SYSTEMS AND SOFTWARE REQUIREMENTS SPECIFICATION (SSRS) FOR Rub BBQ RMS

Version 1.5
10/2/2010

Prepared for:
Jared Leonard, Owner
RUB BBQ
2407 W Lunt Ave.
Chicago, IL, 60645

Prepared by:
Jerod Hodgkin
Mike Schenk
Jessie Floyd
Scott Leonard
Weber State University
Ogden, UT 84408

Rub BBQ RMS SSRS

RECORD OF CHANGES

Change number	Date completed	Location of change	**A** **M** **D**	Brief description of change	Approved by (initials)	Date Approved
001	27 Sep 2010	1-17	A	First Draft	JTH	27 Sep 2010
002	30 Sept 2010	entire document	M	Formatting	JLF	29 Sept 2010
003	01 Oct 2010	3.2	A	Added use cases 2-7	SWL	01 Oct 2010
004	01 Oct 2010	3.1.3	A	Pasted in the Prototype Screen Shots	JTH	01 Oct 2010
005	01 Oct 2010	entire document	M	Edited for spelling, grammar and formatting	MS	01 Oct 2010

*__A__ - ADDED __M__ - MODIFIED __D__ - DELETED

Rub BBQ RMS SSRS

TABLE OF CONTENTS

1. INTRODUCTION

Jared Leonard owns a barbecue restaurant in Chicago, IL called Rub BBQ. As his restaurant grows he needs to automate some of the routine management functions of ordering food supplies. He also needs some way to track inventory, take payment and report on sales and inventory. Jared's key concern is to leverage technology to help him identify trends in sales and simply the task of ordering the correct items at the correct time.

1.1 IDENTIFICATION

The software system being considered for development is referred to as a Restaurant Management System (RMS). As this is the first instance of the product its version number will be 1.0 at release.

Jared Leonard, owner of Rub BBQ 2407 W Lunt Ave. Chicago, IL, is the principal customer Professor Ted Cowan is the student advisor for the project.
Jared has several employees who assist in the day-to-day functions of the business. These personnel are also customers of BBQ RMS, or users of the software.

1.2 PURPOSE

The system under development is proposed to assist in inventory management, product ordering, sales tracing, and profit/revenue projections. Jared Leonard, Ted Cowan and the development staff (Jerod Hodgkin, Mike Schenk, Jessie Floyd, and Scott Leonard) have input to this document. This document is intended to provide a basis for expectations and deliverables. It is managed by the development staff with input from the principals (Jared Leonard, Ted Cowan).

1.3 SCOPE

Primary development efforts shall be focused on the the following areas:

- Inventory management
- Customer orders
- integration with a 3rd party payment system
- menu breakdown
- simple reporting

This system will not provide:

- a payment processing system
- support for 3rd party application integration
- detailed accounting

This project is a new design and as such has not yet created a history of previous efforts. Operations or “sustainment” efforts are not scheduled into the project. All additional software modifications will be contracted on an “as-needed” basis and managed via the project charter.

1.4 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Term or Acronym	Definition
Alpha test	Limited release(s) to selected, outside testers
Beta test	Limited release(s) to cooperating customers wanting early access to developing systems
Final test	aka, Acceptance test, release of full functionality to customer for approval
DFD	Data Flow Diagram
SDD	Software Design Document, aka SDS, Software Design Specification
SRS	Software Requirements Specification
SSRS	System and Software Requirements Specification
RMS	Restaurant Management System
POS	Point of Sale
ERD	Entity Relationship Diagram
WPF	Windows Presentation Foundation
WCF	Windows Communication Foundation
RDBMS	Relational Database Management System
GSC-SVN	Google Source Control - subversion
WBS	Work Break-down Structure (MS project file)
Use case	A diagram and description of a certain function of the system.
GUI	Graphical user interface

1.5 REFERENCES

The development staff will maintain revisions of additional documentation via GSC-SVN. Changes will be checked in and maintained there. The following documents will be available for

review upon request:

RMS Project Charter
WBS
Project Proposal

1.6 OVERVIEW AND RESTRICTIONS

This document is for limited release only to Weber CS personnel working on the project Rub BBQ RMS and Jared Leonard.

Section 2 of this document describes the system under development from a holistic point of view. Functions, characteristics, constraints, assumptions, dependencies, and overall requirements are defined from the system-level perspective.

Section 3 of this document describes the specific requirements of the system being developed. Interfaces, features, and specific requirements are enumerated and described to a degree sufficient for a knowledgeable designer or coder to begin crafting an architectural solution to the proposed system.

Section 4 provides the requirements traceability information for the project. Each feature of the system is indexed by the SSRS requirement number and linked to its SDD and test references.

Sections 5 and up are appendices including original information and communications used to create this document.

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

Rub-BBQ RMS is purposed to be a multi-tier application with its core component as the reporting/inventory management component. Additional software components include an order entry application, cook review screen and a reporting application to be used by the manger. Each component interfaces with a middle-tier running on the managers workstation. These components are designed to solely inter-operate with software designed and developed by Rub-BBQ RMS developers only.

