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

Test Plan

By

**Miss. Nontra Mahachanont 542115026**

**Mr. Parinya Panyanak 542115034**

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

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

**Mr.Phudinan Singkhamfu**

**Document History**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Document Name** | **Version** | **Status** | **Date** | **Viewable** | **Reviewer** | **Responsible** |
| **Documents** | | | | | | |
| Web-based Ordering & Ingredient Estimating for Bakery Manufacturer – Test Plan\_V.1.1.docx | V.1.1  - Add Chapter 1  - Add Chapter 2  - Add Chapter 4 | Drafted | 8-10-2014 | NM, PP, PS | NM, PP | NM, PP |
| Web-based Ordering & Ingredient Estimating for Bakery Manufacturer – Test Plan\_V.1.4.docx | V.1.4  - Add Chapter 3  - Adjust Chapter 4 | Drafted | 10-10-2014 | NM, PP, PS | NM, PP | NM, PP |
| Web-based Ordering & Ingredient Estimating for Bakery Manufacturer – Test Plan\_V.1.6.docx | V.1.6  - Adjust Chapter 3  - Adjust Chapter 4  - Fix inappropriate word | Drafted | 11-10-2014 | NM, PP, PS | NM, PP | NM, PP |
| Web-based Ordering & Ingredient Estimating for Bakery Manufacturer – Test Plan\_V.2.0.docx | V.2.0  - Adjust Chapter 4 | Released | 22-10-2014 | NM, PP, PS | NM, PP | NM, PP |
| Web-based Ordering & Ingredient Estimating for Bakery Manufacturer – Test Plan\_V.2.5.docx | V.2.5  - Adjust Chapter 3  - Adjust Chapter 4 | Reviewed | 29-11-2014 | NM, PP, PS | NM, PP | NM, PP |
| Web-based Ordering & Ingredient Estimating for Bakery Manufacturer – Test Plan\_V.3.0.docx | V.3.0  - Adjust Chapter 4  - Fix inappropriate word | Released | 16-11-2015 | NM, PP, PS | NM, PP | NM, PP |

**\*NM = Nontra Mahachanont**

**\*PP = Parinya Panyanak**

**\* PS = Phudinan Singkhamfu**

**Table of contents**

[**Chapter One | Introduction** 4](#_Toc409568991)

[1.1 Objective 4](#_Toc409568992)

[1.2 Scope 4](#_Toc409568993)

[1.3 Acronyms and Definitions 4](#_Toc409568994)

[1.3.1 Acronyms 4](#_Toc409568995)

[1.3.2 Definitions 4](#_Toc409568996)

[**Chapter Two| Test Plan and Test Procedure** 6](#_Toc409568997)

[2.1 Test Objectives 6](#_Toc409568998)

[2.2 Scope of Testing 6](#_Toc409568999)

[2.3 Test Duration 6](#_Toc409569000)

[2.4 Test Responsibility 6](#_Toc409569001)

[2.5 Result of Testing 7](#_Toc409569002)

[2.6 Test Environent 7](#_Toc409569003)

[2.6.1 Hardware 7](#_Toc409569004)

[**Chapter Three | Unit Testing of Web-based Ordering & Ingredient Estimating for Bakery Manufacturer** 8](#_Toc409569005)

[**Chapter Four | System Testing of Web-based Ordering & Ingredient Estimating for Bakery Manufacturer** 24](#_Toc409569006)

# **Chapter One | Introduction**

## 1.1 Objective

The objectives of the test plan of Web-based Ordering & Ingredient Estimating for Bakery Manufacturer are used to create test plan of the unit testing. It makes sure that the defects are discovered and fixed all case in all project progress report. System testing covers the user requirements system. The unit testing covers all of implemented methods in Web-based Ordering & Ingredient Estimating for Bakery Manufacturer project.

## 1.2 Scope

The test plan describes the unit testing activities to find out the defect in the system and describes the system testing activities for testing an entirely integrated system to check that it covers all of the user’s requirements.

## 1.3 Acronyms and Definitions

### 1.3.1 Acronyms

SRS Software Requirement Specification

URS User Requirement Specification

SDD Software Design Document

UI User Interface

UTC Unit Test Case

STC System Test Case

### 1.3.2 Definitions

Feature Transformation of input parameters to output parameters based on a specified algorithm. It describes the functionality of the product in the language of the product. Used for requirements analysis, design, coding, testing or maintenance. [IEEE90]

IEEE Institute for Electrical and Electronics Engineers. Biggest global interest group for engineers of different branches and computer scientists. [IEEE90]

Requirement (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. (3) A documented representation of a condition or capability as in the definition (1) or (2). [IEEE90]

Specification Precise description of an activity or work product that reserves as basis or input for further activities or work product. A specification can comprise requirements to a product and how they will solve. Different parts of a specification (e.g., what is to bed one, how it will done) must not mixed. [IEEE90]

White box testing A software testing method in which the internal structure/design/implementation of the item being tested known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

Unit testing A level of the software testing process where individual units/components of a software/system tested. The purpose is to validate that each unit of the software performs as designed.

System testing A level of the software testing process where a complete, integrated system/software tested. The purpose of this test is to evaluate the system’s compliance with specified requirements.

Null Define values represent missing unknown data.

# **Chapter Two| Test Plan and Test Procedure**

## 2.1 Test Objectives

The objectives of testing Web-based Ordering & Ingredient Estimating for Bakery Manufacturer project are:

1. The defects are detected.

2. The defects are fixed.

3. Functionalities and user interface covered all requirements that a user want.

4. All functionalities and features that are define in all project progress report, must be follow in the project plan.

## 2.2 Scope of Testing

Web-based Ordering & Ingredient Estimating for Bakery Manufacturer will test by press button testing techniques that are unit testing in web application side.

## 2.3 Test Duration

|  |  |
| --- | --- |
| **Progress** | **Date and Duration** |
| **Progress I** | **Start:** 1 July 2014  **End:** 29 July 2014  **Duration:** 21 Days |
| **Progress II** | **Start:** 19 October 2014  **End:** 22 October 2014  **Duration:** 4 Days |
| **Final Progress** | **Start:** 19 December 2014  **End:** 15 January 2015  **Duration:** 20 Days |

## 2.4 Test Responsibility

|  |  |
| --- | --- |
| **Item** | **Responsibility** |
| **Unit Test** | Parinya Panyanak |
| **System Test** | Parinya Panyanak |

## 2.5 Result of Testing

1. Actual Output: The actual output that are performed by each test case.

2. Pass/Fail criteria:

2.1 Pass: The result of an actual result is same with expected result.

2.2 Fail: The result of an actual result is not same with expected result.

## 2.6 Test Environent

### 2.6.1 Hardware

* Computers

**- Name:** Sony VAIO-VAIO

**Processor:** Intel® Pentium® Processor T4300 (2.10 GHz)

**Memory:** 2.00 GB

**Graphics:** Intel® Graphics Media Accelerator HD

**Operating System:** Window® 7 Home Basic (64-bit)

**- Name:** Sony VAIO

**Processor:** Intel® Core (TM) I3-3120M (2.5 GHz)

**Memory:** 4.00 GB

**Graphics:** Radeon Graphics System

**Operating System:** Window® 8 Based Processor (64-bit)

**2.6.2 Software**

* Microsoft SQL Server Management Studio
* Microsoft Visual Studio Ultimate
* Selenium

# **Chapter Three | Unit Testing of Web-based Ordering & Ingredient Estimating for Bakery Manufacturer**

**Class:** AccountController

**Unit Test Case-01 (UTC-01):** public ActionResultLogin (string returnUrl)

**Description:** This unit test case used to check Login (string returnUrl) method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to the login page of AccountController | {ReturnURrl == null} | return View ("Login", "\_PlainLayout"); |

