Point of Sale System (POSS)

Software Requirements Specification

Version 3.3

10/15/2015

Prepared for

CIT/CSE 480 Senior Project

Instructor: Professor Patel

Fall 2015

# Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Iteration** | **Description** | **Author** | **Comments** |
| 09/17/2015 | NA | Version 0 | Mel Chi | Getting Started |
| 09/20/2015 | NA | Version 0.1 | Mel Chi | Edited formatting for better |
| 09/22/2015 | 1 | Version 1.0 | Tim Parzynski | Adding content. Section 2. General Description |
| 09/22/2015 | 1 | Version 1.1 | Mel Chi | Added Functional requirements, Added the basic MySQLUtilities functions |
| 09/24/2015 | 1 | Version 1.2 | Tim Parzynski | Created Use Cases and User Stories 1-3, Iteration1; Section 2. General Description, Product Perspective and Functions |
| 09/24/2015 | 1 | Version 1.4 | Mitch Moore | Create Login, Create Account, Forgot Password, and basic index html pages added along with javascript files to connect front-end with the java servlets |
| 10/1/2015 | 2 | Version 2.0 | Tim Parzynski | Created Use Cases and User Stories 4-6, Iteration2; UI prototypes added to Section 3.3.3, Back of House Interface |
| 10/8/2015 | 3 | Version 3.0 | Tim Parzynski | Adjusted 2.2 Product functions |
| 10/10/2015 | 3 | Version 3.1 | Tim Parzynski | UI prototypes added to Section 3.3.4, Front of House Interface |
| 10/14/2015 | 3 | Version 3.2 | Tim Parzynski | Created Use Cases and User Stories 7-9, Iteration 3; Adjusted 3.2 Functional Requirements |

**Table of Contents**

**Revision History 2**

**Document Approval 2**

**1.** **Introduction 5**

1.1. Purpose 5

*1.1.1.* *Mel 5*

1.2. Scope 5

1.3. Definitions, Acronyms, and Abbreviations 5

1.4. References 5

1.5. Overview 5

**2.** **General Description 5**

2.1. Product Perspective 6

2.2. Product Functions 6

2.3. User Characteristics 6

2.4. General Constraints 6

2.5. Assumptions and Dependencies 6

**3.** **Specific Requirements 6**

3.1. External Interface Requirements 6

*3.1.1.* *User Interfaces 6*

*3.1.2.* *Hardware Interfaces 6*

*3.1.3.* *Software Interfaces 6*

*3.1.4.* *Communications Interfaces 7*

3.2. Functional Requirements 7

*3.2.1.* *Consumer Interface 7*

*3.2.2.* *Back of House Interface 7*

*3.2.3.* *Front of House Interface 7*

*3.2.4.* *Admin/ Interface 7*

3.3. Use Cases 7

*3.3.1.* *Consumer Interface 7*

*3.3.2.* *Back of House Interface 7*

*3.3.3.* *Front of House Interface 7*

*3.3.4.* *Admin/ Interface 7*

3.4. Classes/Objects 7

*3.4.1.* *MySQLUtility 7*

3.5. Non-Functional Requirements 7

*3.5.1.* *Performance 7*

*3.5.2.* *Reliability 7*

*3.5.3.* *Availability 7*

*3.5.4.* *Security 7*

*3.5.5.* *Maintainability 7*

*3.5.6.* *Portability 7*

3.6. Inverse Requirements 7

3.7. Design Constraints 7

3.8. Logical Database Requirements 7

**4.** **Analysis Models 8**

4.1. Sequence Diagrams 8

4.2. DataFlow Diagrams (DFD) 8

4.3. State-Transition Diagrams (STD) 8

**5.** **Change Management Process 8**

**A.** **Appendices 8**

**6.** **SAMPLE 8**

6.1. SAMPLE 8

*6.1.1.* *SAMPLE 8*

6.2. SAMPLE 8

# Introduction

This Software Requirement Specification (SRS) is of the CSE 480 Point of Service (POS) Software. The POS Software is a complete interface for a restaurant and provides the restaurant owner to provide added services to its patrons.

## Purpose

The SRS is intended for developers to understand all features of the software, and is intended to be used to continuously update the system as needed.

### 

## Scope

The POS Software will include an interface for the hosts and waiters, one for the kitchen, one for the owner and manager, and one for the consumer.

The interface for host and waiters should be able to place orders, process payments,and view currently available seating, while the kitchen interface should allow the kitchen staff to see the orders and let the front of house know when any part of an order is ready. The interface for the owner is all encompassing and will provide the options to edit portions. Lastly the consumer interface will give the consumer information and allow the option to order online.