2.2 PRODUCT FUNCTIONS

The application will provide comprehensive **inventory management** functionality including the following:

- Taking periodic inventory
- Managing inventory item details
- Inventory reporting

The application will provide a **customer order interface** capable of:

- Taking orders
- Calculating totals, taxes and gratuity

- Printing receipts

The application will provide a **cook's interface** that will display customer orders listed side by side in the order that they were received. As the meal preparation is completed the cooks can drop the meal off the list and move on to the next order.

2.3 USER CHARACTERISTICS

The Owner will have full access to the entire application without restriction. This includes the touchscreen interfaces and detailed managerial tasks.

The Employees will operate a touchscreen interface for order entry, taking payment, taking inventory, and the cook's screen. All of the employees in this particular restaurant will fill the functions of server, host, cook, and cashier.

2.4 CONSTRAINTS

All components will deploy and operate on Windows® based operating environments on commodity hardware.

2.5 ASSUMPTIONS AND DEPENDENCIES

The software application will be deployed on a computer running a Windows® operating system with .NET 3.5 or greater APIs, with network access. Additional software can be installed by the user as required. Technical support is available as needed.

The system functions will depend on the availability of a credit card scanner, card payment processing system, a printer, and network access.

2.6 SYSTEM LEVEL (NON-FUNCTIONAL) REQUIREMENTS

2.6.1 Site dependencies

Foundational requirements: Jared will provide a computer system running Windows® 7.
From this baseline we will install additional software components as needed. These include Microsoft .NET framework, a network to support data transfer from hand-held devices to the central computer system having a Microsoft SQL Server® installation. Further dependencies will continue to be refined as the project matures. Additional dependencies might include a credit card scanner or card payment processing system, and a printer.

2.6.2 Safety, security and privacy requirements

The system shall not record or track any personal information beyond simple employee information used to log in to the system. Each user will have a log-on so transactions can be tracked on a user level. Any credit card or other payment card payments that are accepted must be done according to industry standards.

2.6.3 Performance requirements

The system should be fast enough to be used as a real time POS interface to process customer invoices. Customer satisfaction will be determined upon release of the first beta code release, performance analysis will be assessed at that time.

2.6.4 System and software quality

Industry standard programing and coding practices will be followed as closely as we are able. Blatant disregard for robustness and stability will not be tolerated by our development effort. A “code review” process will be adhered to, which is: each module will be reviewed by a second developer to ensure a high standard of coding.

2.6.5 Packaging and delivery requirements

The executable system and all associated documentation (i.e., SSRS, SDD, code listing, test plan including data and results, and user manual) will be delivered to the customer on CDs and/or via email, as specified by the customer at time of delivery. Although document “drops” will occur throughout the system development process, the final, edited version of the above documents will accompany the final, accepted version of the executable system.

2.6.6 Personnel-related requirements

The system under development has no special personnel-related characteristics.

2.6.7 Training-related requirements

No training materials or expectations are tied to this project other than the limited help screens built into the software and the accompanying user manual.

2.6.8 Logistics-related requirements

Implementation and training will impact the flow of the business. The customer is currently using a POS system that will be replaced by the BBQ RMS.

Software will be installed on preexisting hardware. All software will be installed and configured remotely. Existing equipment will be used in the RUB BBQ RMS.

The system will be designed with a tiered architecture in mind, but will be able to run the data and user interface layers on a single machine.

2.6.9 Other requirements

TBD

2.6.10 Precedence and criticality of requirements

1) Inventory management:	high
2) Manager’s management interface:	high
3) Processing customer orders:	medium
4) Inventory update:	low
5) Simple reporting:	medium/high
6) Menu break-down:	low
7) Cook’s interface:	low

8) Credit-card processing service: low

3. SPECIFIC REQUIREMENTS

3.1 EXTERNAL INTERFACE REQUIREMENTS

3.1.1 Hardware Interfaces

The system shall be operated from a touch-sensitive laptop or tablet computer supporting 1280x1024 resolution.

The system may utilize a credit card scanner.

The system may interface with a cash drawer.

The system shall print to a receipt printer.

The system shall make use of standard TCP/IP networking hardware.

3.1.2 Software Interfaces

The system shall interface with a Microsoft SQL Server® RDBMS to maintain inventory and track sales.

The system shall interface with a third-party credit card payment processor via a web or web services interface.

Server's order entry:

Input: Touch screen to record order and send to web middle tier.

Output: send a message via web services to the server to record order.

Cook's display

Input: Touch screen to recall queued orders.

Output: send a message via web services to the server to change status of order.

