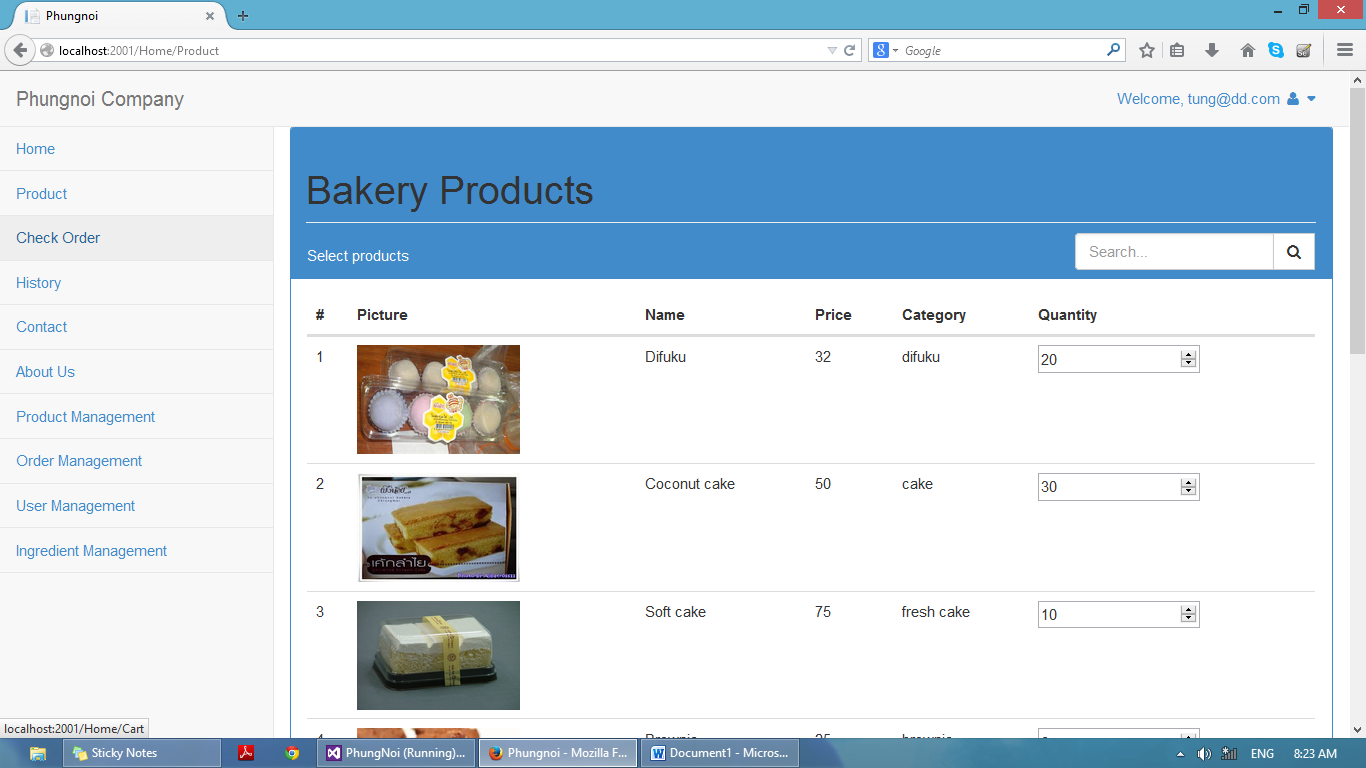
**Figure 50:** UI – 19 Delete ingredient page