The back-end is a MySQL database system, while the front-end and back-end both utilize libraries and APIs available to the public.

## Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| POSS | Point of Sale System |
|  |  |

## References

Java ArrayList Class Document, *Java Platform Standard Ed. 7*

<http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html>

Java Collections Class Document, *Java Platform Standard Ed. 7*

<http://docs.oracle.com/javase/7/docs/api/java/util/Collections.html>

Framework7 Documentation

http://www.idangero.us/framework7/docs/#.VgFnvrw0fH4

## Overview

The rest of this SRS contains more specifics as to what this product is, how it is organized and structured. This document is broken up into sections and subsections and these sections are listed in the table of contents.

# General Description

## Product Perspective

The product is an all-inclusive, cloud based restaurant POS system. The system will allow customers to place orders from home and check if the restaurant has any open tables at the moment. The business end will act as a ticket queue for the kitchen, keep track of inventory, and automatically place orders when an item is low. The goal of this project is to take all of the aspects of a restaurant’s POS system, web site, order placement and ticket systems and integrate them. The system will also add useful features to improve the user’s experience, such as table openings, Google Maps integration, and online payment methods.

## Product Functions

The product is comprised of four mobile applications, each with its own user interface and individual functions, with some functional overlaps between applications.

### Admin

The administrator, as the owner or manager of the company or a designated individual, has full access to every aspect of the system. They have administrative privileges over account creation, management, and deletion, and full access to the backend of the application, able to change inventory, access, add, and change menus and items, and view order history.

### Kitchen

The kitchen will be able to view the current order queue, view a history of the queue, view inventory levels, view the menus, and send notifications to the wait staff when a meal is ready.

### Wait staff

The wait staff interface will have the ability to view open seating, assign guests to open seats, clear full seats, view the menu, place orders, and will receive notifications from the kitchen when an order is ready.

### Consumer

The consumer end will allow the user to view open seating, view the menu, place orders, and pay for their meal.

## User Characteristics

### 2.3.1 Enterprise End

**2.3.1.1 Admin**

Basic knowledge of smartphones and mobile applications, specifically the Android operating system, as well as training with the product and experience with inventory management and account management.

**2.3.1.2 Kitchen**

Basic knowledge of smartphones and mobile applications, specifically the Android operating system, as well as training with the product and experience with inventory management.

**2.3.1.3 Wait staff**

Basic knowledge of smartphones and mobile applications, specifically the Android operating system.

### 2.3.2 Consumer

Basic knowledge of smartphones and mobile applications, specifically the Android operating system.

## General Constraints

The product will abide by all requirements listed herein.

* Will operate on standard Android-based platform
* Will be available and operational during business hours
* Will maintain account security through proper security protocols
* Will abide by constraints placed on by the SRS

## Assumptions and Dependencies

* Users have access to an Android-based mobile platform with internet access.
* It is within the restaurant’s operational hours

# Specific Requirements

## External Interface Requirements

### User Interfaces

The Point of Sale system includes 4 interfaces. One each for the consumer, admin/owner, kitchen staff, and the front of the house. Each one of these interfaces have different roles and limitations.

### Hardware Interfaces

The goal is to create an android app for each. However, we may consider the admin/owner to have a standalone program to help with administration details.

### Software Interfaces

The enterprise software will need to be compatible with Android, but, for the consumer, the goal is to allow any system that can connect to a web browser to access.

### Communications Interfaces

## Functional Requirements

### Consumer Interface

###### Order Food

###### Introduction

Using the consumer interface, the consumer has the option to order online. Ordering will also include some features as notifying the person the food is ready.

###### Inputs

Consumer selects the menu item that he/she would like to buy and order. Connect with PayPal for cash handling.

###### Processing

Each order will be a new invoice number.

###### Outputs

An email with details of the order will be sent to the user. It will also go to the user’s history.

###### Error Handling

If the order fails, at least one more attempt to connect will be made, after which an error message will be displayed.

###### View History

###### Introduction

Using the consumer interface, the consumer can see all previous orders.

###### Inputs

User will just select the dates to view between, default will be 3 months.

###### Processing

The application will query the sql database between the 2 dates given.

###### Outputs

The application will show previous orders and the menu choices it picked.

Option to email as well.

###### Error Handling

Timeout if can not find and tell the user to see later.

### Back of House Interface

###### Get Orders

###### Introduction

The Back of House Interface will allow the kitchen staff to see the orders that have been placed. It is up to them to decide how to do it.

