

# **Working Title**

***High Level Design (HLD)***

# CSC.154.0001 - Group 4

*Eric Perry*

*Eric Hodge*

*Oras Alhammadi*

*George Saad*

*Steven Torres*

*May 2, 2019*

*Version 3.0*



**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary**  **Author(s)** | **Description of Version** | **Date Completed** |
| 1st Draft | Eric Perry  Eric Hodge  Steven Torres  Oras Alhammadi  George Saad | Initial Draft of HLD Doc for Working Title (Group 4’s Project – Spring 2019) | 02/22/19 |
| 2nd Draft | Eric Perry  Eric Hodge  Steven Torres  Oras Alhammadi  George Saad | 2nd Draft of HLD Doc for Working Title (Group 4’s Project – Spring 2019) | 04/12/19 |
| 3rd Draft | Eric Perry  Eric Hodge  Steven Torres  Oras Alhammadi | 3rd and Final draft for submission of the HLD for the Working Title project. | 05/02/19 |

**Review & Approval**

**Requirements Document Approval History**





**Table of Contents**

**Working Title** 1

CSC.154.0001 - Group 4 1

1. General Description 4

1.1 Introduction 4

1.2 Scope of this Document 4

1.3 Overview 4

1.4 Division of Responsibilities 4

2. Architecture 5

3. Security 5

4. Hardware 5

5. User Interface 5

6. Internal Interfaces 5

7. External Interfaces 6

8. Reports 6

8.1 Other Output 6

9. Database 6

10. Configuration Data 6

11. Data Flows 7

10.1 Database UI Interaction Diagram 7

10.2 Application Database Interaction Diagram 8

10.3 Activity Diagram 9

11. UML Diagrams 10

11.1 Use Case Diagram 10

11.2 Database Structure Diagram 11

11.3 Suggested WaitStaff Class Diagram 11

11.4 Sample UI menu 12

11.5 Sample Use Cases 13

## 1. General Description

### 1.1 Introduction

The document is a high-level design overview for (Working Title) App, an application for tracking and managing an ingredients inventory for food industry clients.

### 1.2 Scope of this Document

The scope of this document will cover the structure of the program, architecture, data persistence and flow, user work flow, and hardware requirements.

### 1.3 Overview

*(Working Title)* is an inventory management and tracking application that utilizes a relational database. The system stores an inventory of individual ingredients as well as tables of menu items linked to those ingredients. Users will be able to update, in real time, inventory quantities while processing orders from front house to back house. Wait staff and cooks will be notified if an order cannot be completed due to inventory shortage. Managers will be able to view ingredient and menu item history, to include frequency of use/order by period of time, as well as place orders with vendors online or print out an purchase inventory.

### 1.4 Division of Responsibilities

The following responsibilities have been assigned to group members:

Eric Perry – Project Manager, Front-end to Back-end Integration

Eric Hodge – Project Manager, Back-end to Database Integration

Steven Torres – Front-end design

Oras Alhammadi – Back-end design

Geroge Saad – Database design

## 2. Architecture

The (Working Title) app will utilize a Two-Tier Client/Server architecture to allow multiple work stations connected to a single database. The software will utilize the Model-View-Controller (MVC) architecture pattern. By utilizing MVC, the user’s computer will control data flow, presentation, and computation, freeing up the database from the responsibilities of computational processing.

## 3. Security

The (Working Title) app will allow for individual user accounts with password. Different levels of user accounts will dictate what application views are available to the user. The application will utilize UTF-8 encoded hash and salt implemented with Py-bcrypt.

## 4. Hardware

The (Working Title) app will be designed to run on work stations conforming to the Object Linking & Embedding for Retail Point of Sale (OPOS) Standard as developed for Microsoft Windows based systems as well as standard Windows based PCs.

## 5. User Interface

The server account user interface will offer touchscreen capability to work with common OPOS work stations and is coded with standard HTML5/CSS/JS as a single page application.

The manager account user interface will be served as a web-based application, offering the ability to open new sections within their own tab.

The WaitStaff view will utilize an imaged based user interface with menu items separated into groups. When a menu item is selected, the user will have the option to change ingredients or add a note for special instructions. When submitting an order, the app will display a summary of the order and a confirmation button before submitting the order and printing the order ticket in the back house.

## 6. Internal Interfaces

The program utilizes an MVC pattern that will define the scope of responsibility for each interface. The view and controller will interface through the Flask web microframework to keep external dependencies to a minimum. The front end requests a view from the controller written in Python and with the assistance of Flask web templates, generates and passes a view.

