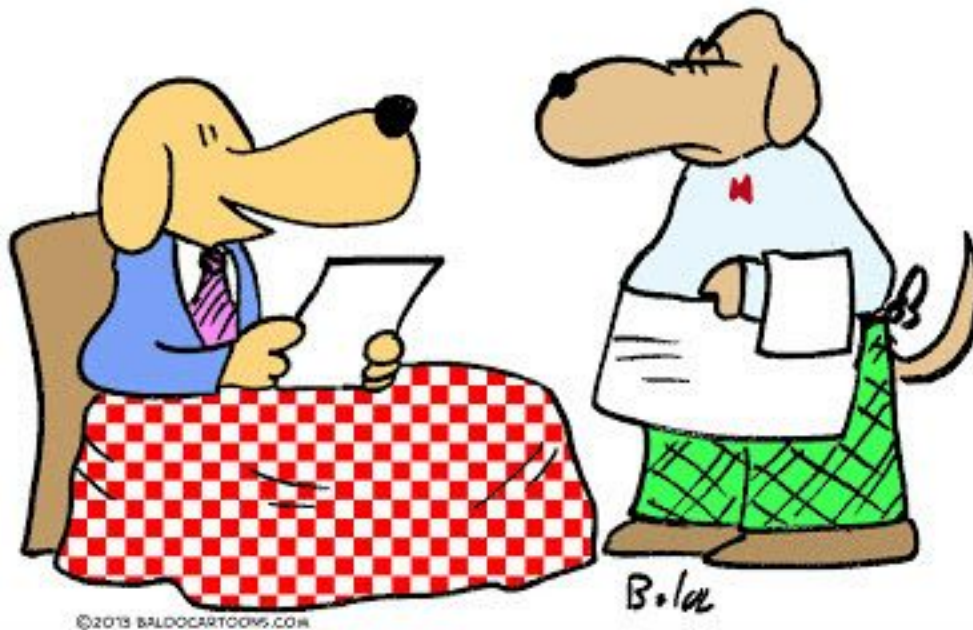


SWE: Serve with Ease

GROUP #1

Project: Restauranting Made Easy



"How's the homework today?"

Team Members: Annie Antony, Athira Haridas, Christina Parry, Emma Roussos, Christina Segerholm, and Nishtha Sharma

332: 452 Software Engineering, Professor Ivan Marsic

Github URL: <https://github.com/powerpuffprogrammers>

Division of Work

Athira: Traceability Matrix, Concept Definitions, References, Plan of Work, Product Ownership, Mathematical Models, Major Revisions on Part 1

Annie: Traceability Matrix, Concept Definitions, Association Definitions, Attribute Definitions, References, Plan of Work, Product Ownership, Major Revisions on Part 1 and 2

Christina S: Traceability Matrix, Concept Definitions, Association Definitions, Attribute Definitions, References, Plan of Work, Product Ownership, Mathematical Models, Major Revisions on Parts 1 and 2

Emma: Plan of Work, Systems Operation Contract, Concept Definitions, Association Definitions, Revisions on Use Case 3

Christina P.: Plan of Work, Gantt Chart, Systems Operation Contract, Association Definitions, Revisions on Use Case 3, Report 1 Formatting

Nishtha: Traceability Matrix, Concept Definitions, Association Definitions, Attribute Definitions, References, Plan of Work, Product Ownership, Report Layout Organization, Major Revisions on Part 1 and 2

Changes Since Last Iteration:

- Host GUI
 - split screen
 - one half shows all empty tables
 - one half shows all occupied/on check tables
 - each table holds the name of the server who has that table
 - Changed table status options:
 - ready = clean and able to be sat
 - seated = party is still at table and not close to leaving
 - on check = waiter took payment for that table already, signifies table should be bussed soon.
- Message Controller
 - We changed the layout of the system so that the message controller only interacts between user interfaces. The databases will be updated by the user interface that is associated with it.
- Chef GUI
 - The ticket queue has changed by modifying the following:
 - color of ticket:
 - green = all dishes are finished
 - yellow = all dishes started
 - orange = some dishes started
 - red = all dishes unstated
 - ordering of tickets:
 - First by color (all greens first)
 - Within those colors by chronological order