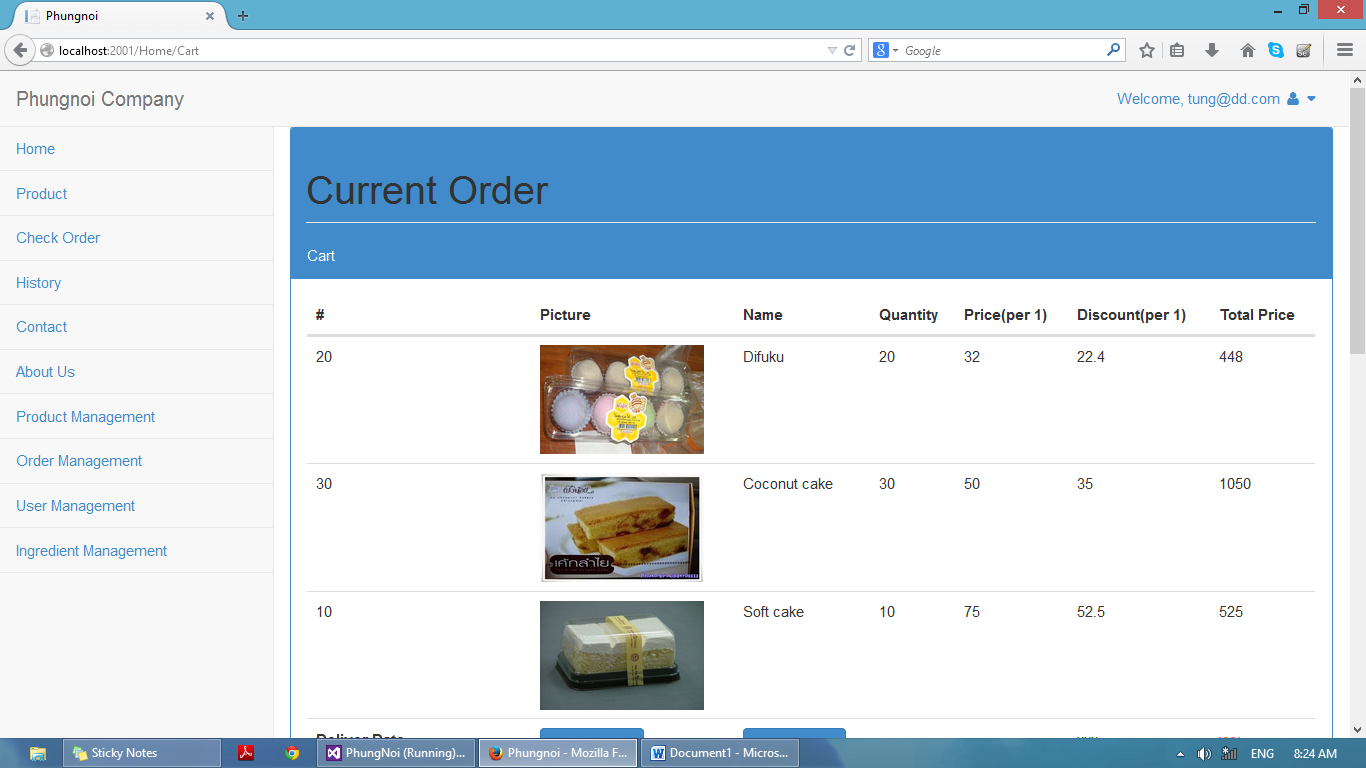
**Figure 51:** UI – 20 Ingredient list page