Manager's Workstation

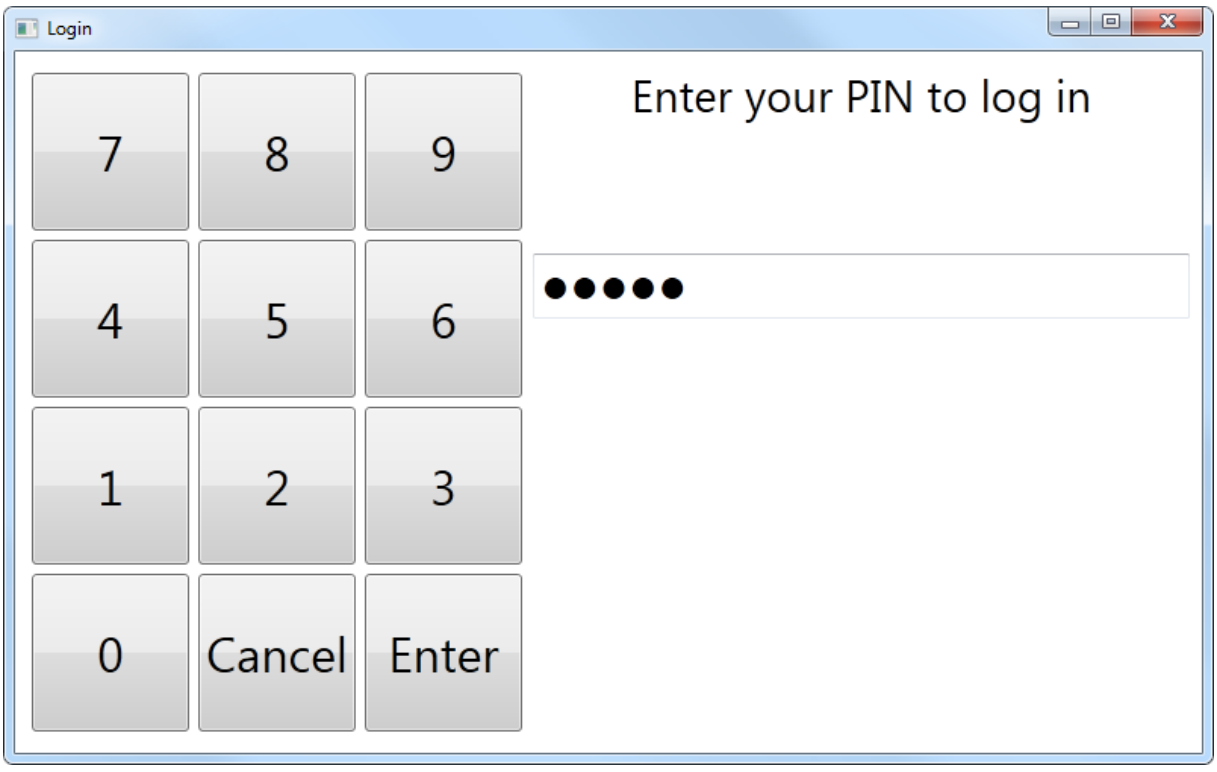
Input: (traditional computer) keyboard, display, mouse

Output: display and formatted, printed reports.