###### Inputs

There are no input as it will be the default screen for the kitchen staff. They will get it as orders come in.

###### Processing

The user will swipe away the order that has been completed to send notification to wait staff

###### Outputs

They will swipe up to clear.

###### Error Handling

###### Get Order History

###### Introduction

The Back of House Interface will allow the kitchen staff to see a history of the orders filled.

###### Inputs

There will be a button to press in the sidebar to see the previous orders.

###### Processing

Query database to retrieve order history.

###### Outputs

They will see all of the orders for the day.

###### Error Handling

### Front of House Interface

###### Seating The Restaurant

###### Introduction

The Front of House Interface will allow A hostess or waitress sit a customer.

###### Processing

Change seating availability for tables

* + - * 1. Inputs

The user will click a chart or tables to sit customers at a desired table.

* + - * 1. Outputs

The output will be an updated table chart with empty and taken tables

3.2.3.2. Placing Orders

3.2.3.2.1. Introduction

Front of house will be able to place orders right from the app in a similar manner to the consumer interface.

3.2.3.2.2. Processing

Input order into queue to be made for the chef

3.2.3.2.3 Inputs

The food that is going to be ordered, one item at a time.

3.2.3.2.4 Outputs

A list sent to the Kitchen showing the full order

### Admin/ Interface

###### 3.2.4.1. Create App Layout For Restaurant

3.2.4.1.1. Introduction

Admin will manage the look and functions which users will be allowed to see

3.2.4.1.2. Processing

Create the Layout for rest of app

3.2.4.1.3. Inputs

Account type when staff account is created, color scheme or layout desired

3.2.4.1.4. Outputs

A front end customized to the admin’s liking

3.2.4.2. Manage Staff Accounts

3.2.4.2.1. Introduction

Add, delete and manage staff accounts under the admin’s restaurant

3.2.4.2.2. Processing

make changes to users

3.2.4.2.3. Inputs

Changes that are wanted to be performed along with the account to perform them on

3.2.4.2.4. Outputs

An updated user list

3.2.4.3. Manage UI that interacts with Database

3.2.4.3.1. Introduction

Allowed to manage menu, and other things that may need to be stored in a database from the user interface provided

3.2.4.3.2. Processing

Make changes to database

3.2.4.3.3. Inputs

Changes the admin would like to see made

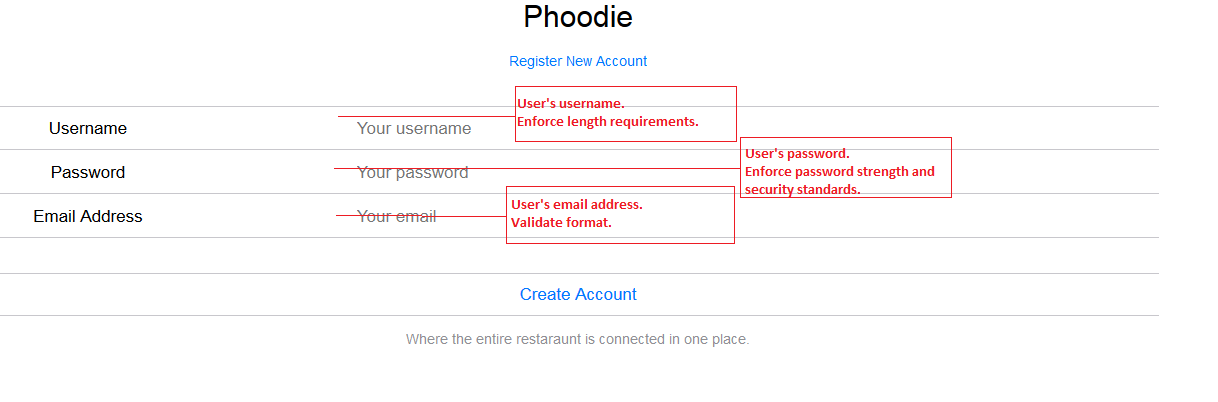
3.2.4.3.4. Outputs

An updated database

## Use Cases

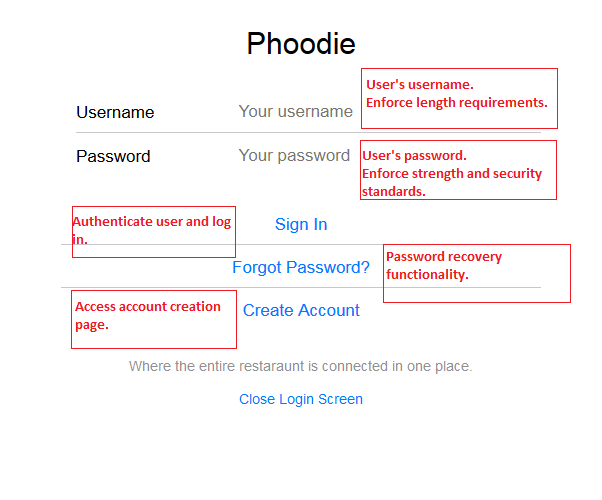
* + 1. Login (Iteration 1)