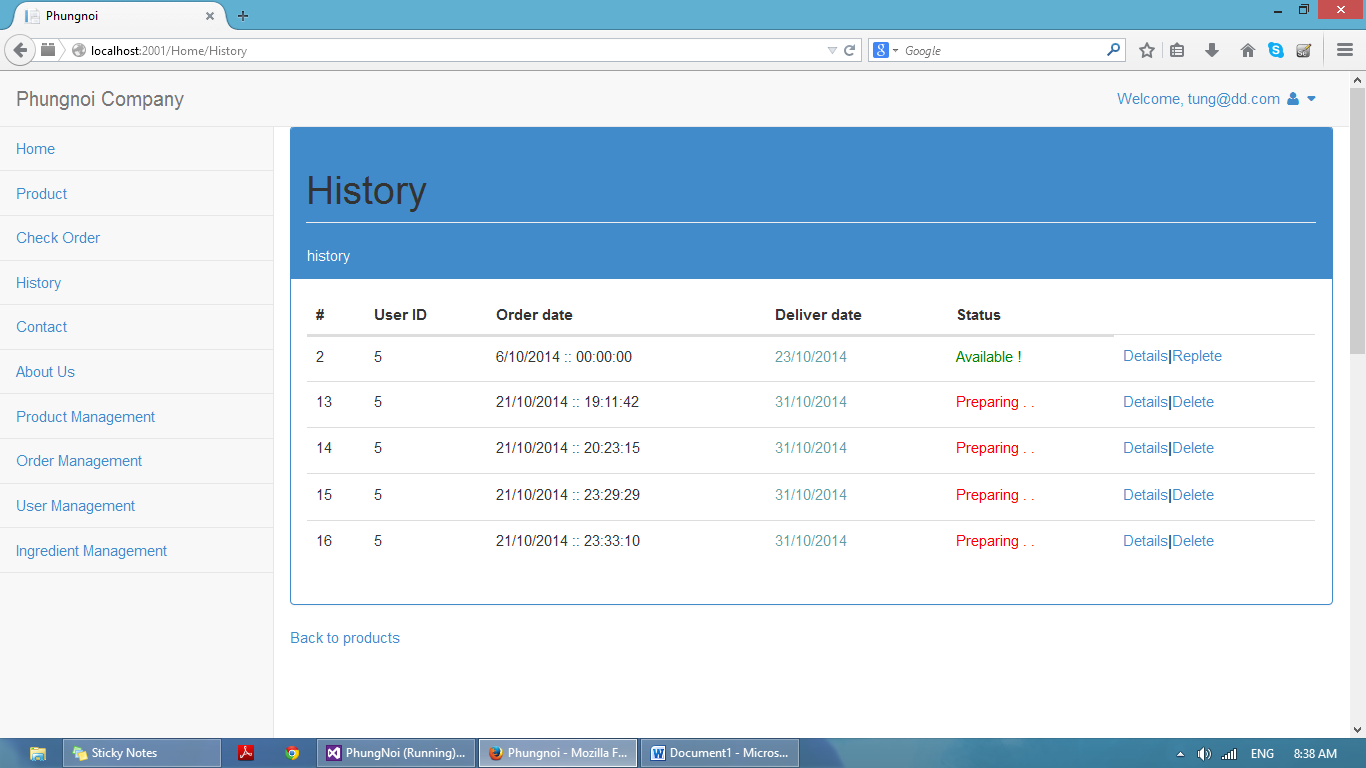
**Figure 52:** UI – 21 Ingredient Details page