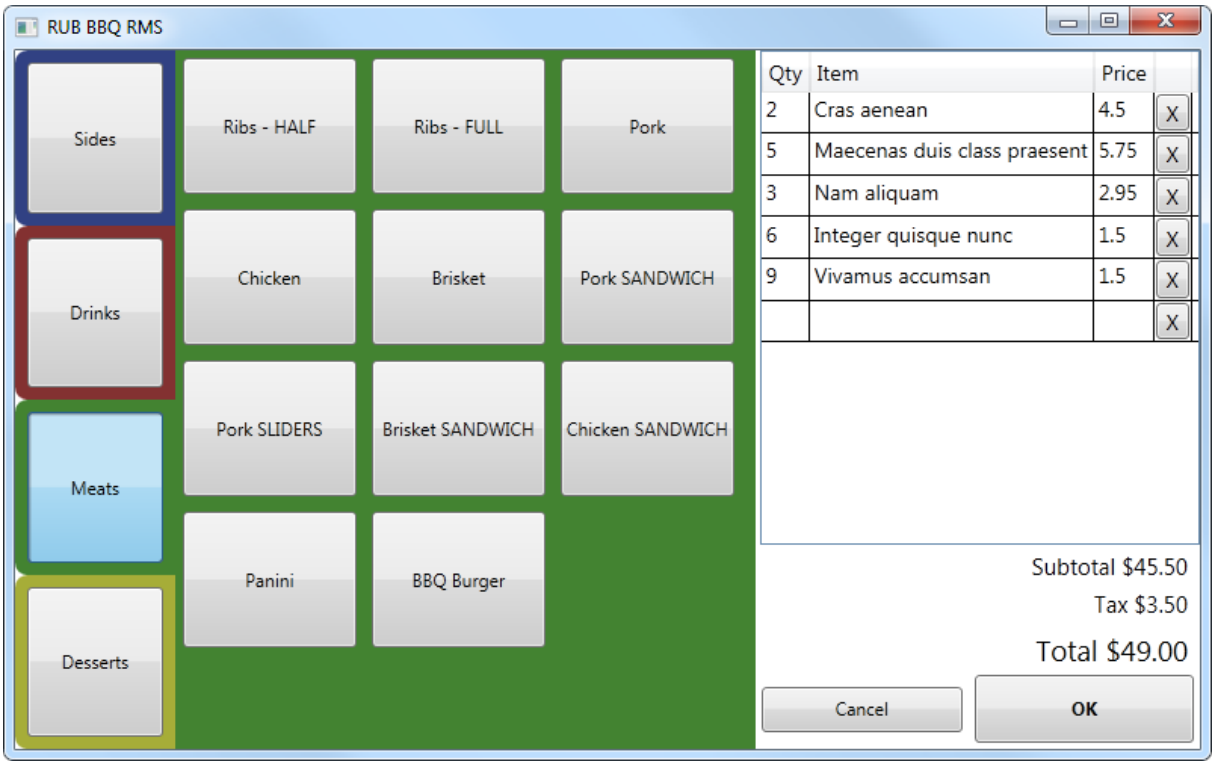
3.1.3 User Interfaces

First Draft Prototypes

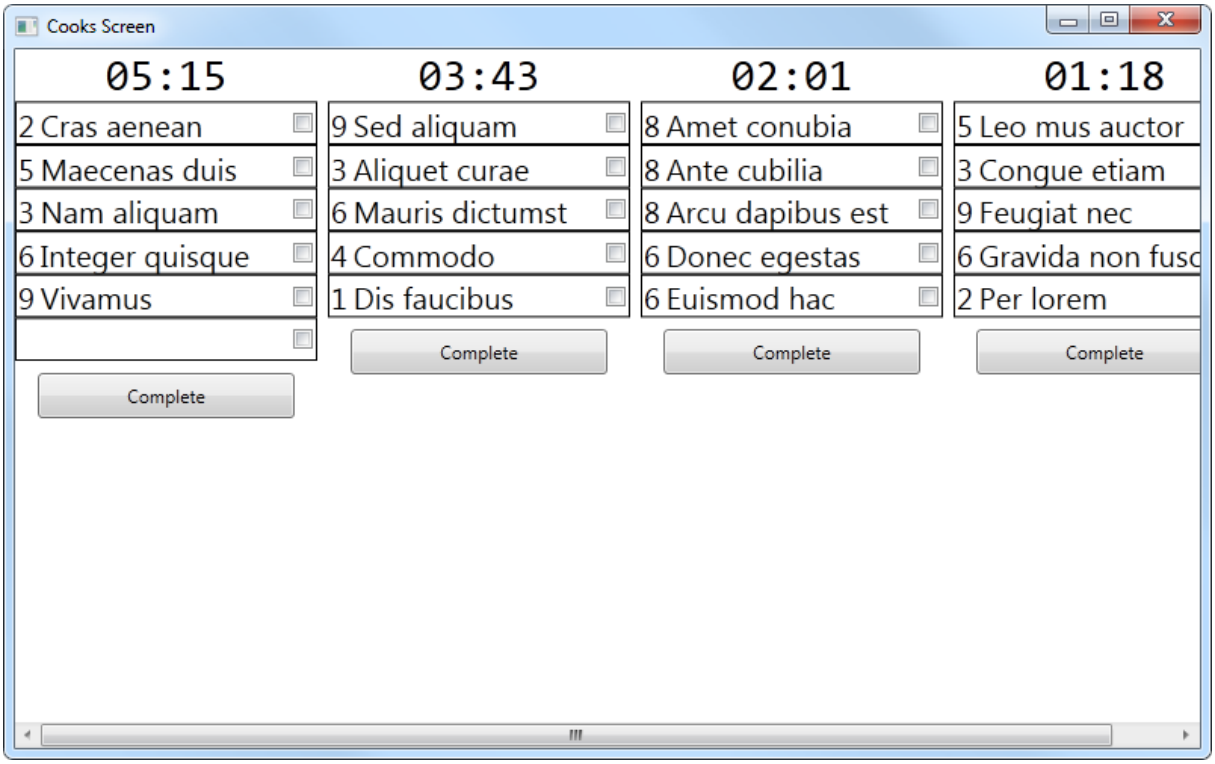