|  |  |
| --- | --- |
| User Story 1: | As a new user, I want to create an account so I can gain access to all of the application’s content. |



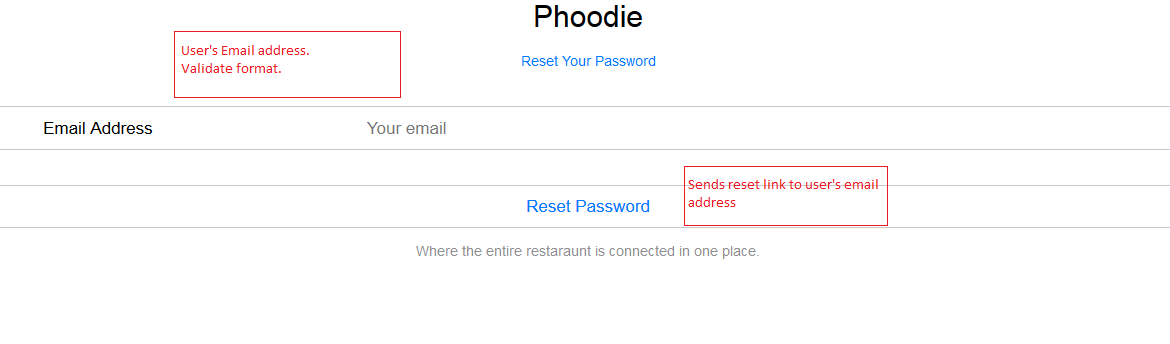
|  |  |
| --- | --- |
| Use Case 1: | Create Account |
| Brief Description: | A user creates a new Phoodie account. |
| Primary Actors: | New User |
| Basic Flow: | |
| 1. User clicks “Create Account” 2. System prompts user for registration information (username, password, email address) 3. User inputs information and clicks “Create Account” | |

|  |  |
| --- | --- |
| User Story 2: | As a new user, I want to login to my account so I can access all of the application’s content. |



|  |  |
| --- | --- |
| Use Case 2: | Login to Account |
| Brief Description: | A user logs in to an existing Phoodie account. |
| Primary Actors: | Existing User |
| Basic Flow: | |
| 1. User clicks “Login” 2. System prompts user for username and password 3. User inputs information and clicks “Sign In” 4. System redirects user to the home page | |

|  |  |
| --- | --- |
| User Story 3: | As a new user, I want to be able to reset my password so, if I forget it, I can recover my account. |



