

# **Working Title**

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

# CSC.154.0001 - Group 4

*Eric Perry*

*Eric Hodge*

*Oras Alhammadi*

*George Saad*

*Steven Torres*

*February 15, 2019*

*Version 1.0*



**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 |

**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. External Interface 5

7. Reports 5

7.1 Other Output 5

8. Database 5

9. UML Diagrams 6

9.1 Use Case Diagram 6

9.2 Database UI Interaction Diagram 7

9.3 Application Database Interaction Diagram 8

9.4 Activity Diagram 9

9.5 Database Structure Diagram 9

9.6 Suggested Wait Staff Class Diagram 10

## 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. Market prices for ingredients will also be available through third party integration of APIs/EPIs.

### 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. External Interface

The (Working Title) app will interface with external API/EPIs such as The Farmer’s Market API to assist with online ordering of ingredients and real time market pricing.

## 7. 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.

### 7.1 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.

## 8. 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.

## 9. UML Diagrams

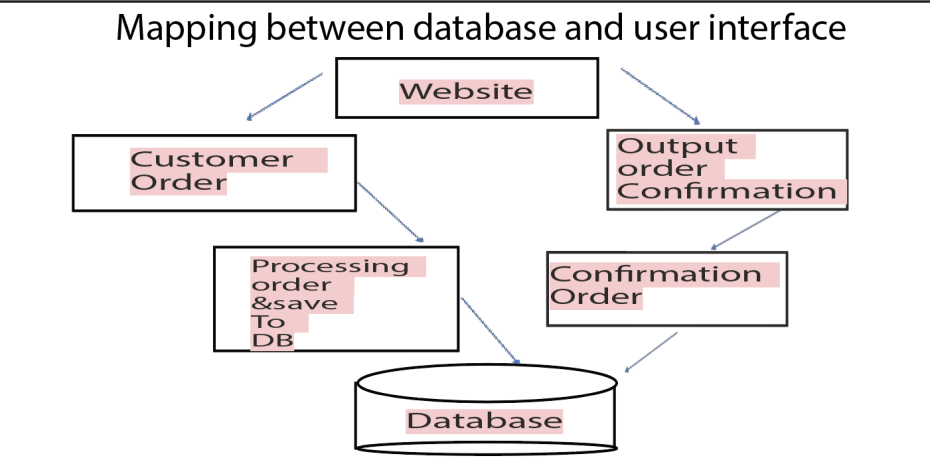
### 9.1 Use Case Diagram

manager

supplier

<<include>>

### 9.2 Database UI Interaction Diagram

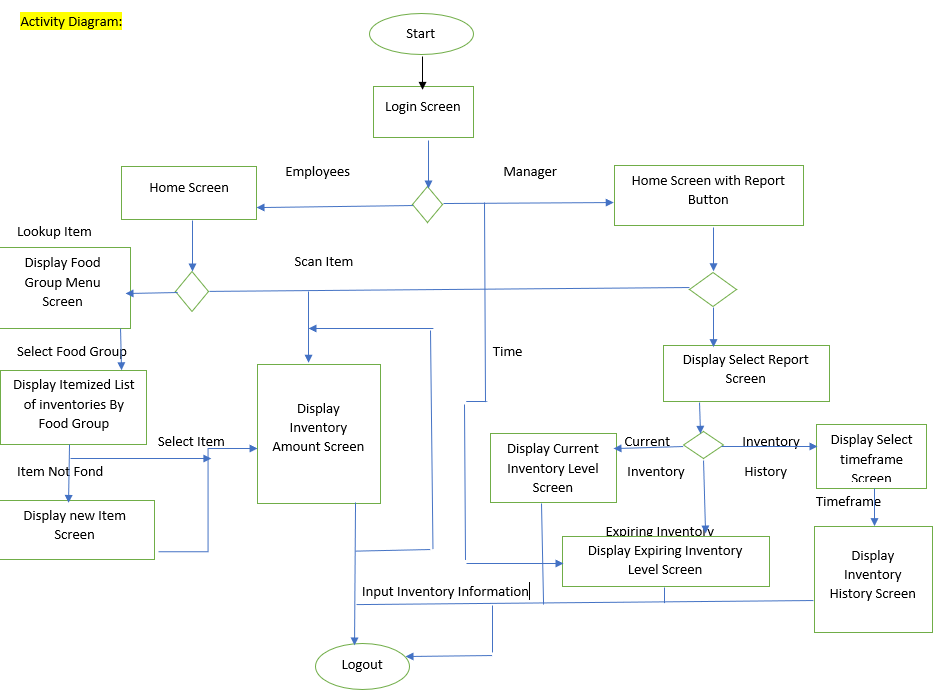


### 9.3 Application Database Interaction Diagram

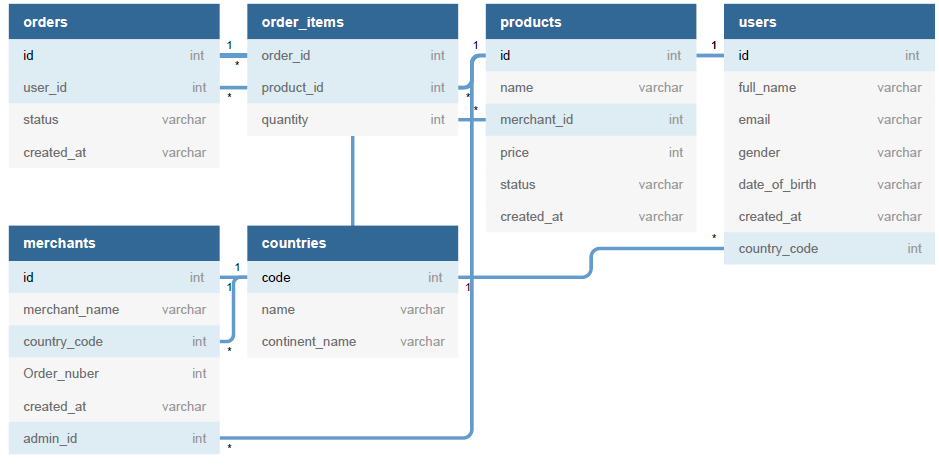
|  |
| --- |
| Execution Environment Execution Environment  Python MySQL  Recipe Management  Ingredient Management    Occasion Management  Vendor Management  Database Subsystem    Updates Management  Correction Management    Login Subsystem  Order Management |