Login Screen



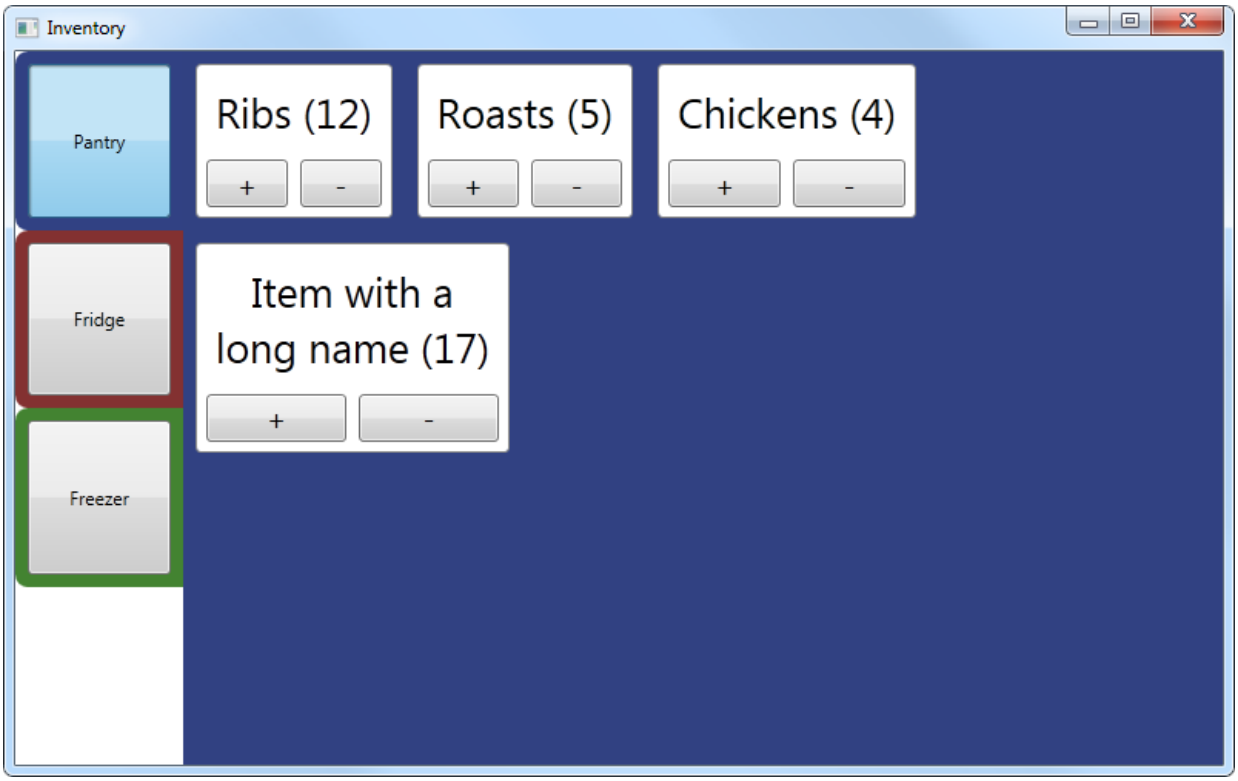
Order Entry Screen



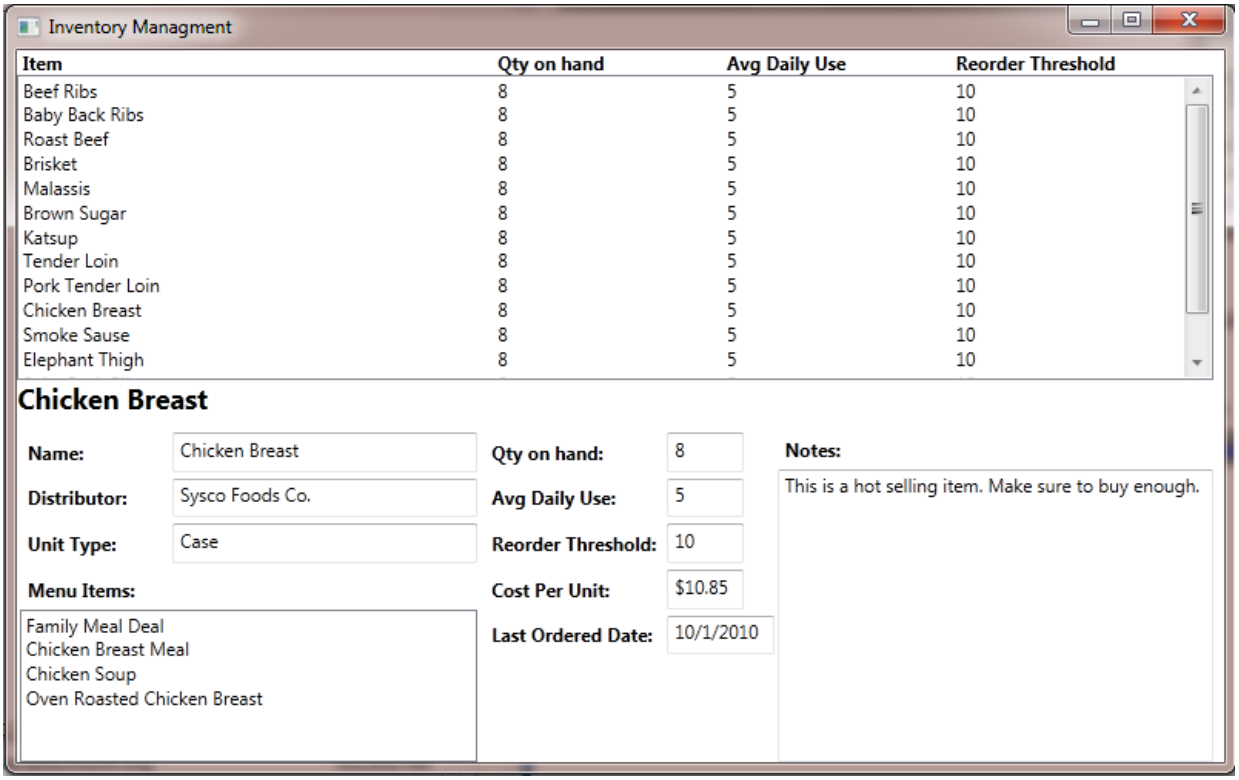