The Python back end will interface with a SQLAlchemy database implementation, and with the assistance of Flask ORM extensions, pull the model into the backend. To ensure framework independence, declarative models will be written in Python with the option to utilize Django ORM deferred if the scale of the project expands in the maintenance phase.

## 7. External Interfaces

The (Working Title) app will be required to take input from both traditional keyboard and mouse controllers but also from OPOS conforming touch interfaces. These inputs will be handled by the Kivy framework for Python to pass user inputs as a simplified Python event.

External interfaces for printing will be limited to Windows OS to ensure OPOS conformity. Printer functionality will be implemented with plain Python utilizing os.startfile with “print” verbs.

## 8. Reports

The (Working Title) app will create reports through web-based application views. Reports will allow for aggregation of historical pricing and use of ingredients and dishes in a customizable time period. Additionally, reports for current inventory, inventory to order, and inventory on order will be available.

### 8.1 Other Output

System will send a notification to servers through the POS when items are low or out as well as notify menu items on special or out.

## 9. Database

The (Working Title) app will utilize a MySQL database for storage and retrieval of inventory, inventory quantities, menu items, archived reports, and managerial notes. The database will interface with the backend utilizing SQLAlchemy with a non-framework specific modeling to allow for future implementation of Django ORS if needed.

## 10. Configuration Data

The (Working Title) app will allow configuration by manager level users for the following items:

Expiration date of items

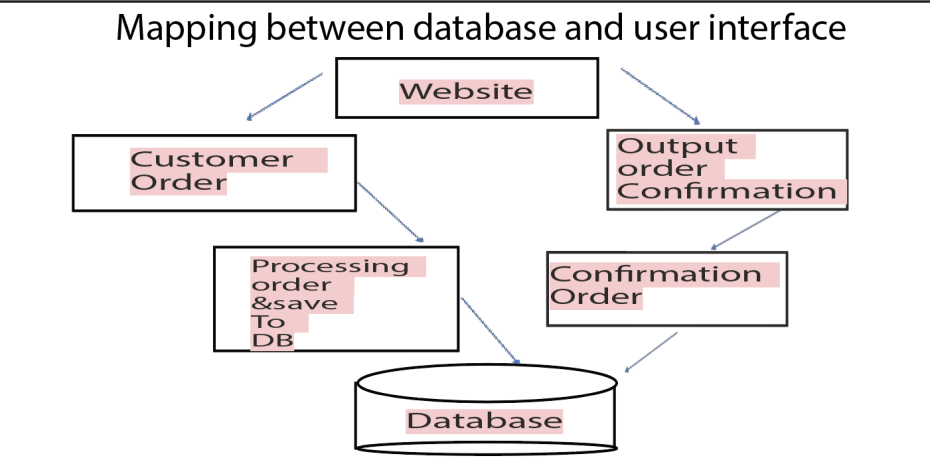
Inventory thresholds

Full SQL query functionality for sorting, exclusion or inclusion, and range for generating reports.

Menu items listed in wait staff view.

## 11. Data Flows

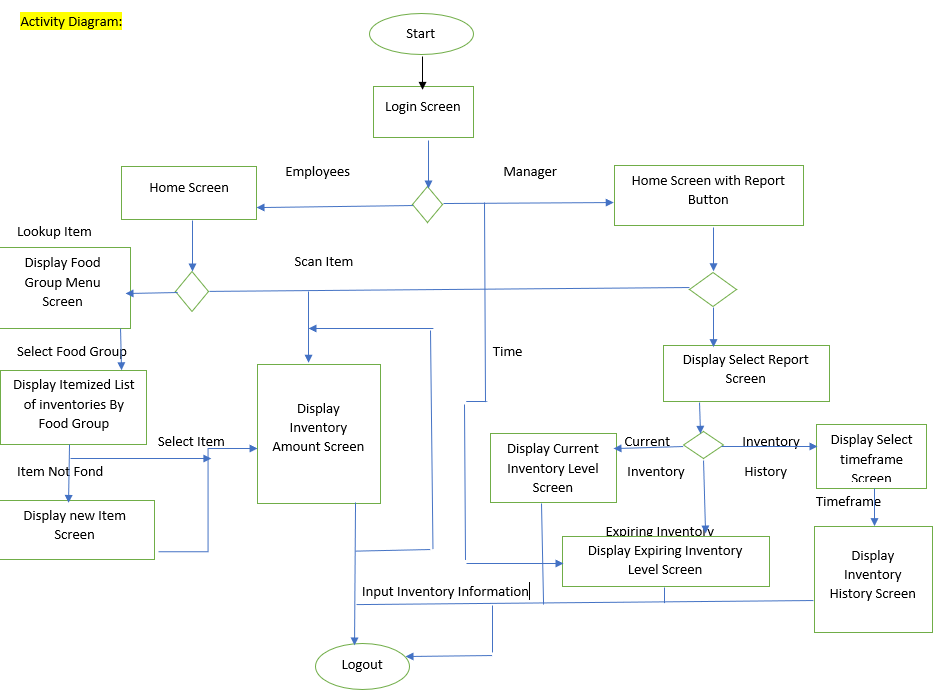
### 10.1 Database UI Interaction Diagram