**Class:** AccountController

**Unit Test Case-02 (UTC-02):** public ActionResultLogin (LoginModel model, string returnUrl)

**Description:** This unit test case used to check procedure of Login () method that can work correctly.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Login with valid username and password | { Username == “[tung@dd.com](mailto:tung@dd.com)”, Password == “123456”,  RememberMe == false},  {returnUrl == null} | RedirectToLocal (null); |
| 2 | Login with invalid username and password | { Username == “[tung@dd.com](mailto:tung@dd.com)”, Password == “555555”,  RememberMe == false},  {returnUrl == null} | return View ("Login", "\_PlainLayout",  { Username == “[tung@dd.com](mailto:tung@dd.com)”, Password == “555555”,  RememberMe == false}); |

**Class:** UserController

**Unit Test Case-03 (UTC-03):** public ActionResultIndex ()

**Description:** This unit test case used to check Index () method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to Index page of a member | - | return View ("Index", List<Member>); |

**Class:** UserController

**Unit Test Case-04 (UTC-04):** public ActionResultDetails (int id)

**Description:** This unit test case used to check Details (int id) method that can return to the correct page and member.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | User is not null | {id == 2} | Return View ("Details",  {Username == “[tung@gmail.com](mailto:tung@gmail.com)”, Password == “123456”, firstName == “Parinya” ,  lastName == “Panyanak”,  phoneNumber == “0863338313”,  address == “xxxxxx”}); |
| 2 | User is null | {id == 500} | return HttpNotFound (); |

**Class:** UserController

**Unit Test Case-05 (UTC-05):** public ActionResultUserProfile (string id)

**Description:** This unit test case used to check UserProfile (string id) method that can return to the correct page and member.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | User is not null | {username == “tung@dd.com”} | return View ("UserProfile",  {Username == “[tung@dd.com](mailto:tung@dd.com)”, Password == “123456”, firstName == “tung”,  lastName == “lastname”, phoneNumber == “089133423”, address == “xxxxxx”});); |
| 2 | User is null | {username == “nook@dd.com”} | return HttpNotFound (); |

**Class:** UserController

**Unit Test Case-06 (UTC-06):** public ActionResultCreate ()

**Description:** This unit test case used to check Create () method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to the created page of a member | - | return View ("Create"); |

**Class:** UserController

**Unit Test Case-07 (UTC-07):** public ActionResultCreate (Member member)

**Description:** This unit test case used to check Create (Member member) method that can create new member to the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {Username == [nontra@gmail.com](mailto:nontra@gmail.com) ,Password == nontra789,  firstName == Nontra,  lastName == Mahachanont,  phoneNumber == 0875885737,  address == 42/1 m.6 Muang Chaingmai} | return RedirectToAction ("Details", { Username == [nontra@gmail.com](mailto:nontra@gmail.com) , Password == nontra789,  firstName == Nontra,  lastName == Mahachanont,  phoneNumber == 0875885737,  address == 42/1 m.6 Muang Chaingmai}); |
| 2 | Test with input incorrect information | { Username == NestNontra,  Password == nontra789,  FirstName == Nontra,  LastName == Mahachanont,  PhoneNumber == 0875885737,  Address == 42/1 m.6 Muang Chaingmai} | return View (); |

**Class:** UserController

**Unit Test Case-08 (UTC-08):** public ActionResultEdit (int id = 0)

**Description:** This unit test case used to check Edit (int id = 0) method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | User is not null | {id == 2} | return View ("Edit",  {Username == “[tung@gmail.com](mailto:tung@gmail.com)”, Password == “123456”,  firstName == “Parinya”,  lastName == “Panyanak”,  phoneNumber == “0863338313”,  address == “xxxxxx”});); |
| 2 | User is null | {id == 500} | return HttpNotFound (); |

**Class:** UserController

**Unit Test Case-09 (UTC-09):** public ActionResultEdit (Member member)

**Description:** This unit test case used to check Edit (Member member) method that can update information to the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {Username == “[tortung@gmail.com](mailto:tortung@gmail.com)”,  Password == “123456”,  firstName == “Parinya” ,  lastName == “Panyanak”,  phoneNumber == “0863338313”,  address == “xxxxxx”}); | return View ("Index") |
| 2 | Test with input incorrect information | {Username == “[tortung](mailto:tung@gmail.com)”,  Password == “123456”,  firstName == “Parinya”,  lastName == “Panyanak”,  phoneNumber == “0863338313”,  address == “xxxxxx”}); | return View ({Username == “[tortung](mailto:tung@gmail.com)”,  Password == “123456”,  firstName == “Parinya”,  lastName == “Panyanak”,  phoneNumber == “0863338313”,  address == “xxxxxx”});); |

**Class:** UserController

**Unit Test Case-10 (UTC-10):** public ActionResultDelete (int id = 0)

**Description:** This unit test case used to check Delete (int id = 0) method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return delete page of a member that is selected | {id == 2} | return View ("Delete",  {Username == “[tung@gmail.com](mailto:tung@gmail.com)”, Password == “123456”,  firstName == “Parinya” ,  lastName == “Panyanak”, phoneNumber == “0863338313”, address == “xxxxxx”}); |
| 2 | Return delete page of a member that not selected | {id == 500} | return HttpNotFound (); |

**Class:** UserController

**Unit Test Case-11 (UTC-11):** public ActionResultDeleteConfirmed (int id)

**Description:** This unit test case used to check procedure of DeleteConfirmed (int id) method that can delete a member from the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Delete with valid member identification | {id == 2} | return View ("Index"); |
| 2 | Delete with invalid member identification | {id == 500} | return View (user); |

**Class:** ProductController

**Unit Test Case-12 (UTC-12):** public ActionResultIndex()

**Description:** This unit test case used to check Index () method that can go to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to the index page of Product | - | Return View ("Index", List<Product>) |

**Class:** ProductController

**Unit Test Case-13 (UTC-13):** public ActionResultDetails (int id = 0)

**Description:** This unit test case used to check Details (int id=0) method that can go to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to the product information page and found valid product identification | {id == 2} | return View ("Details", {Name == “Longgang cake”,  Price == 50, Category == cake,  Picture == \\UploadImages\\2images (4).jpg}); |
| 2 | Return to the product information page and found invalid product identification | {id == 500} | return HttpNotFound (); |

**Class:** ProductController

**Unit Test Case-14 (UTC-14):** public ActionResultCreate ()

**Description:** This unit test case used to check Create () method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **input** | **Expected output** |
| 1 | Return to the created page of product | - | return View ("Create"); |

**Class:** ProductController

**Unit Test Case-15 (UTC-15):** public ActionResultCreate(Product product)

**Description:** This unit test case used to check Create (Product product) method that can create new bakery product information to the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {Name ==“Soft brownie”, Price == 5, Category == brownie ,  Picture ==\UploadImages\5ดาวน์โหลด.jpg } | return RedirectToAction ("Index"); |
| 2 | Test with input incorrect information | {Name == “banana tart”, Price == zxc, Category == “tart”  Picture == \UploadImages\5ดาวน์โหลด.jpg} | return View ({Name == “banana tart”,  Price == zxc, Category == “tart”  Picture == \UploadImages\5ดาวน์โหลด.jpg}); |

**Class:** ProductController

**Unit Test Case-16 (UTC-16):** public ActionResultEdit (int id = 0)

**Description:** This unit test case used to check Edit (int id = 0) method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return Edit page of a product that is selected | {id == 2} | return View ("Edit", ",  {Name == “Longgang cake”,  Price == 50, Category == cake ,  Picture == \\UploadImages\\2images (4).jpg}); |
| 2 | Return Edit page of a product that not selected | {id == 500} | return HttpNotFound(); |