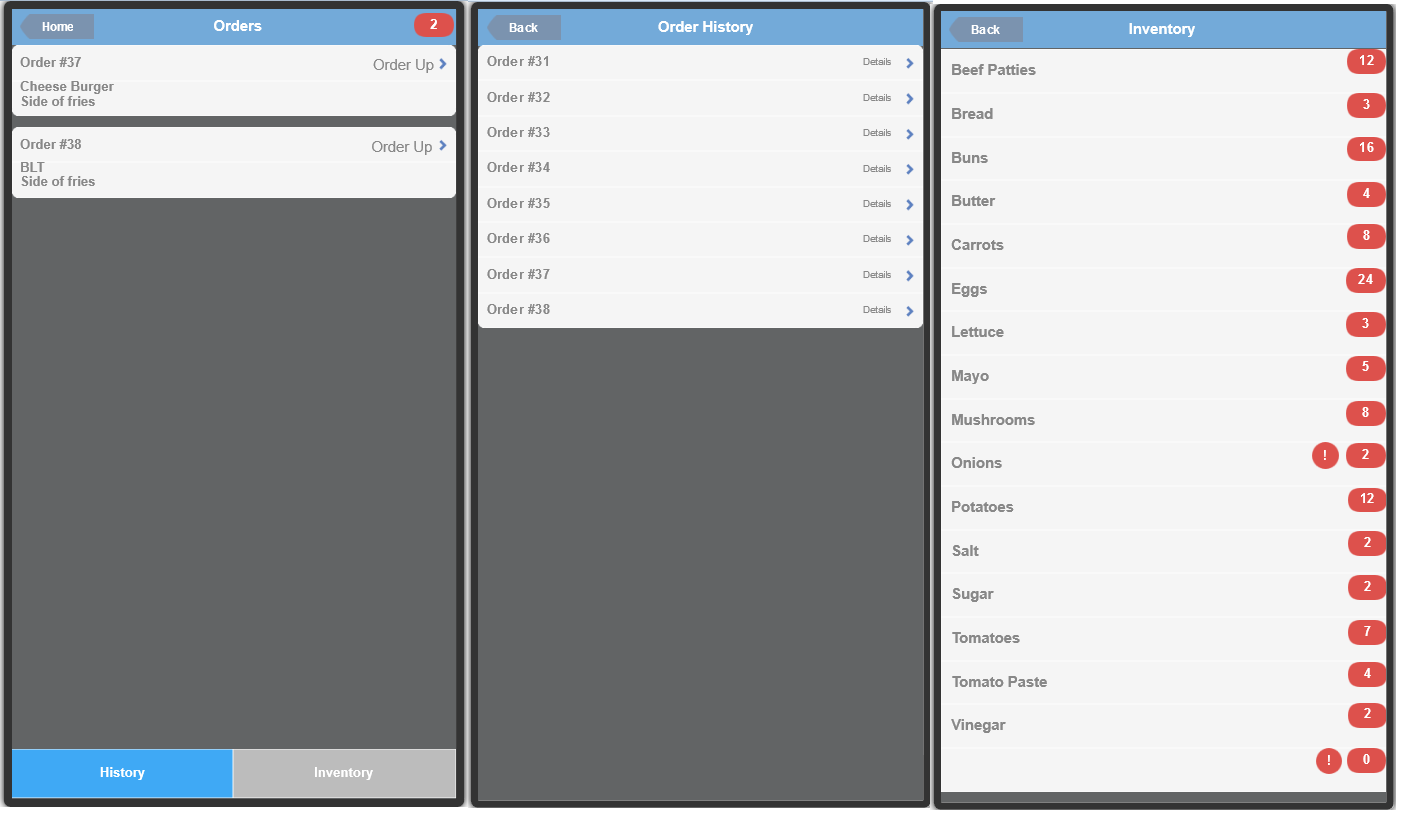
|  |  |
| --- | --- |
| Use Case 3: | Reset Password |
| Brief Description: | A user lost their password and wishes to reset it |
| Primary Actors: | Existing User |
| Basic Flow: | |
| 1. User clicks “Forgot Password?” 2. User is prompted to enter the account’s email address 3. System sends confirmation email to the user with randomly generated password | |

### Consumer Interface

###### 480_user_story.jpg

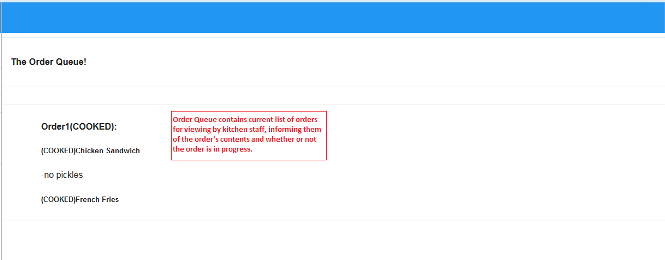
### Back of House Interface

* + - 1. Prototypes

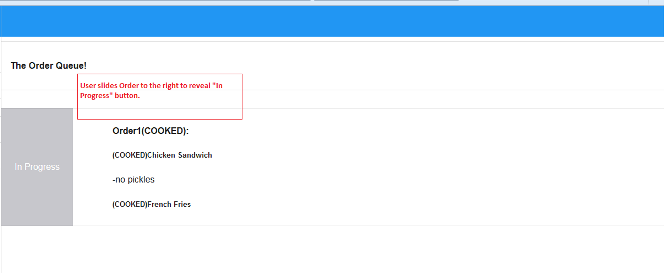


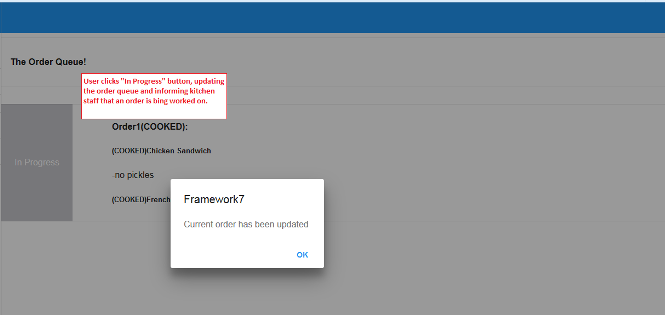
* + - 1. User Stories

|  |  |
| --- | --- |
| User Story 4: | As kitchen staff, I want to be able to view a queue of current orders. |