### 10.2 Application Database Interaction Diagram

|  |
| --- |
| Execution Environment Execution Environment  Python MySQL  Recipe Management  Ingredient Management    Occasion Management  Vendor Management  Database Subsystem    Updates Management  Correction Management    Login Subsystem  Order Management |

### 10.3 Activity Diagram



## 11. UML Diagrams

### 11.1 Use Case Diagram

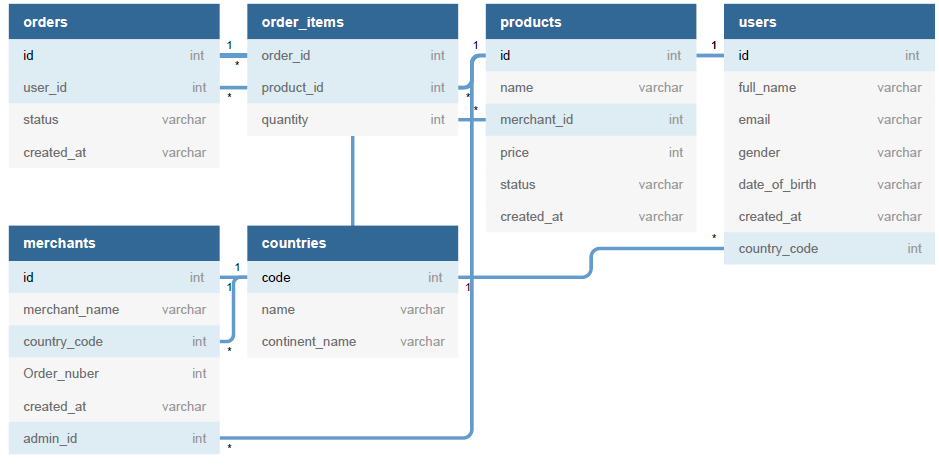
<<include>>

supplier

manager

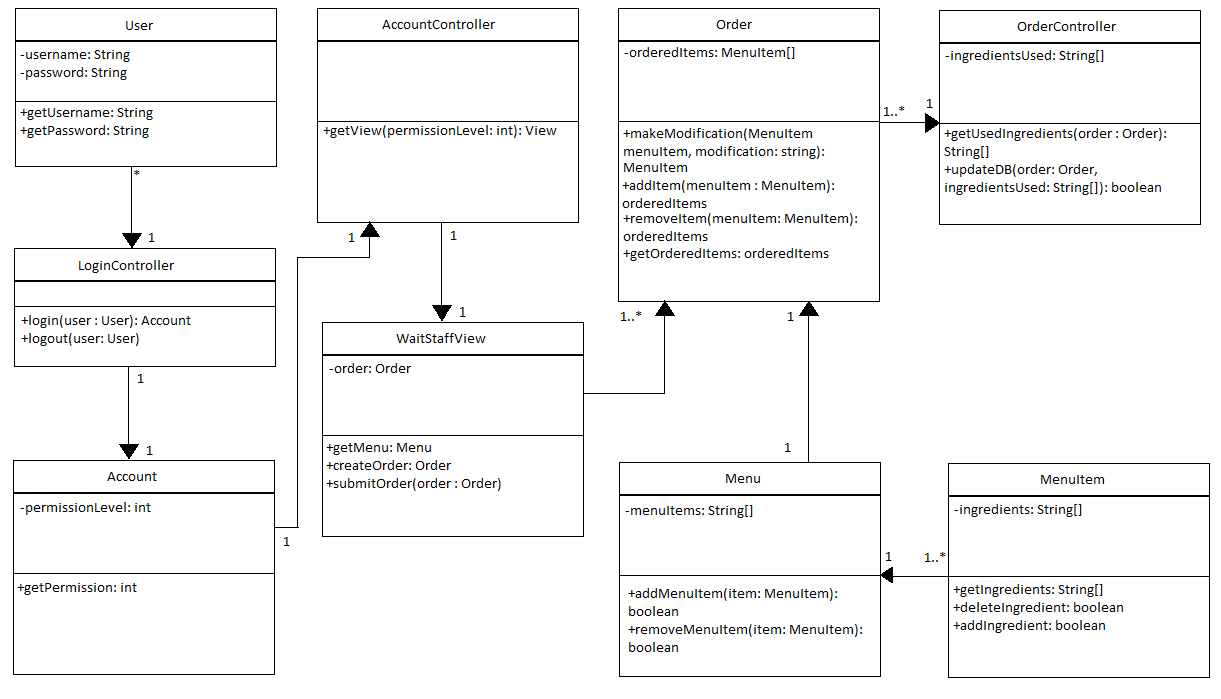
### 

### 11.2 Database Structure Diagram

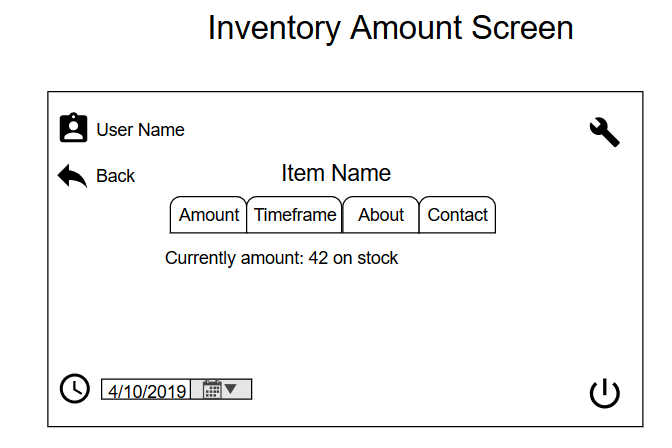
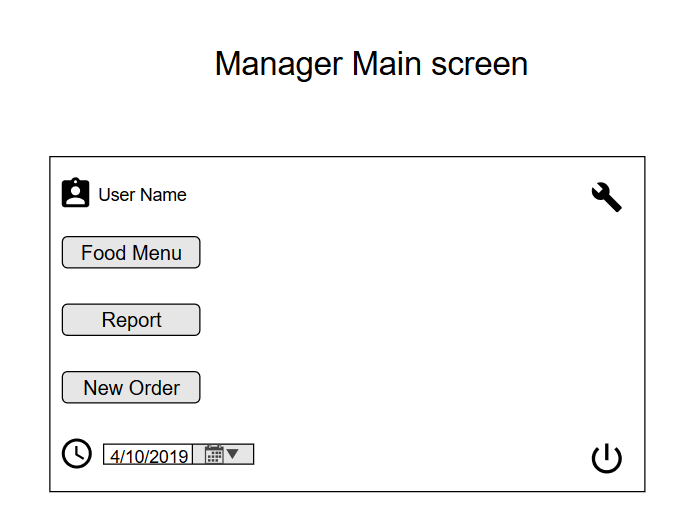


### 11.3 Suggested WaitStaff Class Diagram

Show below is the suggest flow and classes utilized in a WaitStaff transaction.



### 11.4 Sample UI menu



### 

### 11.5 Sample Use Cases

Update Resource Database

|  |  |
| --- | --- |
| Usecase name | **UpdateResourceDatabase** |
| Participating Actors | Initiated by Manager(admin) |
| Flow of events | 1. The manager activates the update resource database function. 2. The system presents a form to the *Manager* . The form asks for details of the sold food items during the course of the week and the corresponding quantity of the food sold. 3. The Manager input the data of the sold food for the week and the quantity that was sold. 4. The System reads the sold food data and then further reads, from the ingredients database, the ingredients that were used in making of the food items that were sold. 5. The System now calculates the amount of resources used and will deduct the amount of ingredients that were used up from the resource database. 6. The system now invokes the **CheckThreshold** Usecase. |
| Entry condition | The Manager(admin) is logged on to the System. |
| Exit condition | If the process was successful, the Manager receives an acknowledgment that the process was completed successfully.  OR  If the process was not successful, the Manager will receive an explanation of what error had occurred during the process. |
| Quality Requirements | The update process must complete successfully and without errors |