**Class:** ProductController

**Unit Test Case-17 (UTC-17):** public ActionResultEdit (Product product)

**Description:** This unit test case used to check Edit (Product product) method that can update bakery product information to the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {Name == “Coconut cake”,  Price == 50, Category == cake ,  Picture == \\UploadImages\\2images (4).jpg} | return RedirectToAction ("Index", "Product", {Name == “Coconut cake”,  Price == 50, Category == cake ,  Picture == \\UploadImages\\2images (4).jpg}); |
| 2 | Test with input incorrect information | {",{Name == “Coconut cake”,  Price == xyz, Category == cake ,  Picture == \\UploadImages\\2images (4).jpg} | return View ({",{Name == “Coconut cake”, Price == xyz, Category == cake ,  Picture == \\UploadImages\\2images (4).jpg}); |

**Class:** ProductController

**Unit Test Case-18 (UTC-18):** public ActionResultDelete (int id = 0)

**Description:** This unit test case used to check Delete (int id = 0) method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return delete page to the product that is selected | {id == 2} | return View ("Delete",  {Name == “Longgang cake”, Price == 50, Category == cake ,  Picture == \\UploadImages\\2images (4).jpg}); |
| 2 | Return delete page to the product that not selected | {id == 500} | return HttpNotFound(); |

**Class:** ProductController

**Unit Test Case-19 (UTC-19):** public ActionResultDeleteConfirmed (int id)

**Description:** This unit test case used to check procedure of DeleteConfirmed (int id) method that can delete a product from the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Delete with valid product identification | {id == 2} | return View("Index"); |
| 2 | Delete with invalid product identification | {id == 500} | return View (product); |

**Class:** AccountController

**Unit Test Case-20 (UTC-20):** public ActionResultLogOut ()

**Description:** This unit test case used to check procedure of ActionResultLogOut () method that can logout from the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test can logout and go to the index page | - | return RedirectToAction ("Index", "Home"); |

**Class:** AccountController

**Unit Test Case-21 (UTC-21):** private ActionResultRedirectToLocal(string returnUrl)

**Description:** This unit test case used to check procedure of ActionResultRedirectToLocal (string returnUrl) method that return to the true Url.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return Url is null | {return Url == null} | return RedirectToAction ("Index", "Home"); |
| 2 | Return Url is not null | {return Url == \Product\Index} | return Redirect (returnUrl); |

**Class:** IngredientController

**Unit Test Case-22 (UTC-22):** public ActionResultDetails (int id = 0)

**Description:** This unit test case used to check Details (int id=0) method that can go to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to an ingredient information page and valid ingredient identification | {id == 2} | return View ("Details", {Name == “Raisin”,  Size == 50, ingredID ==2, Category == Raisin, Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg}); |
| 2 | Return to an ingredient information page and Invalid ingredient identification | {id == 500} | return HttpNotFound (); |

**Class:** IngredientController

**Unit Test Case-23 (UTC-23):** public ActionResult Create ()

**Description:** This unit test case used to check Create () method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **input** | **Expected output** |
| 1 | Return to create ingredient information page of the admin main layout | - | return View ("Create"); |

**Class:** IngredientController

**Unit Test Case-24 (UTC-24):** public ActionResultCreate(Ingredient ingred)

**Description:** This unit test case used to check Create (Ingredient ingre) method that can create new ingredient information to the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {Name ==“Raisin”, Size == 1,  Category == Raisin , Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg } | return RedirectToAction ("Index"); |
| 2 | Test with input incorrect information | {Name ==“Raisin”, Size == xxx,  Category == Raisin , Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg } | {Name ==“Raisin”, Size == xxx,  Category == Raisin , Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg } |

**Class:** IngredientController

**Unit Test Case-25 (UTC-25):** public ActionResultEdit (int id = 0)

**Description:** This unit test case used to check Edit (int id = 0) method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to edit ingredient information page that is selected | {id == 2} | return View ("Edit", {Name == “Raisin”, Size == 1, ingredID ==2, Category == Raisin, Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg}); |
| 2 | Return to edit ingredient information page that not selected | {id == 500} | return HttpNotFound (); |

**Class:** IngredientController

**Unit Test Case-26 (UTC-26):** public ActionResultEdit (Ingredient ingre)

**Description:** This unit test case used to check Edit (Ingredient ingre) method that can update ingredient information to the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {Name ==“Raisin”, Size == 20, Category == Raisin, Picture \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg } | return RedirectToAction ("Index", "Ingredient", {Name ==“Raisin”, Size == 20, Category == Raisin, Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg } |
| 2 | Test with input incorrect information | {Name ==“Raisin”, Size == xxx, Category == Raisin , Picture \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg } | return View {Name ==“Raisin”, Size == xxx, Category == Raisin , Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg }); |

**Class:** IngredientController

**Unit Test Case-27 (UTC-27):** public ActionResultDelete (int id = 0)

**Description:** This unit test case used to check Delete (int id = 0) method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to delete ingredient page that is selected | {id == 2} | return View ("Delete", {Name ==“Raisin”,  Size == 1, Category == Raisin, Picture == \\UploadImages\\210733446\_10202588282381156\_594892420\_n.jpg }); |
| 2 | Return to delete ingredient page that not selected | {id == 500} | return HttpNotFound (); |

**Class:** IngredientController

**Unit Test Case-28 (UTC-28):** public ActionResultDeleteConfirmed (int id)

**Description:** This unit test case used to check procedure of DeleteConfirmed (int id) method that can delete ingredient information from the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Delete with valid ingredient id | {id == 2} | return View ("Index"); |
| 2 | Delete with invalid product identification | {id == 500} | return View (product); |

**Class:** HomeController

**Unit Test Case-29(UTC-29):** public SendCart (Order ord, string submit)

**Description:** This unit test case used to check SendCart (Order ord, string submit) method that can add the order to the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to the order page and add order information to the database | {id == 22, orderDate, Duedate == {10/28/2014 12:00:00 AM}, orderStatus == 0, memberId == 5} | return View ("History",{id == 22, orderDate, Duedate == {10/28/2014 12:00:00 AM}, orderStatus == 0,  memberId == 5}); |
| 2 | Return to the product page of Homecontroller | - | return View ("Product", "Home"); |

**Class:** HomeController

**Unit Test Case-30(UTC-30):** public ActionResult Product ()

**Description:** This unit test case used to check Product () method that can return to the correct page.

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Expected output** |
| 1 | Return to the product page and load a product list | return View ("Product", List of product); |

**Class:** HomeController

**Unit Test Case-31(UTC-31):** public ActionResult Product (IList<Product> products)

**Description:** This unit test case used to check Product () method that can save a product to session and send to the product cart page.

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Expected output** |
| 1 | Return to the product cart page and show a product list in session | Return RedirectToAction ("Cart", "Home"); |

**Class:** HomeController

**Unit Test Case-32(UTC-32):** public ActionResult History ()

**Description:** This unit test case used to check History() method that can return to the correct page.

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Expected output** |
| 1 | Return to the order page and show order history | return View ("History", {List of wholesaler order}); |

**Class:** OrderController

**Unit Test Case-33(UTC-33):** public ActionResult Index ()

**Description:** This unit test case used to check Index () method that can return to the correct page.

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Expected output** |
| 1 | Return to the index page and show an order list | return View ("Index",List of order); |

**Class:** RecipeController

**Unit Test Case-34(UTC-34):** public ActionResult Delete (int id=0)

**Description:** This unit test case used to check Delete () method that can delete an ingredient from menu recipe correctly.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Delete with valid ingredient identification | {id == 2} | return View ("EditRecipe"); |
| 2 | Delete with invalid ingredient identification | {id == 500} | return View (recipe); |