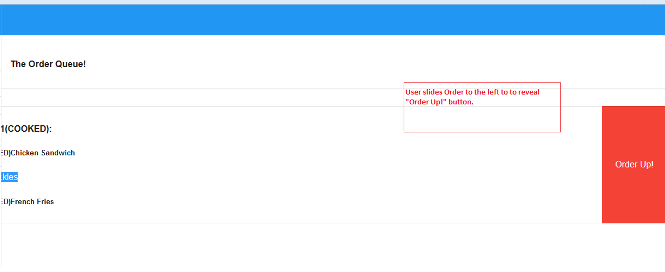
|  |  |
| --- | --- |
| User Story 5: | As kitchen staff, I want to tell the other staff that I am in the process of filling an order, so others do not work on the same order as me. |

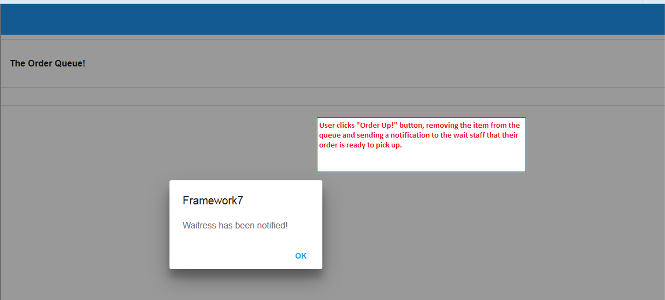




|  |  |
| --- | --- |
| Use Case 5: | Process Order |
| Brief Description: | User marks an order as “In Progress”. |
| Primary Actors: | Kitchen Staff |
| Basic Flow: | |
| 1. User slides Order to the right. 2. System displays “In Progress” button. 3. User clicks “In Progress” button. 4. System updates Order in queue 5. System popup informs user of updated queue. | |

|  |  |
| --- | --- |
| User Story 6: | As kitchen staff, I want to inform the wait staff when an order is ready for pickup. |

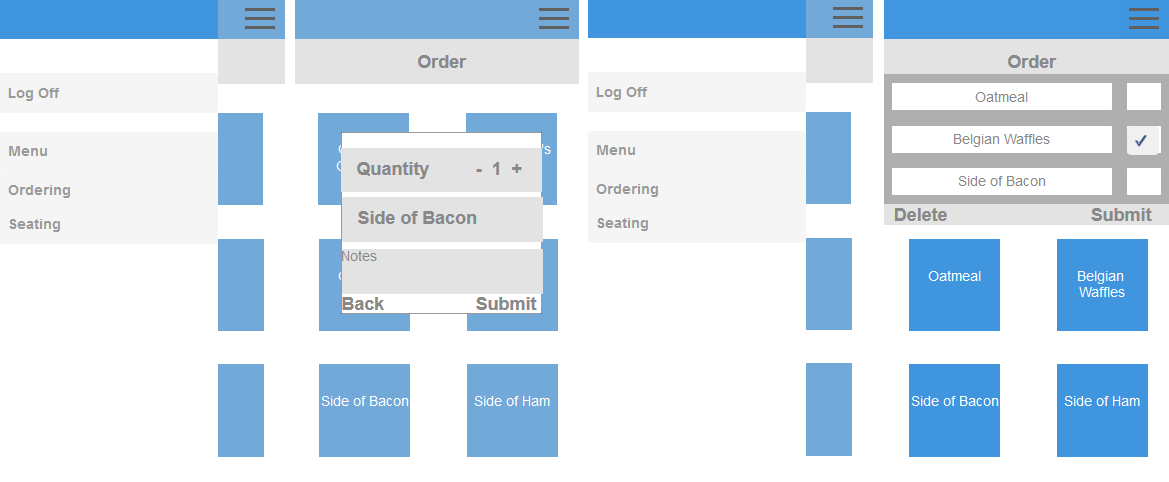




|  |  |
| --- | --- |
| Use Case 6: | Clear Order |
| Brief Description: | User marks an order as “Up” and clears from queue. |
| Primary Actors: | Kitchen Staff |
| Basic Flow: | |
| 1. User slides Order to the left. 2. System displays “Order Up!” button. 3. User clicks “Order Up!” button. 4. System removed Order from queue 5. System popup informs user of updated queue. 6. Future integration with Front of house interface:    1. System sends notification to wait staff that Order is ready for pickup. | |