Table of Contents:

1. Customer Statement of Requirements (CSR)	
a. Problem Statement	5
i. Host/Hostess	
ii. Waiter	
iii. Manager	
iv. Manager	
v. Head Chef	
b. Novelties in our Solution	7
c. Glossary	11
2. System Requirements	
a. Functional and On-Screen Appearance Requirements	11
i. User Stories	
b. Non-Functional Requirements	15
3. Functional Requirements Specification	
a. Stakeholders	16
b. Actors and Goals	16
i. Initiating Actors	
ii. Participating Actors	
c. Use Cases	17
i. Casual Descriptions	
ii. Internal Structure Diagram	
iii. Use Case Diagram	
iv. Traceability Matrix	
v. Fully Dressed Description - Specific Use Cases	
1. UC1: Closing a table	
a. Casual Description	
b. Fully Dressed Description	
c. System Sequence Diagram	
d. User Effort Estimation	
2. UC2: Low Inventory	
a. Casual Description	
b. Fully Dressed Description	
c. System Sequence Diagram	
d. User Effort Estimation	
3. UC3: Placing an Order	
a. Casual Description	
b. Fully Dressed Description	

c.	System Sequence Diagram	
d.	User Effort Estimation	
4.	User Interface Specification	
a.	Preliminary Designs	30
i.	General Employee Screens	
ii.	Head Chef Screens	
iii.	Host Screens Screens	
iv.	Waiter Screens Screens	
v.	Manager Screens	
b.	User Effort Estimation	34
i.	Scenario 1: Waiter Places an Order	
ii.	Scenario 2: Starting a dish on a ticket	
iii.	Scenario 3: Closing a Table	
5.	Domain Analysis	
a.	Domain Model	35
i.	Concept Definitions	
ii.	Association Definitions	
iii.	Attribute Definitions	
iv.	Traceability Matrix	
b.	System Operation Contracts	41
c.	Mathematical Model	43
6.	Plan of Work	
a.	Project Management	45
b.	Product Ownership	45
c.	Breakdown of Responsibilities	46
7.	References	48

Customer Statement of Requirements

a. Problem Statement

As a restaurant owner, my customers expect various services to be delivered fast, such as seating and ordering. In order to help my restaurant perform better, I would like a system which automates certain services to assist the employees in their roles. It should improve each employees' organization of their tasks, communication with the rest of the staff, and speed of service to the customer.

The following are problems faced by my employees:

- Host: Deciding which table to bus next
- Waiter: Taking and placing orders; Managing a newly sat table, an existing table, or unsatisfied tables
- Head Chef: Managing inventory, Organizing food preparation
- Manager: Alerting staff

These and more are further described below.

Host / Hostess

Problem 1: Promptly Bussing Tables

As a host working in a busy restaurant, one of my main responsibilities is to determine which tables need to be bussed and to convey that information to the busboy. Currently, I look around the restaurant to see if customers are getting up to leave or if any tables are empty and need to be cleaned. This actually takes up a lot of my time that could be spent assisting and seating customers. It is also a little embarrassing when customers that are waiting to be sat point out to me that certain tables aren't clean. I am the first person customers interact with in the restaurant and my top priorities are to make sure customers don't have to wait for a long time and that they have an enjoyable experience. I would like to have a software application that helps make my job easier so I can focus more on the customers. I need to have a way to determine which tables need to be bussed and when. Also, I need the application to show me which tables are ready for customers to be seated. By having a list of tables that are ready to be seated, I will be more organized and can work faster since I won't have to look around the restaurant for empty tables or ones that need to be bussed. I feel that through this system, I will be able to improve how well I do my job.

Waiter

Problem 1: Taking an order

Everyone knows a waiter's main job is to take the table's order. It doesn't sound complicated but when I'm on my third table, it's hard to stay on top of everything. It's easy to forget to ask a guest ordering the steak how he would like it cooked when I have a million other things on my list to do in the next few minutes. In this case, two situations can occur.