**Class:** RecipeController

**Unit Test Case-35(UTC-35):** public ActionResult Edit (int id=0)

**Description:** This unit test case used to check Edit () method that edit ingredient quantity and units of the menu recipe correctly.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {id == 2} | return View ("EditRecipe ", productID == 2,  productName == “Coconut Cake”, price==50,  category == ”Cake”, productPicture == “..\..\\UploadImages\2images (4).jpg”, information ==null} |
| 2 | Test with input incorrect information | {id == 200} | return View ("EditRecipe ", productID == 2,  productName == ”Coconut Cake”, price==50,  category == ”Ca320”, productPicture == ”..\..\\UploadImages\2images (4).jpg”, information ==null} |

**Class:** RecipeController

**Unit Test Case-36 (UTC-36):** public ActionResult Get (int id=0)

**Description:** This unit test case used to check get () method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Return to the menu recipe page and show a menu recipe. | {id == 2} | return View ("EditRecipe ", productID == 2 ,  productName ==” Coconut Cake”, price == 50,  category == ”Cake”, productPicture == ”..\..\\UploadImages\2images (4).jpg”, information == null} |
| 2 | Return to menu recipe page and not show a menu recipe. | {id == 200} | Return HttpNotFound (); |

**Class:** RecipeController

**Unit Test Case-37(UTC-37):** public ActionResult Index ()

**Description:** This unit test case used to check Index () method that can return to the correct page.

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Expected output** |
| 1 | Return to Index page and show a menu recipe list | return View ("Index",List of recipe); |

**Class:** RecipeController

**Unit Test Case-38(UTC-38):** public ActionResult Insert (int id=0)

**Description:** This unit test case used to check Insert () method that can add ingredient quantity and units to the menu recipe correctly.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {id == 2 } | return View ("EditRecipe ", productID == 2,  productName == ”Coconut Cake”, price == 50,  category == ”Cake”, productPicture == ”..\..\\UploadImages\2images (4).jpg”, information == null} |
| 2 | Test with input incorrect information | {id ==200 } | return View ("EditRecipe ", productID == 2,  productName == ”Coconut Cake”, price==jkjpo,  category == ”Cake”, productPicture == ”..\..\\UploadImages\2images (4).jpg”, information == null} |

**Class:** RecipeController

**Unit Test Case-39(UTC-39):** public ActionResult List ()

**Description:** This unit test case used to check List () method that can return the correct page.

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Expected output** |
| 1 | Return to the index page and show a menu recipe list with a condition. | return View ("Index",List of recipe); |

**Class:** EstimationController

**Unit Test Case-40(UTC-40):** public ActionResult GenerateReport (Report reportType, DateTime startPeriod, DateTime endPeriod)

**Description:** This unit test case used to check GenerateReport () method that can generate correct report information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | {reportType == ”order1” ,startPeriod == 12/23/2014 12:00:00 AM, endPeriod == 1/23/2015 12:00:00 AM } | return View ("Report ", report} |
| 2 | Test with input incorrect information | {reportType == ”order1”, startPeriod == 20/01/2554 AM, endPeriod == 20/01/2555 12:00:00 AM } | return View (“ListAll “) |

**Class:** EstimationController

**Unit Test Case-41(UTC-41):** public ActionResult GenerateGraph(Product product, DateTime startPeriod, DateTime endPeriod)

**Description:** This unit test case used to check GenerateGraph () method that can generate correct graph information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | { productID == 2, productName == ”Coconut Cake”, price == 50, category == ”Cake”, productPicture == ”..\..\\UploadImages\2images (4).jpg”,  information == null, startPeriod == 12/23/2014 12:00:00 AM, endPeriod == 1/23/2015 12:00:00 AM } | return View ("Graph ", graph} |
| 2 | Test with input incorrect information | { productID == 2, productName == ”Coconut Cake”, price == 50, category == ”Cake”, productPicture == ”..\..\\UploadImages\2images (4).jpg”,  information == null, startPeriod == 20/01/2554 AM, endPeriod == 20/01/2555 12:00:00 AM } | return View (“ListAll “) |

**Class:** EstimationController

**Unit Test Case-42(UTC-42):** public ActionResult Get(int id=0)

**Description:** This unit test case used to check Get () method that can return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Index** | **Expected output** |
| 1 | Return to get ingredient estimation page and show a product and ingredient usage. | {int id = 2} | return View ("Get", productID == 2,  productName == ”Coconut Cake”, price == 50,  category == ”Cake”, productPicture == ”..\..\\UploadImages\2images (4).jpg”, information == null); |

**Class:** EstimationController

**Unit Test Case-43(UTC-43):** public ActionResult List(DateTime startPeriod, DateTime endPeriod)

**Description:** This unit test case used to check List () method that can select correct period and return to the correct page.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input correct information | { startPeriod == 12/23/2014 12:00:00 AM, endPeriod == 1/23/2015 12:00:00 AM } | return View ("List ",List of order } |
| 2 | Test with input incorrect information | { startPeriod == 20/01/2554 AM,  endPeriod == 20/01/2555 12:00:00 AM } | return HttpNotFound () |

**Class:** EstimationController

**Unit Test Case-44(UTC-44):** public ActionResult ListAll()

**Description:** This unit test case used to check ListAll () method that can return to the correct page.

**Test Cases:**

|  |  |  |
| --- | --- | --- |
| **No.** | **Description** | **Expected output** |
| 1 | Return to ingredient estimation page and show all ingredient estimation list from the order | return View ("Listall",List of order); |

# 

# **Chapter Four | System Testing of Web-based Ordering & Ingredient Estimating for Bakery Manufacturer**

**System Test Case-01 (STC-01):** Admin can register to activate member to access the application by create new member information.

**Description:** This test case is a system testing for URS-01. Admin can register member information (username, password, first name, last name, phone number, address, role name, and profile picture) and add member information into the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:**  1. Admin clicks “Member Management” link.

2. Admin clicks “Create Member” link.

3. Admin inputs new member information into the registration form.

4. Admin clicks “Save” button.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the registration form and member information can add into the database. | **First name:** “Nontra”  **Last name:** “Mahachanont”  **Username:** nontranest@gmail.com  **Password:** aWefk2343  **Phone Number:** “0812345678”  **Address:** “House No.197 Chroenrat Road, Watgate sub-district, Mueng Chiang Mai district, Chiang Mai province, 5000, Thailand”  **Role name:** Admin  vcm_s_kf_m160_155x160.jpg | Add new member information success and keep new information in the database. After that the system will redirect to member profile page. |
| 2 | Test with do not input username information to the registration form. | **Username:** | Display error message “Username is required.” After that the system gives an admin input new information again. |
| 3. | Test with do not input password information to the registration form. | **Password:** | Display error message “Password is required.” After that the system gives an admin input new information again. |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 4. | Test with do not input first name information to the registration form. | **First name:** “ ” | Display error message “First name is required.” After that the system gives an admin input new information again. |
| 5 | Test with do not input last name information to the registration form. | **Last name:** “ ” | Display error message “Last name is required.” After that the system gives an admin input new information again. |
| 6 | Test with do not input phone number information to the registration form. | **Phone Number:** “ ” | Display error message “Phone number is required.” After that the system gives an admin input new information again. |
| 7 | Test with do not input address information to the registration form. | **Address:** “ ” | Display error message “Address is required.” After that the system gives an admin input new information again. |
| 8 | Test with invalid type of input information. To the registration form. | **Username:** go | Display error message “Please input with Email.” After that the system gives an admin input new information again. |

**System Test Case-02 (STC-02):** Member authenticates the web application account by login.

**Description:** This test case is a system testing for URS-02. Member can authentication to the web application account by login.