### 9.4 Activity Diagram

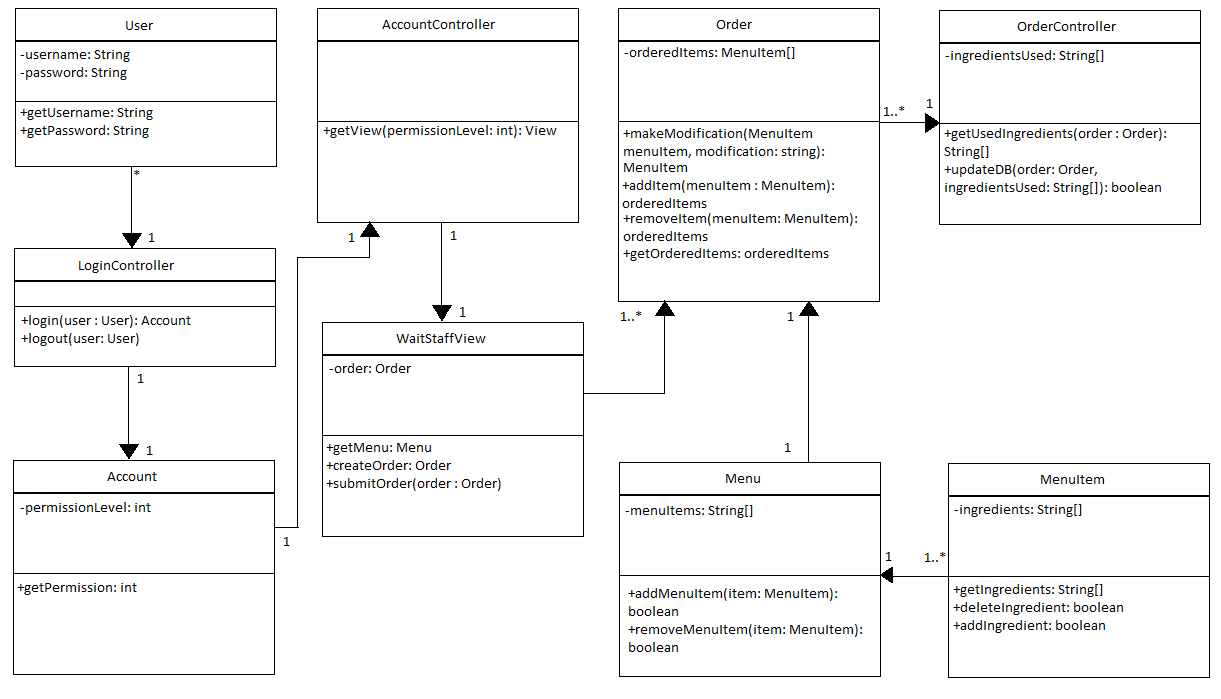


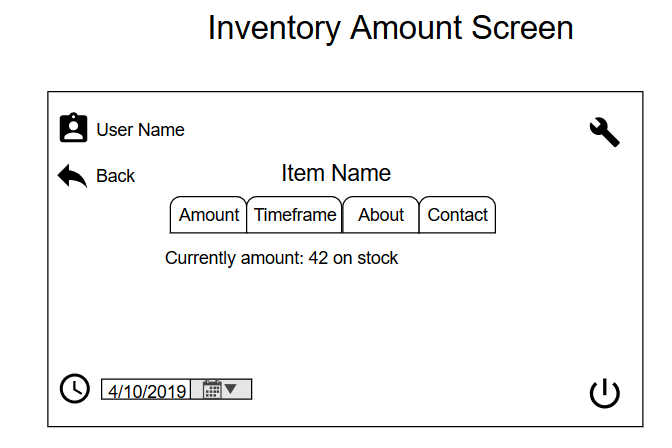
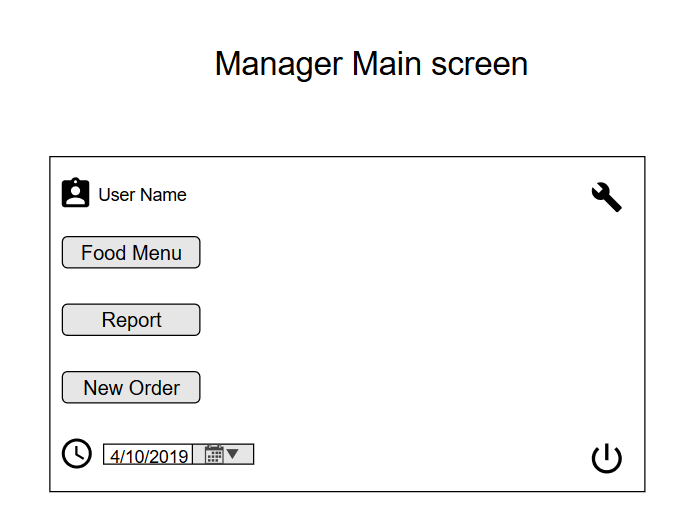
### 9.5 Database Structure Diagram



### 9.6 Suggested WaitStaff Class Diagram

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





1. 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. |

1. 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. |