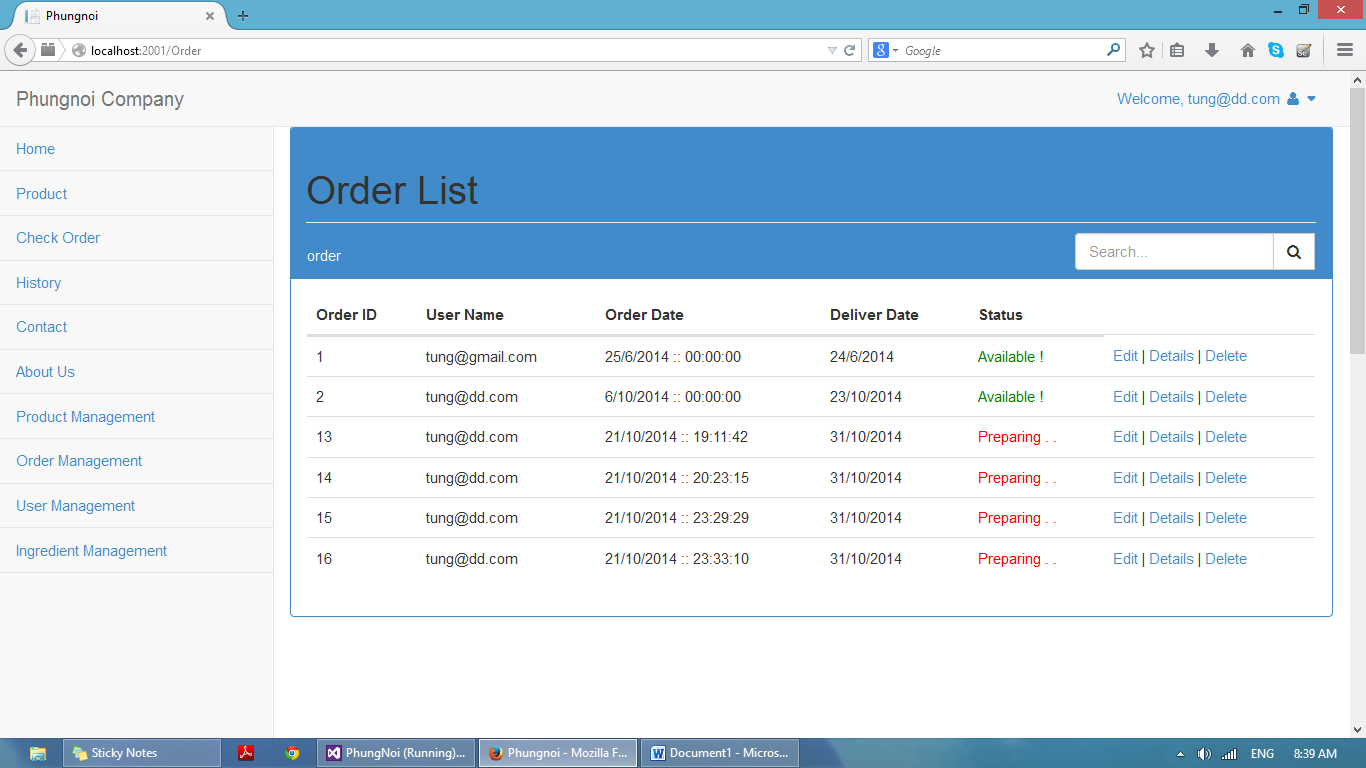
**Figure 53:** UI – 22 Order page



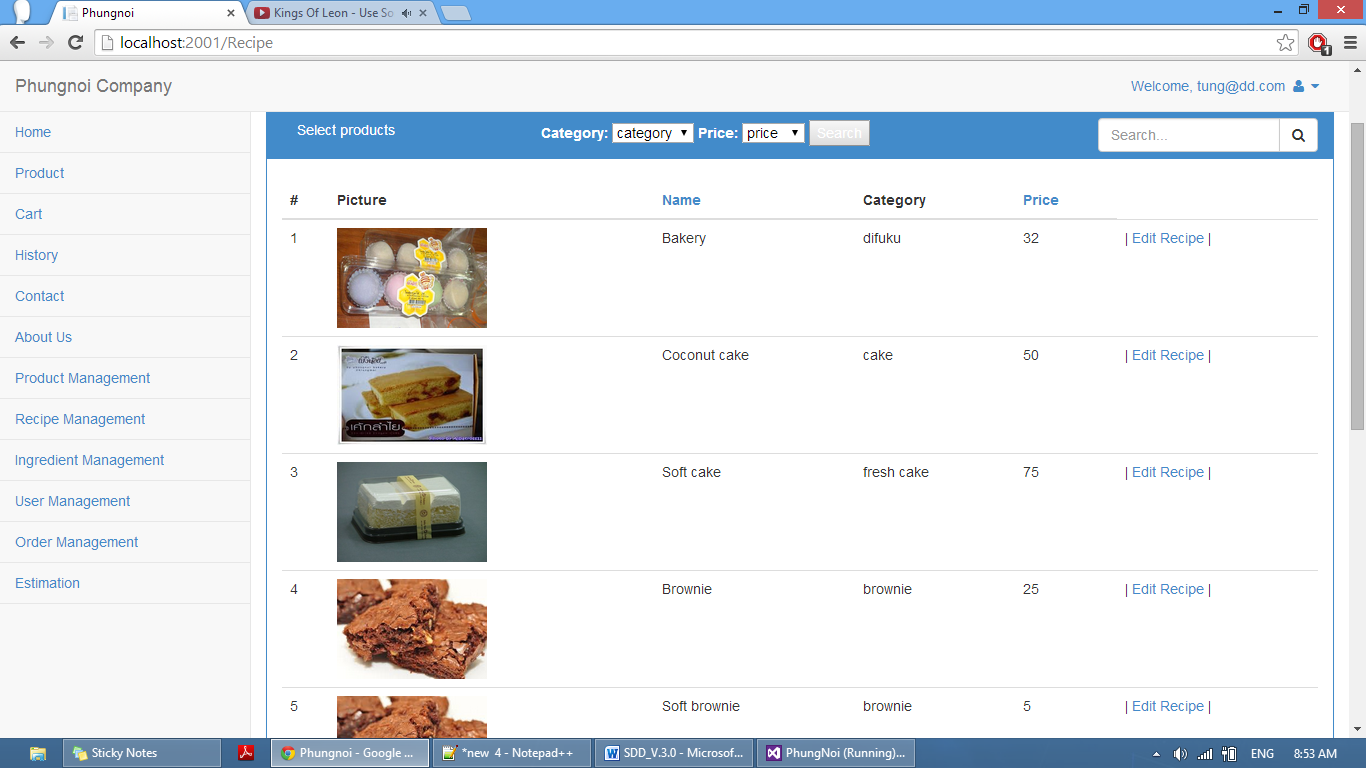
**Figure 54:** UI – 23 View current order page