**Prerequisite:** Member must have own account.

**Test Script:** 1. Member enters to the website.

2. Member fills username as “tung@dd.com” and password “123456”.

3. Member clicks “Login” button to sign in to the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input corrects both username and password to the login form. | **Username** = [tung@dd.com](mailto:tung@dd.com)  **Password** = 123456 | The system will redirect to the main layout page. |
| 2 | Test with inputs username is invalid with password in the database. | **Username** = [tung@gmail.com](mailto:tung@gmail.com)  **Password** = 147852369 | Display error message “The username or password are incorrect.” After that the system give a member input information again. |
| 3 | Test with inputs username is not Email in login form. | **Username** = “No”  **Password** = 123456 | Display error message “Please input with Email.” After that the system gives a member input information again. |
| 4 | Test with do not input username information to login form. | **Username** = “ ”  **Password** = 123456 | Display error message “Username is required.” After that the system gives a member input information again. |
| 5 | Test with do not input password information to login form. | **Username** = [tung@dd.com](mailto:tung@dd.com)  **Password** = “ ” | Display error message “Password is required.” After that the system gives a member input information again. |
| 6 | Test with login can reach and keep sign in date. | - | Login to the system success and keep a sign in date in the database. |

**System Test Case-03 (STC-03):** Member authenticates the web application account by logout.

**Description:** This test case is a system testing for URS-02. Member can authentication to the web application account by logout.

**Prerequisite:** Member must login first.

**Test Script:** 1. Top right of a page, member clicks “menu” list.

2. Member selects “Logout” link.

3. Member clicks “Logout” button to sign out from the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with logout can out from the system and return to the home page. | - | Logout from the system successful. After that the system redirects to the home page. |

**System Test Case-04 (STC-04):** Admin can manage all member information by edit member.

**Description:** This test case is a system testing for URS-03. Admin can manage all member information by edit member information (username, password, first name, last name, phone number, address, role name, and profile picture) and update member information into the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. Admin clicks “Member Management” link.

2. Admin clicks “Edit Member” link.

3. Admin updates information into the member profile form.

4. Admin clicks “Save” button.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the member profile form and member information can update into the database. | **First name:** “Nontra”  **Last name:** “Mahachanont”  **Username:** nontranest@gmail.com  **Password:** aWefk2343  **Phone Number:** “0812345678”  **Address:** “House No.197 Chroenrat Road, Watgate sub-district, Mueng Chiang Mai district, Chiang Mai province, 5000, Thailand”  **Role name:** Admin  vcm_s_kf_m160_155x160.jpg | Update member information success and keep updated information in the database. After that the system will redirect to the member profile page. |
| 2 | Test with invalid type of input information. To the member profile form. | **Username:** gone | Display error message “Please input with Email.” After that the system gives an admin input information again. |

**System Test Case-05 (STC-05):** Admin can manage all member information by delete member.

**Description:** This test case is a system testing for URS-03. Admin can manage all member information by delete member and clear member information from the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. Admin clicks “Member Management” link.

2. Admin clicks “Delete Member” link.

3. To confirms, admin clicks “Save” button again.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with a member account can delete from the database. | - | Member information has removed and keep in the database. After that the system will redirect to the admin main layout page. |

**Test Cases:**

**System Test Case-06 (STC-06):** Member can update own information.

**Description:** This test case is a system testing for URS-04. Member can update own information (username, password, first name, last name, phone number, address, and profile picture) and update own information into the database.

**Prerequisite:** Member must login with their member account.

**Test Script:** 1. Top right of a page, member clicks “menu” list.

2. Member selects “profile” button.

3. Member views own details.

4. Member clicks “Edit” button to go to edit personal profile.

5. Member can update own information (STC-04).

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the member profile form and member information can update into the database. | **First name:** “Nontra”  **Last name:** “Mahachanont”  **Username:** nontranest@gmail.com  **Password:** aWefk2343  **Phone Number:** “0812345678”  **Address:** “House No.197 Chroenrat Road, Watgate sub-district, Mueng Chiang Mai district, Chiang Mai province, 5000, Thailand”  **Role name:** Admin  vcm_s_kf_m160_155x160.jpg | Update member information success and keep updated information in the database. After that the system will redirect to the member profile page. |
| 2 | Test with invalid type of input information. To the member profile form. | **Username:** gone | Display error message “Please input with Email.” After that the system gives an admin input information again. |

**System Test Case-07 (STC-07):** Member can view own personal information.

**Description:** This test case is a system testing for URS-05. Member can view their personal information from the database.

**Prerequisite:** Member must has login to the system.

**Test Script:** 1. Member clicks “Profile” link.

2. Member clicks, “View” button.

4. Member views own member information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks view profile button. | - | Display member information on the profile page. |

**System Test Case-08 (STC-08):** Admin can view all member information by search the filter keyword of a member.

**Description:** This test case is a system testing for URS-06. Admin can search all member information (member ID, username, first name, last name, phone number, address, role name, and profile picture) and view member information from the database.

**Prerequisite:** Adminer must has login to the system.

**Test Script:** 1. Admin clicks “Member Management” link.

2. Admin inputs the filter keyword of a member on a search box.

3. Admin clicks “search” button to search the expected member result.

4. Admin views member information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with valid information in the database | “Nontra” | Found information and display member information. |
| 2 | Test with invalid information in the database | “d” | Not found a member information and display the not found message. |
| 3 | Test with clicks search button. | - | Display all member are related to the filter keyword on the member information page. |

**System Test Case-09 (STC-09):** Admin can manage all bakery product information by create new product.

**Description:** This test case is a system testing for URS-07. Admin can manage all bakery product information by create new bakery product information (product name, information, price, category, and product picture) and add new bakery product information into the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. Admin clicks “Product management” button.

2. Admin clicks “Create new” link.

3. Admin inputs new bakery product information to the product form.

4. Admin clicks “Save” button to save new bakery product.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the product form and new bakery product information can add into the database. | **Product Name:** “Brownie”  **Information:** A chocolate brownie is a flat or [baked](http://en.wikipedia.org/wiki/Baking) square.  **Price:** 29.00  **Category:** “Bread”  **Aglayan pasta.jpg** | Add bakery product information success and keep new information in the database. After that the system will redirect to the product page. |
| 2 | Test with do not input product name to the product form. | **name** = “ ” | Display error message “Name is required.” After that the system gives an admin input new information again. |
| 3 | Test with do not input price information to the product form. | **price** = “ ” | Display error message “Price is required.” After that the system gives an admin input new information again. |
| 4 | Test with do not input category information to the product form. | **category** = | Display error message “Category is required.” After that the system gives an admin input new information again. |
| 5 | Test with do not input picture information to the product form. | **picture** = “ ” | Display error message “Picture is required.” After that the system gives an admin input new information again. |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 6 | Test with invalid input information type to the product form. | **Price:** goods | Display error message “Please input with a number.” After that the system gives an admin input new information again. |

**System Test Case-10 (STC-10):** Admin can manage all bakery product information by edit product.

**Description:** This test case is a system testing for URS-07. Admin can manage all bakery product information by edit bakery product information (product name, information, price, category, and product picture) and update bakery product information into the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. Admin clicks “Product Management” button.

2. Admin clicks “Edit Product” link.

3. Admin inputs update information to product form.

4. Admin clicks “Save” button.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the product form and bakery product information can update into the database. | **Product Name:** “Brownie”  **Information:** A chocolate brownie is a flat or [baked](http://en.wikipedia.org/wiki/Baking) square.  **Price:** 29.00  **Category:** “Bread”  **Aglayan pasta.jpg** | Update bakery product information success and keep updated information in the database. After that the system will redirect to the product page. |
| 2 | Test with invalid input information type to the product form. | **Price:** goods | Display error message “Please input with a number.” After that the system gives an admin input new information again. |

**System Test Case-11 (STC-11):** Admin can manage all bakery product information by delete product.

**Description:** This test case is a system testing for URS-07. Admin can manage all bakery product information by remove a bakery product and clear bakery product information from the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. Admin clicks “Product Management” button.