Cook's Screen



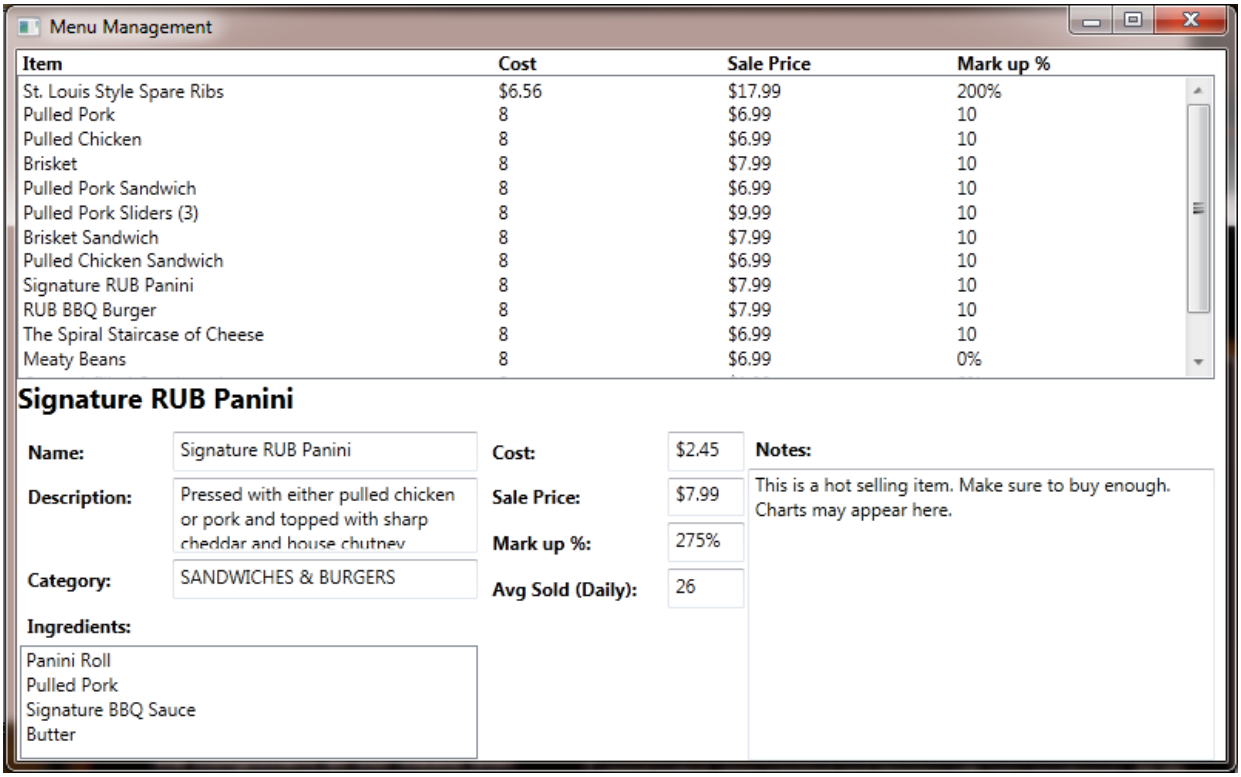
Inventory Management (Touch Screen)