Check Threshold Use Case

|  |  |
| --- | --- |
| Usecase name | CheckThreshold |
| Participating actors | Initiated by UpdateResourceDatabase Usecase or by addOccasion Usecase |
| Flow of Events | 1. The system now compares the current levels of the resources with the threshold levels of the resources. It now lists all the ingredients that are below the threshold level, along with the ingredients and presents it to the Manager 2. The manager can now send out orders   Or  Press cancel , no orders are processed. |
| Entry Conditions | The manager is logged on to check the inventory levels at the interval of a certain time period. |
| Exit Conditions | The inventory levels are checked, and the appropriate action is taken. |
| Quality Requirements | The system correctly calculates the correct threshold differences. |

Process Order Use Case

|  |  |
| --- | --- |
| Use case name | ProcessOrder |
| Participating Actors | Initiated by the manager |
| Flow of events | 1. The System now gather a list of vendors from whom the corresponding ingredients are orderd. The System now matches the corresponding ingredients with the vendors from whom ingredients are available.   Loop    2. The System now creates an order and then presents an order summary to the manager. The form has three option: approve the order, revise the order, and cancel the order.  3. The Manager approve the generated order and send it to the vendors, then receiving acknowledgment of the reception of the order and the process ends.  4.The Manager can also choose to revise the order and enter the quantity to be ordered manually for every corresponding ingredient. An updated order summary form is presented to the Manager and the flow returns back to point2.  5.The Manager can also choose to cancel the order, Then the generated order is cancelled, and no orders are sent out. |
| Entry Condition | The user is logged into the system |
| Exit Condition | 1. The Manager approved the order and receiving an acknowledgment of the order being sent. 2. If the orders are not sent out successfully , the Manager receives a massage that the orders were not sent out. 3. The Manager choose to cancel the order. |
| Quality Requirements | The order is sent to the correct vendor |

Add Recipe Use Case

|  |  |
| --- | --- |
| Usecase name | AddRecipe |
| Participating Actors | Initiated by Manager |
| Flow of events | 1. The Manager choose “create New Recipe” from his/her Interface. 2. The System responds by presenting a form to the Manager. The form asks for details associated with the recipe. 3. The Manager complete the form by inserting ingredients to be used in the new recipe. The Manager also input the amount of ingredient to be used in a single order of the recipe. After the form has been completed the Manager submit the form to the System. 4. The System acknowledges that the recipe has been created. It also adds it to the recipe database and any new ingredient to the ingredient database. |
| Entry Condition | The Manager is logged into the System |
| Exit Condition | The Manager has received an acknowledgment from the System  OR  The Manager has received an explanation of why the process couldn’t be completed. |
| Quality Requirements | This use is extended by the AddIngredient Usecase.  The process must complete successfully with the new recipe added to the recipe database without any errors. |

Update Recipe Use Case

|  |  |
| --- | --- |
| Usecase name | UpdateRecipe |
| Participating Actors | Initiated by Manager |
| Flow of events | 1. The Manager choose “Update Recipe” on System. 2. System responds by bringing up a list of recipes. 3. The Manager selects a recipe to change. 4. System now shows updateRecipe form with the list of recipe and corresponding amount. 5. The Manager changes the recipe by adding/ removing or updating the amount of ingredients used in the recipe. The Manager can also add new ingredients that are not available currently by invoking the AddIngredient Usecase. |
| Entry Condition | The Manager is logged into the System. |
| Exit Condition | The Manager receives an acknowledgment from the System  OR  The Manager has received an explanation of why the process couldn’t be completed. |
| Quality Requirements | 1. The Usecase is extended by the AddIngredient Usecase. 2. The update process must be complete successfully without any errors. |

Remove Recipe Use Case

|  |  |
| --- | --- |
| Usecase name | RemoveRecipe |
| Participating Actors | Initiated by Manager |
| Flow of Events | 1. The Manager choose “Remove Recipe” from his/her interface. 2. The System responds by showing the current list of recipes saved on the System. 3. The Manager chooses which recipe(s) to remove. 4. The System confirms with each deletion the Manager if he/she wants to delete the recipe. 5. The Manager confirms his/her decision with a yes/no. 6. The System acknowledges the decision by either removing the recipe if responded with “yes” or by cancelling the delete if responding by “no”. It then displays an acknowledgment of the decision by displaying a delete successful or a canceled request. |
| Entry Condition | The Manager is logged into the System. |
| Exit Condition | The Manager has received an acknowledgment that the recipe has been deleted.  OR  The Manager has received an acknowledgment that the recipe has not been deleted.  OR  The Manager has received an explanation from that the recipe has been deleted. |
| Quality Requirements | The removed recipe should reflect in any other list or connected database. |