2. Admin clicks “Delete Product” link.

3. To confirms, admin clicks “Save” button again.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with bakery product information can delete from the database. | - | Bakery product information has deleted and keep in the database. After that the system will redirect to the product page. |

**System Test Case-12 (STC-12):** All user can view all bakery product information by search the filter keyword of a bakery product.

**Description:** This test case is a system testing for URS-08. User can search all bakery product information (product ID, product name, price, and category) and view bakery product information from the database.

**Prerequisite:** User must enters to the website.

**Test Script:** 1. User clicks “Product” link.

2. User inputs a filter keyword of bakery product on search box.

3. User clicks “search” button to search the expected bakery product result.

4. Wholesaler views bakery product information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with valid information in the database | “Brownie” | Found information and display a bakery product information. |
| 2 | Test with invalid information in the database | “lsd” | Not found bakery product information and display the empty page. |
| 3 | Test with clicks search button. | - | Display all bakery products are related to with the filter keyword on the product page. |

**System Test Case-13 (STC-13):** Wholesaler can order bakery products from the existing product in the system.

**Description:** This test case is a system testing for URS-09. Wholesaler can order bakery products from the existing bakery product information. An order can select by select product name, product quantity, and delivery date.

**Prerequisite:** Wholesaler must login with the wholesaler account.

**Test Script:** 1. On left menu bar of the wholesaler main layout page, wholesaler clicks “Product” link.

2. Wholesaler inputs new order information into the product cart.

3. Wholesaler clicks “Submit” button for add bakery product into the product cart.

4. Wholesaler clicks “Send Order” button.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the product cart and order information can send into the database. | **Product Name:** “Difuku”  **Product Quantity:** 340  **Delivery Date:** 13 October 2014 | Send new order information success and keep into the database. After that the system will redirect to current order page. |
| 2 | Test with do not input product quantity to the product cart. | **Product Name:** “Difuku”  **Product Quantity:**  **Delivery Date:** 13 October 2014 | Send new order information unsuccessful and not keep into the database. After that the system gives a wholesaler input new information again. |
| 3 | Test with do not input delivery date to the product cart. | **Product Name:** “Difuku”  **Product Quantity:** 340  **Delivery Date:** | Display error message “The Delivery Date field is required.” After that the system gives a wholesaler input new information again. |
| 4. | Test with send order follows the time condition. | **-** | Send new order information success and keep into the database. After that the system will redirect to current order page. |
| 5 | Test with send order out of the time condition. | **-** | Display unsuccessful message “After midday, you cannot make an order because the bakery product manufacturing in the stage.” |

**System Test Case-14 (STC-14):** Wholesaler can manage own order information by edit bakery product name, bakery product quantity and delivery date in the order.

**Description:** This test case is a system testing for URS-10. Wholesaler can manage own order information; product name, product quantity, and delivery date in the order page.

**Prerequisite:** The wholesaler must has login to the system and current order information has created.

**Test Script:** 1. On left menu bar of the wholesaler main layout page, wholesaler clicks “Order” link.

2. Wholesaler inputs update order information into the product cart.

3. Wholesaler clicks “Submit” button for add updated information to the system.

4. Wholesaler clicks “Send Order” button. (STC-13)

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the product cart and order information can send into the database. | **Product Name:** “Difuku”  **Product Quantity:** 340  **Delivery Date:** 13 October 2014 | Send new order information success and keep into the database. After that the system will redirect to current order page. |
| 2 | Test with do not input product quantity to the product cart. | **Product Name:** “Difuku”  **Product Quantity:**  **Delivery Date:** 13 October 2014 | Send new order information unsuccessful and not keep into the database. After that the system gives a wholesaler input new information again. |
| 3 | Test with do not input delivery date to the product cart. | **Product Name:** “Difuku”  **Product Quantity:** 340  **Delivery Date:** | Display error message “The Delivery Date field is required.” After that the system gives a wholesaler input new information again. |
| 4. | Test with send order follows the time condition. | **-** | Send new order information success and keep into the database. After that the system will redirect to current order page. |
| 5 | Test with send order out of the time condition. | **-** | Display unsuccessful message “After midday, you cannot make an order because the bakery product manufacturing in the stage.” |

**System Test Case-15 (STC-15):** Wholesaler can manage own order information by cancel all bakery product information from the system.

**Description:** This test case is a system testing for URS-10. Wholesaler can manage own order information; product name, product quantity, and delivery date in the order page.

**Prerequisite:** The wholesaler must has login to the system and current order information has created.

**Test Script:** 1. On left menu bar of the wholesaler main layout page, wholesaler clicks “Order” link.

2. Wholesaler clicks “Cancel” link for remove own ordered.

3. Wholesaler clicks “Save” button for clear order information from the database.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with order information can remove from the database and return to the order page. | - | Not found order information remain in the system and the system redirect to the order page. |
| 2 | Test with cancel order information after midday and the system display an unsuccessful message to the wholesaler. | - | Display unsuccessful message “After midday, you cannot cancel the order because the bakery product manufacturing in the stage.” |

**System Test Case-16 (STC-16):** Wholesaler can view their current order’s information.

**Description:** This test case is a system testing for URS-11. Wholesaler can view their current order’s information; product name, product quantity, total price, and discount of the order.

**Prerequisite:** Wholesaler must has login to the system.

**Test Script:** 1. On left menu bar of the wholesaler main layout page, wholesaler clicks “Order” link.

2. Wholesaler selects “View current” button.

3. Wholesaler views current order information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with the current order can retrieve all information from the system and return to the current order page. | - | Display order information from the system in the current order page. |

**System Test Case-17 (STC-17):** Employee can view current order’s information of all members by search the filter keyword.

**Description:** This test case is a system testing for URS-12. Employee can view current order’s information of all members by search the filter keyword of order ID, first name, last name, order status, and delivery date in the database.

**Prerequisite:** Employee must login with the employee account.

**Test Script:** 1. On left menu bar of the employee main layout page, employee clicks “Order Management” link.

2. Employee inputs the filter keywords into the search box.

3. Employee selects one label from all current order list to view order information.

4. Employee views current order information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with valid information in the database | **First name:** “Nontra” | Found information and display order information are related to the filter keyword. |
| 2 | Test with invalid information in the database | **First name:** “xxx” | Not found order information and display the not found message. |

**System Test Case-18 (STC-18):** Employee can view total ordered information.

**Description:** This test case is a system testing for URS-13. Employee can view total ordered information; product name, total product, and summary price of all order information.

**Prerequisite:** Employee must has login to the system.

**Test Script:** 1. On left menu bar of the employee main layout page, employee clicks “Order” link.

2. Employee inputs the filter keywords into the search box.

3. Employee selects one label from all total ordered list to view a result.

4. Employee views total ordered information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with the total ordered can retrieve all information from the system and return to the total ordered page. | - | Display total ordered information from the system in the total ordered page. |
| 2 | Test with valid information in the database | **Search Box:** 23 January 2014 | Found information and display order information are related to the filter keyword. |
| 3 | Test with invalid information in the database | **Search Box:** xxx | Not found order information and display the not found message. |

**System Test Case-19 (STC-19):** Employee can mark progressive of order status while the order in the preparation stage.

**Description:** This test case is a system testing for URS-14. Employee can mark progressive of order status while the order in the preparation stage.

**Prerequisite:** Employee must login with the employee account.

**Test Script:** 1.On left menu bar of the employee main layout page, employee clicks “Order Management” link.

2. Employee selects a current order.

3. Employee selects an order status from a drop down list.

4. To confirms, employee clicks “Save” button again.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with order status can changed and display new order status in the system. | - | Order status has changed and keep in the database. After that the system will display update order status to the web application. |

**Test Cases:**

**System Test Case-20 (STC-20):** Wholesaler can view their ordered history.

**Description:** This test case is a system testing for URS-15. Wholesaler can view their ordered history by search the filter keyword of the delivery date in the database.