Inventory Management



Menu Management

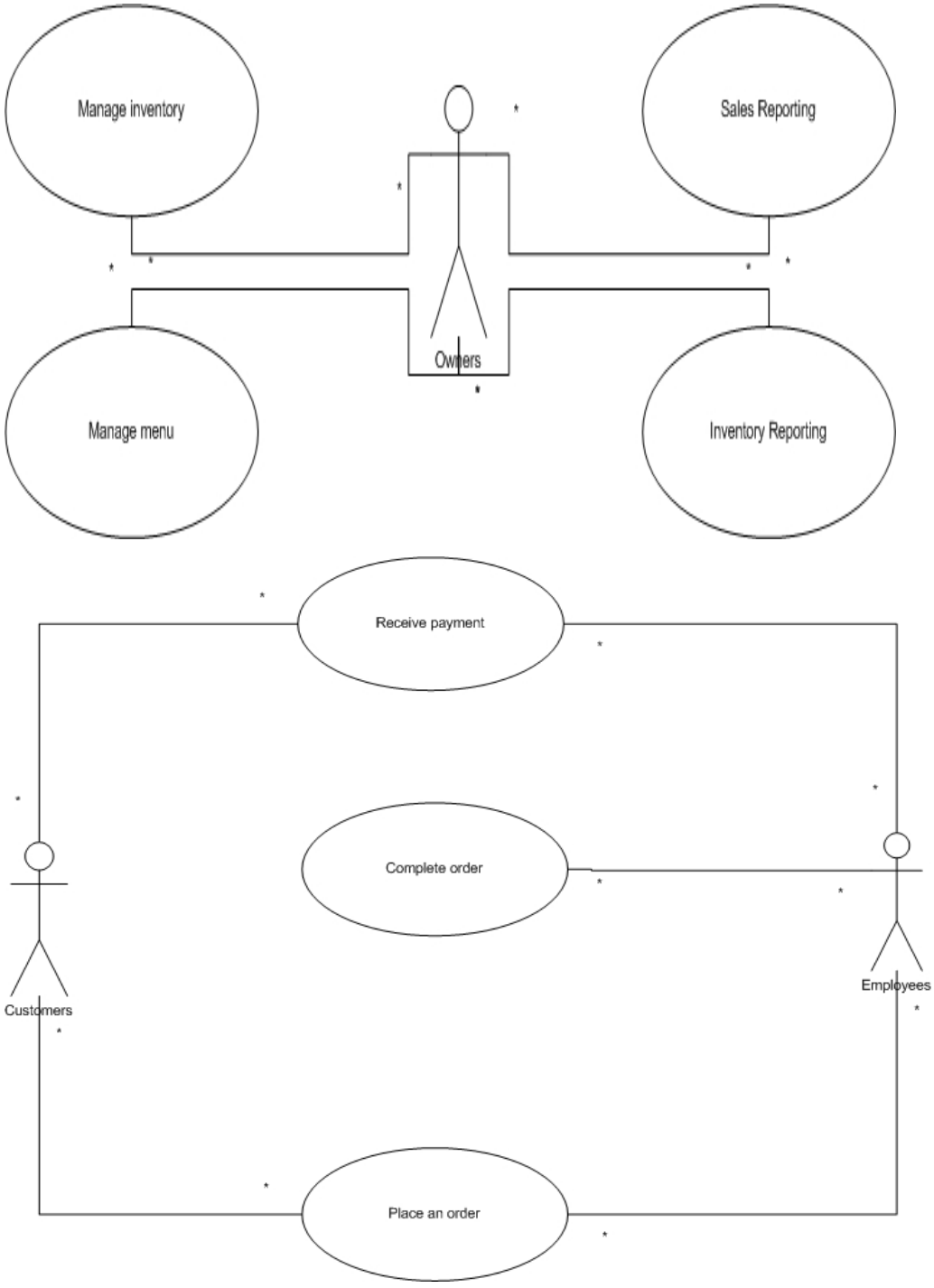


3.1.4 Other Communication Interfaces

None at this time

3.2 SYSTEM FEATURES

3.2.1 Use Case Diagrams



3.2.2 System feature 1: Inventory Management

Use Case Description	Non-task feature description
Inventory management **Actors:** Owners **Goals:** Keep count of each item in inventory. Allow the client to adjust inventory as needed. **Preconditions:** We have not identified the preconditions. **Summary:** The owners should be able to track the use of inventory through the day and make adjustments to inventory when a physical inventory count is taken. **Related use cases:** Inventory reporting Complete order **Steps** We have not yet created user stories for the use cases. We need information from our client **Alternatives** There are no alternatives in this use case. **Postconditions** The client will have an accurate count of their inventory.	**Introduction/Purpose of this feature** The inventory management feature shall allow the owners to track their inventory. Inventory management shall keep a count of all inventory, and approximate the quantity of certain ingredients that do not have exact measurements. **Input/Output sequence for this feature** The client shall enter an initial inventory count when they go live on the system. The client shall reduce the inventory by an appropriate quantity as inventory items are used to make menu items. The client shall take a regular physical inventory to reconcile the difference between the system inventory count and the actual inventory count. **Design constraints of this feature** The inventory shall be stored in the database. The database shall exist such that the inventory items and quantities shall be available in other areas of the system. Reporting, ordering, and the menu shall access the inventory as needed. **Performance requirements of this feature** Inventory management shall be able to store up to 10,000 inventory items. The inventory screen shall take less than 10 seconds to load when all inventory items are listed. **Detailed functional requirements of this feature** We have not gathered detailed functional requirements at this time. **Functional requirement 1.1** We have not gathered functional requirements - this is a placeholder.

3.2.3 System feature 2: Inventory Reporting

Use Case Description	Non-task feature description
Inventory Reporting **Actors:** Owners **Goals:** See current inventory Track how inventory is used **Preconditions:** There exists an inventory with some history of inventory transactions. **Summary:** The owners should be able to look at reports that describe how their inventory is being used. This may include seeing how much inventory is used each day and what items are used in certain recipes. **Related use cases:** Inventory management **Steps** We have not yet created user stories for the use cases. We need information from our client **Alternatives** There are no alternatives in this use case. **Postconditions** NA	**Introduction/Purpose of this feature** The inventory reporting feature shall allow the owners to generate reports related to store inventory. **Input/Output sequence for this feature** The inventory shall be viewed from the database. The reporting feature shall read the database and shall output data in a readable and attractive format. **Design constraints of this feature** Inventory management shall read information about the inventory from the database. Inventory management shall not make any changes to the inventory. **Performance requirements of this feature** No report shall take more than 120 seconds to load. **Detailed functional requirements of this feature** We have not gathered detailed functional requirements at this time. **Functional requirement 1.1** We have not gathered functional requirements - this is a placeholder.

3.2.4 System feature 3: Menu Management

Use case description	Non-task feature description
Menu Management **Actors:** Owners **Goals:**	**Introduction/Purpose of this feature** The menu management feature shall allow the client to create menu items to be sold to their customers. **Input/Output sequence for this feature**

The owners should be able to create new menu items from existing inventory items and change details of the current menu items including, recipe, description and sale price. **Preconditions:** There exist items in the inventory with which to construct menu items. **Summary:** The menu items will be the items sold to the the customers. Menu items are constructed of inventory items **Related use cases:** Inventory management Place order Complete order **Steps** We have not yet created user stories for the use cases. We need information from our client **Alternatives** There are no alternatives in this use case. **Postconditions** There exists a menu item that can be sold to a customer.	The inventory items shall be read from the database and be combined to create a recipe that is associated with a menu item. When a menu item is built, there shall be 1 menu item created and all the inventory items shall be reduced by an appropriate amount. **Design constraints of this feature** Menu management shall read the inventory to construct menu items. Menu items shall be available to be added to an order. **Performance requirements of this feature** No performance requirements have been identified. **Detailed functional requirements of this feature** We have not gathered detailed functional requirements at this time. **Functional requirement 1.1** We have not gathered functional requirements - this is a placeholder.