**Figure 55:** UI – 24 Ordered history page



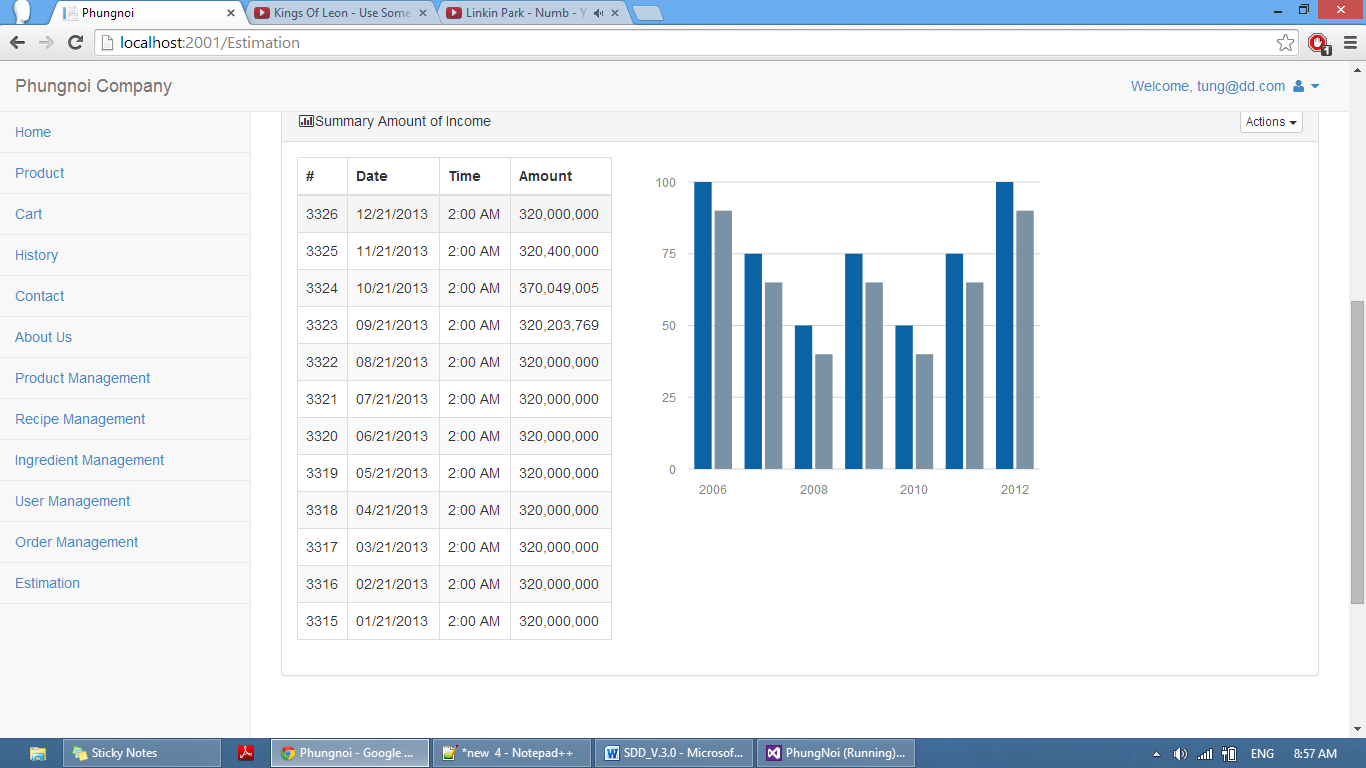
**Figure 56:** UI – 25 Ordered list page



**Figure 57:** UI – 26 menu recipe page



**Figure 58:** UI – 27 Report page



**Figure 59:** UI – 28 Graph page