**Prerequisite:** Wholesaler must login with the wholesaler account.

**Test Script:** 1. On left menu bar of the wholesaler main layout page, wholesaler clicks “Order” link.

2. Wholesaler inputs the filter keywords into the search box.

3. Wholesaler selects one label from all ordered history list to view a result.

4. Wholesaler views own ordered history.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks views ordered history on the order bar. | - | Display the ordered history information on the order page. |
| 2 | Test with valid information in the database | **Search Box:** 23 January 2014 | Found information and display ordered history information are related to the filter keyword. |
| 3 | Test with invalid information in the database | **Search Box:** xxx | Not found ordered history information and display the not found message. |

**System Test Case-21 (STC-21):** Employee can view all ordered history.

**Description:** This test case is a system testing for URS-16. Employee can view all ordered history by search the filter keyword of the delivery date, order ID, member ID, first name, and last name in the database.

**Prerequisite:** Employee must login with the employee account.

**Test Script:** 1. On left menu bar of the employee main layout page, employee clicks “Order” link.

2. Employee inputs the filter keywords into the search box.

3. Employee selects one label from all ordered history list to view a result.

4. Employee views all ordered history. (STC-20)

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks views ordered history on the order page. | - | Display the ordered history information on the order page. |
| 2 | Test with valid information in the database | **Search Box:** 23 January 2014 | Found information and display ordered history information are related to the filter keyword. |
| 3 | Test with invalid information in the database | **Search Box:** xxx | Not found ordered history information and display the not found message. |

**System Test Case-22 (STC-22):** Wholesaler can comment a suggestion message to the feedback page.

**Description:** This test case is a system testing for URS-17. Wholesaler can publish own suggestion message to the feedback page.

**Prerequisite:** The wholesaler must has login to the system.

**Test Script:** 1. On left menu bar of the wholesaler main layout page, wholesaler clicks “Feedback” link.

2. Wholesaler comments a suggestion message into the textbox.

3. Wholesaler clicks “Post” button for add comment text to the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with comment and post to the feedback page. | **Comment:** “I unable to received the order entirely.” | Post a comment success and keep into the database. After that the system will redirect to the feedback page. |
| 2 | Test with do not comment a message and post. | **Comment:** “ ” | Post a comment unsuccessful and not keep into the database. After that the system gives a wholesaler comment new message again. |

**System Test Case-23 (STC-23):** Employee and Wholesaler can view all suggestion message on the feedback page.

**Description:** This test case is a system testing for URS-18. Employee and Wholesaler can view all suggestion message on the feedback page.

**Prerequisite:** Employee and Wholesaler must login with their account.

**Test Script:** 1. On left menu bar of the main layout page, clicks “Feedback” link.

2. Employee and Wholesaler view all suggestion message.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks feedback link on the menu bar. | - | Display the feedback page and display all suggestion message on the feedback page. |

**System Test Case-24 (STC-24):** Admin can manage all ingredient information by create new ingredient information.

**Description:** This test case is a system testing for URS-19. Admin can manage all ingredient information by creating ingredient name, category, and ingredient picture in the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Ingredient Management” link.

2. Admin clicks “Create new” link.

3. Admin inputs new ingredient information to the ingredient form.

4. Admin clicks “Save” button to save new ingredient.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the ingredient form and new ingredient information can add into the database. | **Ingredient Name:** “Garlic”  **Category:** “Bread”  **Picture**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Add ingredient information success and keep in the database. After that the system will redirect to the ingredient management page. |
| 2 | Test with do not input ingredient name to the ingredient form. | **Ingredient Name:** “ ”  **Category:** “Bread”  **Picture:**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Display error message “Ingredient Name is required.” After that the system gives an admin input new information again. |
| 3 | Test with do not input category information to the ingredient form. | **Ingredient Name:** “Garlic”  **Category:** “ ”  **Picture:**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Display error message “Category is required.” After that the system gives an admin input new information again. |
| 4 | Test with do not input picture information to the ingredient form. | **Ingredient Name:** “Garlic”  **Category:** “Spices”  **Picture:** | Display error message “Ingredient Picture is required.” After that the system gives an admin input new information again. |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 5 | Test with input invalid information type. To the ingredient form. | **Ingredient Name:** “Garlic”  **Category:** “6786”  **Picture:**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Display error message “Please input with alphabet only.” After that the system gives an admin input new information again. |

**System Test Case-25 (STC-25):** Admin can manage all ingredient information by edit the existing ingredient information.

**Description:** This test case is a system testing for URS-19. Admin can manage all ingredient information by editing ingredient name, category, and ingredient picture in the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Ingredient Management” link.

2. Admin clicks “Edit Ingredient” link.

3. Admin inputs update ingredient information to the ingredient form. (STC-24)

4. Admin clicks “Save” button to update ingredient information to the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the ingredient form and new ingredient information can add into the database. | **Ingredient Name:** “Garlic”  **Category:** “Bread”  **Picture**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Add ingredient information success and keep in the database. After that the system will redirect to the ingredient management page. |
| 2 | Test with do not input ingredient name to the ingredient form. | **Ingredient Name:** “ ”  **Category:** “Bread”  **Picture:**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Display error message “Ingredient Name is required.” After that the system gives an admin input new information again. |
| 3 | Test with do not input category information to the ingredient form. | **Ingredient Name:** “Garlic”  **Category:** “ ”  **Picture:**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Display error message “Category is required.” After that the system gives an admin input new information again. |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 4 | Test with do not input picture information to the ingredient form. | **Ingredient Name:** “Garlic”  **Category:** “Spices”  **Picture:** | Display error message “Ingredient Picture is required.” After that the system gives an admin input new information again. |
| 5 | Test with input invalid information type. To the ingredient form. | **Ingredient Name:** “Garlic”  **Category:** “6786”  **Picture:**  http://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg/1024px-Allium_sativum._Restra_de_allos_de_Oroso-_Galiza.jpg  Garlic.jpg | Display error message “Please input with alphabet only.” After that the system gives an admin input new information again. |

**System Test Case-26 (STC-26):** Admin can manage all ingredient information by delete the existing ingredient information.

**Description:** This test case is a system testing for URS-19. Admin can manage all ingredient information by deleting ingredient ID, ingredient name, category and ingredient picture from the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Ingredient Management” link.

2. Admin clicks “Delete Ingredient” link.

3. Admin clicks “Save” button to delete ingredient information from the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with ingredient information can delete from the database. | - | Ingredient information has deleted and keep updated in the database. After that the system will redirect to the ingredient management page. |

**System Test Case-27 (STC-27):** Admin can view all ingredient information by search the filter keywords of ingredient information.

**Description:** This test case is a system testing for URS-20. Admin can view all ingredient information by search the filter keyword of the ingredient ID, ingredient name, and category in the database.

**Prerequisite:** The admin must has login to the system and the ingredient information has created in the system.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Ingredient Management” link.

2. Admin inputs the filter keywords in a search box.

3. Admin clicks “Search” button to find ingredient information

4. Admin clicks “View” button to view ingredient information from the ingredient list

5. Admin views ingredient information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks ingredient management link. | - | Display all ingredient list on the ingredient management page. |
| 2 | Test with search valid ingredient information in the database. | **Filter Keyword:** “Garlic” | Found information and display ingredient information that matched to the filter keyword. |
| 3 | Test with search invalid ingredient information in the database. | **Keyword:** “l[[[k” | Not found ingredient information and display the not found message in the ingredient page. |

**System Test Case-28 (STC-28):** Admin can manage all menu recipe information by create new menu recipe information.

**Description:** This test case is a system testing for URS-21. Admin can manage all menu recipe information by creating the product name, ingredient name, ingredient quantity, and units in the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Recipe Management” link.