3.2.5 System feature 4: Place an order

Use case description	Non-task feature description
Place an order **Actors:** Customers Employees **Goals:** Customers and employees should be able to place orders of food items. **Preconditions:** There exist menu items that can be sold on the	**Introduction/Purpose of this feature** Customers shall place orders through this feature so the food can be prepared and eaten and payment can be received. **Input/Output sequence for this feature** Menu items shall be added to an order. The order shall be passed to the cook so the food can be prepared. **Design constraints of this feature** Place order shall read menu items in the

order. **Summary:** Customers shall be able to place orders on their own, and employees should be able to place orders on behalf of customers. **Related use cases:** Menu management Complete order Receive payment **Steps** We have not yet created user stories for the use cases. We need information from our client **Alternatives** There are no alternatives in this use case. **Postconditions** An order will exist in the system that can be completed.	database so an order can be created. Place order shall save orders in the database. Place order shall do some simple calculations to determine order price based partly on menu item price. Several people shall be able to place an order simultaneously. **Performance requirements of this feature** This feature shall be in continuous use when the restaurant is open. This feature shall run without slowing down over long periods of time. As more orders are placed the performance of this feature shall not be impacted. As menu items are added to an order the performance of this feature shall not be impacted. **Detailed functional requirements of this feature** We have not gathered detailed functional requirements at this time. **Functional requirement 1.1** We have not gathered functional requirements - this is a placeholder.

3.2.6 System feature 5: Complete order

Use case description	Non-task feature description
Complete order **Actors:** Employees **Goals:** Employees in the kitchen should receive orders and complete them by preparing the menu items on the orders. **Preconditions:** There exist orders that have not been completed or paid for. **Summary:** After an order is placed, it will show up on the	**Introduction/Purpose of this feature** Complete order shall allow the employees in the kitchen to receive orders that have been placed and mark the order complete once the items have been prepared. **Input/Output sequence for this feature** Orders shall be visible to the complete order feature through the database. Once an order is marked completed it shall be available to be paid for. **Design constraints of this feature** Complete order shall read from the database all the orders that have been placed. More than one employee shall be able to view the active orders

cook screen. An employee in the kitchen will prepare the items on the order and mark the order completed. **Related use cases:** Menu management Place order Receive payment **Steps** We have not yet created user stories for the use cases. We need information from our client **Alternatives** There are no alternatives in this use case. **Postconditions** An order will exist in the system that can be paid for.	at once. **Performance requirements of this feature** This feature shall be capable of showing new orders in near real time. A large number of orders shallnot impact the performance of this feature. **Detailed functional requirements of this feature** We have not gathered detailed functional requirements at this time. **Functional requirement 1.1** We have not gathered functional requirements - this is a placeholder.

3.2.7 System feature 6: Receive payment

Use case description	Non-task feature description
Complete order **Actors:** Employees Customers **Goals:** Employees at the cash register should receive completed orders and take payment. **Preconditions:** There exist orders that have been completed but not paid for. **Summary:** After an order is placed, it will show up on the cook screen. An employee in the kitchen will prepare the items on the order and mark the order completed. These orders will show up on the cashier screen so the employee can ring them up and receive payment for the order.	**Introduction/Purpose of this feature** Receive payment is the cashiering function of the program. It allows employees to take a completed order and take payment on it. This finally finishes the order and places it into sales history. **Input/Output sequence for this feature** Completed orders shall be viewed in receive payment. Orders shall be paid and go to history. Orders for which payment is not received shall be marked unpaid at the end of the day, and shall still be a part of the history. **Design constraints of this feature** Receive payment shall read completed orders from the database and see the price of menu items so the invoice can be totalled. Receive payment shall be able to add tax and tip to the total. **Performance requirements of this feature**

Related use cases: Menu management Place order Complete order Sales reporting **Steps** We have not yet created user stories for the use cases. We need information from our client **Alternatives** There are no alternatives in this use case. **Postconditions** A completed paid order will show up in the order history.	This feature shall be capable of showing completed orders in near real time. A large number of orders shall not impact the performance of this feature. **Detailed functional requirements of this feature** We have not gathered detailed functional requirements at this time. **Functional requirement 1.1** We have not gathered functional requirements - this is a placeholder.

3.2.8 System feature 7: Sales Reporting

Use case description	Non-task feature description
Complete order **Actors:** Owners **Goals:** Owners should be able to view details about sales as they happen through the day. They should be able to see a sales breakdown by hour, day, month, and year. **Preconditions:** There exist orders that have been paid. **Summary:** When an order has been marked paid or not paid, it will be saved in history to be counted and reported on. Reporting will allow the owners to track their sales. **Related use cases:** Receive payment Inventory reporting **Steps** We have not yet created user stories for the use	**Introduction/Purpose of this feature** Sales reporting shall give the owners the ability to see information about their sales history. **Input/Output sequence for this feature** The orders that have been marked paid or not paid shall show up in sales reports. The client shall be able to view or print reports. **Design constraints of this feature** Sales reporting shall read sales history from the database. **Performance requirements of this feature** No sales report shall take longer than 120 seconds to load. **Detailed functional requirements of this feature** We have not gathered detailed functional requirements at this time. **Functional requirement 1.1** We have not gathered functional requirements -

cases. We need information from our client **Alternatives** There are no alternatives in this use case. **Postconditions** NA	this is a placeholder.