### Front of House Interface

* + - 1. Prototypes

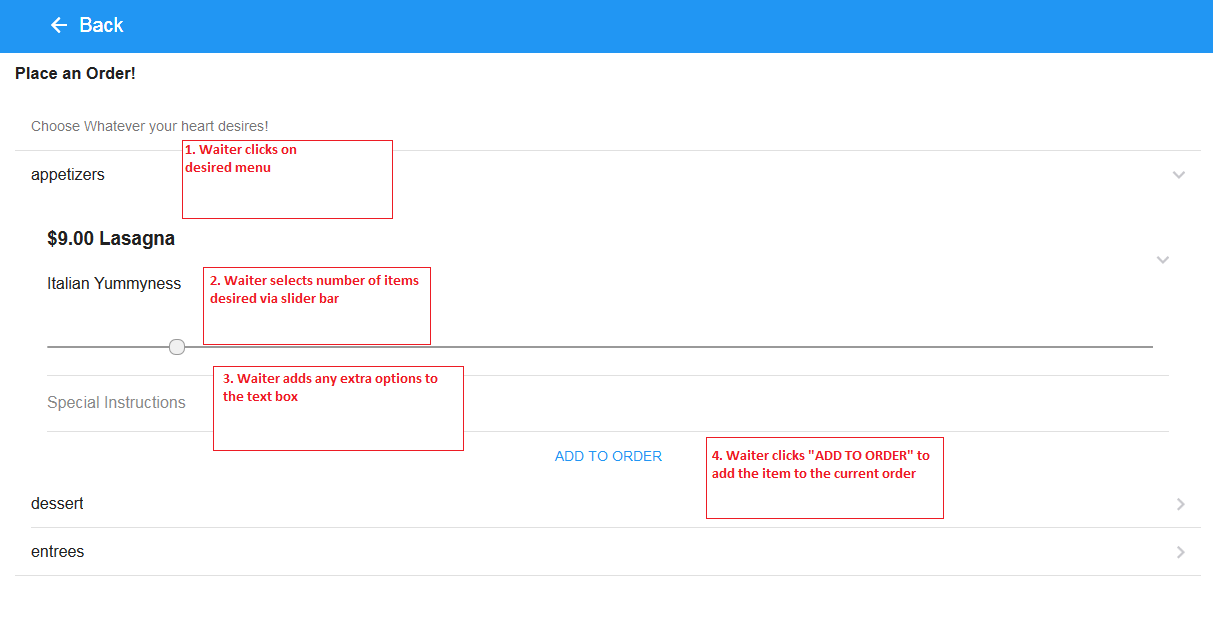


* + - 1. Use Cases

|  |  |
| --- | --- |
| User Story 7: | As a waiter or waitress, I want to be able to view seating and the menu, and place orders. |