2. Admin clicks “Create new” link.

3. Admin inputs new menu recipe information to the recipe form.

4. Admin clicks “Save” button to save new menu recipe.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the recipe form and new menu recipe information can add into the database. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** 230  **Units:** Gram | Add menu recipe information success and keep in the database. After that the system will redirect to the recipe management page. |
| 2 | Test with do not input product name to the recipe form. | **Product Name:** “ ”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** 230  **Units:** Gram | Display error message “Product Name is required.” After that the system gives an admin input new information again. |
| 3 | Test with do not input ingredient name to the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “ ”  **Ingredient Quantity:** 230  **Units:** Gram | Display error message “Ingredient Name is required.” After that the system gives an admin input new information again. |
| 4 | Test with do not input ingredient quantity information to the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:**  **Units:** Gram | Display error message “Ingredient Quantity is required.” After that the system gives an admin input new information again. |
| 5 | Test with do not input ingredient units information to the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** 230  **Units:** | Display error message “Ingredient Units is required.” After that the system gives an admin input new information again. |
| 6 | Test with input invalid information type. To the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** jks  **Units:** Gram | Display error message “Please input with number only.” After that the system gives an admin input new information again. |

**System Test Case-29 (STC-29):** Admin can manage all menu recipe information by edit the existing menu recipe information.

**Description:** This test case is a system testing for URS-21. Admin can manage all menu recipe information by editing product name, ingredient name, ingredient quantity, and units in the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Recipe Management” link.

2. Admin clicks “Edit Recipe” link.

3. Admin inputs update menu recipe information to the recipe form. (STC-28)

4. Admin clicks “Save” button to update menu recipe information to the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with input all correctly information to the recipe form and new menu recipe information can add into the database. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** 230  **Units:** Gram | Add menu recipe information success and keep in the database. After that the system will redirect to the recipe management page. |
| 2 | Test with do not input product name to the recipe form. | **Product Name:** “ ”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** 230  **Units:** Gram | Display error message “Product Name is required.” After that the system gives an admin input new information again. |
| 3 | Test with do not input ingredient name to the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “ ”  **Ingredient Quantity:** 230  **Units:** Gram | Display error message “Ingredient Name is required.” After that the system gives an admin input new information again. |
| 4 | Test with do not input ingredient quantity information to the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:**  **Units:** Gram | Display error message “Ingredient Quantity is required.” After that the system gives an admin input new information again. |

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 5 | Test with do not input ingredient units information to the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** 230  **Units:** | Display error message “Ingredient Units is required.” After that the system gives an admin input new information again. |
| 6 | Test with input invalid information type. To the recipe form. | **Product Name:** “Cake”  **Ingredient Name:** “Powder”  **Ingredient Quantity:** jks  **Units:** Gram | Display error message “Please input with number only.” After that the system gives an admin input new information again. |

**System Test Case-30 (STC-30):** Admin can manage all menu recipe information by delete the existing menu recipe information.

**Description:** This test case is a system testing for URS-21. Admin can manage all menu recipe information by removing product name, ingredient name, ingredient quantity, and units in the database.

**Prerequisite:** Admin must login with the admin account.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Recipe Management” link.

2. Admin clicks “Delete Recipe” link.

3. Admin clicks “Save” button to delete menu recipe information from the system.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with menu recipe information can delete from the database. | - | Menu recipe information has deleted and keep updated in the database. After that the system will redirect to the recipe management page. |

**System Test Case-31 (STC-31):** Admin can view all menu recipe information by search the filter keywords of menu recipe information.

**Description:** This test case is a system testing for URS-22. Admin can view all menu recipe information by search the filter keyword of recipe ID, product ID, product name, ingredient ID, ingredient name, and category in the database.

**Prerequisite:** The admin must has login to the system and the menu recipe information has created in the system.

**Test Script:** 1. On left menu bar of the admin main layout page, admin clicks “Recipe Management” link.

2. Admin inputs the filter keywords in a search box.

3. Admin clicks “Search” button to find menu recipe information

4. Admin clicks “View” button to view menu recipe information from the menu recipe list

5. Admin views menu recipe information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks recipe management link. | - | Display all menu recipe list on the recipe management page. |
| 2 | Test with search valid menu recipe information in the database. | **Search Box:** “Garlic” | Found information and display menu recipe information that matched to the filter keyword. |
| 3 | Test with search invalid menu recipe information in the database. | **Search Box:** “l[[[k” | Not found menu recipe information and display the not found message on the recipe page. |

**System Test Case-32 (STC-32):** Employee can view current ingredient estimation information in the system

**Description:** This test case is a system testing for URS-23. Employee can view current ingredient estimation information in the database.

**Prerequisite:** Employee must login with the employee account.

**Test Script:** 1. On left menu bar of the employee main layout page, employee clicks “Ingredient Estimation” link.

2. Employee selects “View current” button.

3. Employee views current ingredient estimation information.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Result** |
| 1 | Test with the current ingredient estimation can retrieve all information from the system and return to the ingredient estimation page. | - | Display current ingredient estimation information from the system in the ingredient estimation page. |

**System Test Case-33 (STC-33):** Employee can view all ingredient estimation history.

**Description:** This test case is a system testing for URS-24. Employee can view all ingredient estimation history by search the filter keyword of the delivery date in the database.

**Prerequisite:** Employee must login with the employee account.

**Test Script:** 1. On left menu bar of the employee main layout page, employee clicks “Ingredient Estimation” link.

2. Employee inputs the filter keywords into the search box.

3. Employee selects one label from all ordered history list to view a result.

4. Employee views all ingredient estimation history.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks views ingredient estimation history on the ingredient estimation page. | - | Display the ingredient estimation history information on the ingredient estimation page. |
| 2 | Test with valid information in the database | **Search Box:** 23 January 2014 | Found information and display ingredient estimation history information are related to the filter keyword. |
| 3 | Test with invalid information in the database | **Search Box:** xxx | Not found ingredient estimation history information and display the not found message. |

**System Test Case-34 (STC-34):** Employee and Wholesaler can make a report

**Description:** This test case is a system testing for URS-25. Wholesaler and Employee can make a report from the existing information in the database.

**Prerequisite:** Employee and Wholesaler must have login to the system.

**Test Script:** 1. On left menu bar of the main layout page, clicks “Report” link.

2. Employee and Wholesaler click “Make Report” button.

3. Employee and Wholesaler views a report content.

4. Employee and Wholesaler click “Save” button to export as PDF file format.

5. Employee and Wholesaler click “Print” button to print out as a paper document.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks makes a report on the report page. | - | Display the report contents on the report page. |
| 2 | Test with clicks saves a report on the report page. | - | Export the report contents as PDF file format. |
| 3 | Test with clicks prints a report on the report page. | - | Print out the report contents as a paper document. |

**System Test Case-35 (STC-35):** Employee and Wholesaler can make a graph

**Description:** This test case is a system testing for URS-26. Wholesaler and Employee can make a graph from the statistical data by search the filter keyword of the period in the database.

**Prerequisite:** Employee and Wholesaler must have login to the system.

**Test Script:** 1. On left menu bar of the main layout page, clicks “Graph” link.

2. Employee and Wholesaler click “Make Graph” button.

3. Employee and Wholesaler views a graph image.

4. Employee and Wholesaler click “Save” button to export as PDF file format.

5. Employee and Wholesaler click “Print” button to print out as a paper document.

**Test Cases:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected output** |
| 1 | Test with clicks makes a graph on the graph page. | - | Display the graph image on the graph page. |
| 2 | Test with clicks saves a graph on the graph page. | - | Export the graph image as PDF file format. |
| 3 | Test with clicks prints a graph on the graph page. | - | Print out the graph image as a paper document. |