Add Occasion Use Case

|  |  |
| --- | --- |
| Usecase name | AddOccasion |
| Participating Actors | Initiated by Manager |
| Flow of events | 1. The Manager choose “Add Occasion or Event” from his/her interface. 2. The System displays a form to be filled out by the manager. 3. The Manager fills out the form by adding a name of the event or occasion and selecting the date(s) the event is to be held. 4. The Manager now fills list of recipes that will be utilized more on the selected day. 5. The System takes the data from the form and calculates the amount of ingredients that may be used up for the given dates based on past data to the ingredients database. 6. The System now invokes the CheckThreshold Usecase. |
| Entry Condition | The Manager logged into the System. |
| Exit Condition | The Manager receives a notification of successful completion of the process  OR  The Manager is notified that the process was not completed with a valid explanation of the error that had occurred during the process. |
| Quality Requirements | The Occasion is accurately added to the database. |

Update Inventory Use Case

|  |  |
| --- | --- |
| Usecase name | UpdateInventory |
| Participating Actors | Initiated by Manager |
| Flow of events | 1. The Manager choose the “Update Stock inventory” from his/her interface. 2. The System now presents a form to the Manager asking for details of the received amount of ingredients. 3. The Manager enters the ingredients and the corresponding quantity received and presses the submit button. 4. The System adds the corresponding amount to the resources database and acknowledges the completion of the process. |
| Entry Condition | The Manager logged into the System. |
| Exit Condition | The Inventory are successfully updated. |
| Quantity Requirements | The number shown to the manager accurately shows the actual amount of ingredients stored. |

Add Vendor Use Case

|  |  |
| --- | --- |
| Usecase name | AddVendor |
| Participating Actors | Initiated by Manager |
| Flow of events | 1. The Manager choose ”Add Vendor “ from his/her interface. 2. The System responds by displaying a form to be completed by the Manager for the vendor to be created. 3. The Manager completes the form by filling the information of the vendor to be created and also the ingredients that will be ordered from that vendor. After all the information has ben filled in, the Manager then submits the form. 4. The System takes the information from the form and adds the vendor the database of vendors. It then displays an acknowledgment to the Manager that the Vendor has been added. |
| Entry Condition | The Manager is logged in System. |
| Exit Condition | The Manager has received an acknowledgment that the vendor has been created.  OR  The Manager has received an explanation of why the process couldn’t be completed. |
| Quality Requirements | The Vendor has been accurately stored into the database. |

Remove Vendor Use Case

|  |  |
| --- | --- |
| Usecase name | RemoveVendor |
| Participating Actors | Initiated by Manager |
| Flow of events | 1. The Manager choose the “Remove Vendor” from his/her interface. 2. The System responds by showing the current list of Vendors saved to the System. 3. The Manager chooses which vendor(s) to remove and removes them. 4. The System confirms with each deletion with the Manager if he/she wants to remove the vendor. 5. The Manager confirms his/ her decision with yes/no. 6. The System acknowledges the decision by either removing the vendor if responded with “yes” or by cancelling the delete if responding by “no”. It then displays an acknowledgment of the decision by displaying a delete successful or a canceled request. |
| Entry Condition | The Manager is logged into the System. |
| Exit Condition | The Manager received an acknowledgment that the vendor has been removed.  OR The Manager has received an acknowledgment that the vendor has not been removed  OR  The Manager has received an explanation of why the process couldn’t be completed. |
| Quality Requirements | The Vendor should not appear in the list of active vendors or any other database. |

Add Ingredients Use Case

|  |  |
| --- | --- |
| Usecase name | AddIngredient |
| Participating Actors | Initiated by Manager |
| Flow of events | 1. The system presents a form to the Manager for adding the new ingredient. 2. The Manager inputs the details of the ingredients and confirms. 3. The System now makes available the new ingredient to the Manager for including it in the recipe. |
| Entry Condition | The Add function is currently running. |
| Exit Condition | The Ingredient is successfully added to the Database. |
| Quality Requirements | The details for the ingredient are correctly added to the database. |