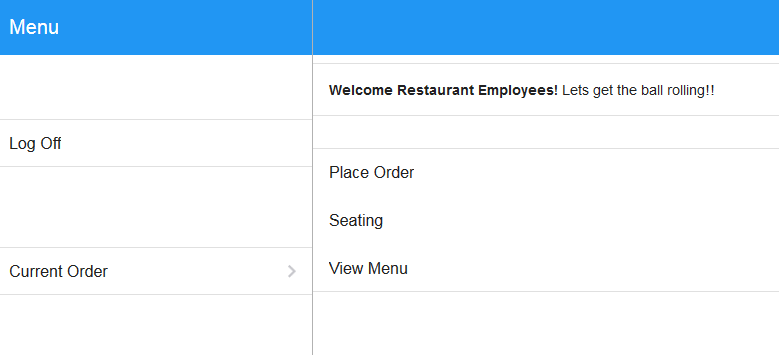
###### Waitress.png

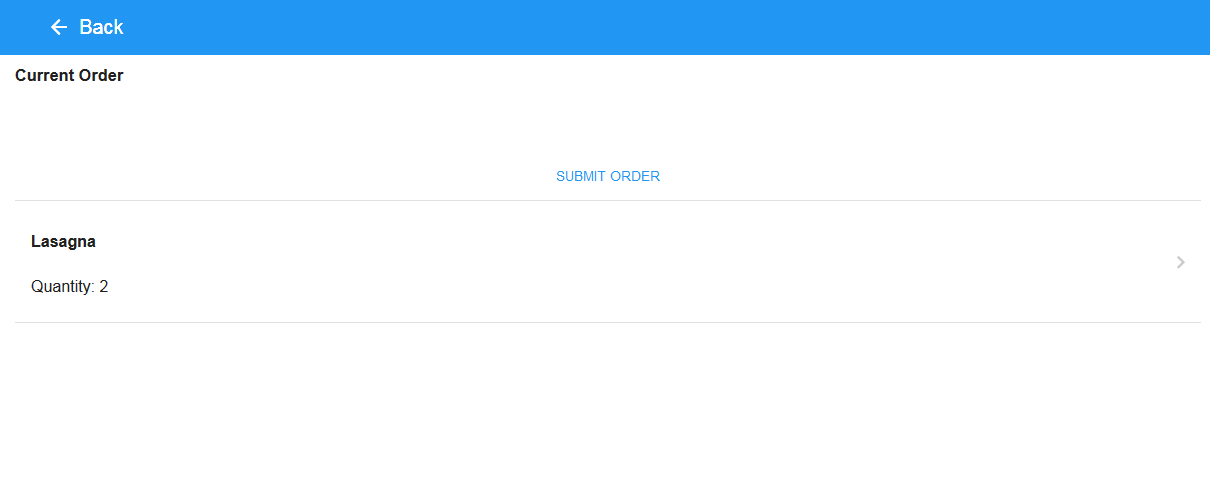
|  |  |
| --- | --- |
| User Story 8: | As a waiter or waitress, I want to be able to add items to an order. |



|  |  |
| --- | --- |
| Use Case 8: | Add Item |
| Brief Description: | Waitstaff adds an item to the current order. |
| Primary Actors: | Waiter/waitress |
| Basic Flow: | |
| 1. User clicks on the desired menu 2. System responds by displaying a list of available items 3. User selects an item 4. User selects the desired number of items via slider bar 5. User adds notes to the “Special Instructions” text box 6. User clicks “ADD TO ORDER” 7. System responds by adding the item(s) to the user’s current order | |

|  |  |
| --- | --- |
| User Story 9: | As a waiter or waitress, I want to be able to send an order to the kitchen |





|  |  |
| --- | --- |
| Use Case 9: | Place Order |
| Brief Description: | Waitstaff sends order to kitchen. |
| Primary Actors: | Waiter/waitress |
| Basic Flow: | |
| 1. User clicks “Current Order” 2. System responds by displaying the “Current Order” page 3. User clicks “SUBMIT ORDER” 4. System responds by sending the order to the kitchen’s order queue | |

### Admin/ Interface

###### Iteration 5

## Classes/Objects

### MySQLUtilities

###### Attributes

The basics of this Utility class is to allow a generic sql class to be used for certain statements

###### Functions

1. InsertSql(sql string) or (string tablename, hashmap <column name, value>
   1. Can take both a insert sql command or Table Name, and values that are listed in a hashmap
2. SelectSQL(sql string)
   1. Takes the SQL string and returns a result set
3. DeleteSQL(sql string)
   1. Takes the SQL string and executes it
4. CreateTableNoRestrictions(TableName, Hashmap<String, Object>
   1. Creates a generic table
5. DropTable(tablename, password)
   1. Drops the tablename given, password to be added on as well to prevent drops

## Non-Functional Requirements

### Performance

The Point of Service system will be available at all times during restaurant operation. During the hours of 10 AM to 1 AM in the morning.

### Reliability

There will be no downtime for the admin, kitchen, and host end during business hours. There will be no downtime for the consumer end with the exception of regular scheduled maintenance during the developmental process.

### Availability

The whole system shall be available during standard business hours. The consumer interface shall be available at all times with the exception of regular scheduled maintenance during the developmental process.

### Security

The system shall not

### Maintainability

Updates and Backups will happen during the non-service hours every night.

### Portability

The product, as a mobile application, will be available wherever service is available.

## Inverse Requirements

## Design Constraints

## Logical Database Requirements

FOR DEVELOPMENT

# *Analysis Models*

## *Sequence Diagrams*

## *DataFlow Diagrams (DFD)*

## *State-Transition Diagrams (STD)*

# Change Management Process

# Appendices

FORMATTING, DO NOT TOUCH BELOW

# SAMPLE

## SAMPLE

### SAMPLE

###### Mel

###### Mel

###### Mel

## SAMPLE