Scenario A, I take the walk of shame back to the table and ask the question that I forgot. This not only wastes my time but also makes me look unprofessional. Or scenario B, where I assume that the customer wants their steak done medium, based on popular choice. But now I run the risk of creating an angry customer who wanted a rare steak and causing more work for me in the long run. I would like to have a system to place orders while customers are requesting them. The system would need to have all of the menu items on it as well as the necessary preference options. So now when a customer orders a steak, I can open up the table, click the steak, and then be prompted to pick a wellness level, the size, and a choice of side for the dish. Using an application to generate a computer ticket will help me take orders faster which will allow me to focus more on my guests and handle more of them at once.

Problem 2: Being notified when I get sat/when hot food is ready

One of the most embarrassing mistakes a waiter can make is not greeting a table within a certain amount of time after the party sits down. When the host seats people at my corner table, it can take me a while before I realize that they have been sat there. Now I have a couple of hungry and angry customers at my table because of my negligence to check my section frequently. It would be great if on the device, there was a notification system that would notify me when I get sat. In order to keep track of what tables I have, I would also like the system to tell me the table numbers of my active tables. This way when I go to place an order, I can see a list of all the tables I have at the moment and click which one I want to place an order for. I also would like to be notified when my table's food is ready to be run out. It is very time consuming to keep walking back and forth into the kitchen checking if any of my tables' food is ready. Instead, I should be notified that my food is ready through the application. This system will improve my communication with the kitchen and the host and allow more clarity on what I should be doing at the moment.

Problem 3: Notifying a manager about unhappy customer

One of the hardest things to deal with as a waiter is an unhappy customer. No matter what the cause, sometimes I do not have the ability to change their mind. In this case, I have to call my manager over and they take care of the rest. For example, when my table gets their steak rare instead of medium. Cluelessly, I go over and check how everything came out but I get an ear full of complaints from the customer. My next move is to notify my manager, but this may take a while depending on what I am doing at the time, making my customer more unsatisfied. I would like to be able to notify my manager through the app without having to look for him. This communication tool will save manager and I precious time and will lead to happier customers.

Manager

Problem 1: *Notifying all employees*

A manager is pretty much the backbone of the restaurant. It is my job to keep all employees updated when we run out of a certain food so they can notify their customers. The sooner I update them, the better the chance of us not having any problems with customers that might try to order that specific food item. But it is not easy trying to track down each waiter. I wind up running around the restaurant and sometimes I can't deliver the information soon enough. If that happens then the waiter and I have to reproach the customers, apologize, and try to see if they would like to change their order or not. Most of the time the customer gets disappointed, frustrated, or angry that they cannot have what they originally wanted. We need a way to send each other notifications very quickly and efficiently. The faster I can get the message out, the better. I would like something to ease the communication between the staff.

*Head Chef*Problem 1: *Keeping track of inventory*

The main attractions of a restaurant is the food that it serves. As head chef, my job involves supervising the sous chefs and making sure all the orders have been covered. It doesn't sound so chaotic, however, on weekends, the restaurant is packed with hungry customers. It is easy to lose track of what ingredients are running low, when I am busy trying to make sure that the current dishes come out correctly and quickly. Low inventory causes some of the customers to not get the dishes they order. It wouldn't be efficient for me to keep running to the inventory periodically to make sure that we have all the items we need for our dishes. I would like the application to keep track of my inventory and alert me when it is low. Having this would allow me to focus on my main job of running the kitchen.

Problem 2: *Keeping track of ticket orders*

As a chef, one of the problems that I run into is how to manage the immense amount of orders I get during rush hours. The current system in use is paper tickets on a rotating ticket holder. This is very unorganized and creates chaos in the kitchen when I have to decide which order is next. I would like a way to organize my incoming tickets in chronological order. It is also difficult to keep track of the progress of multiple dishes for one table. The goal is to have all dishes come out together. I would like the application to have a system which keeps track of which dishes on the ticket are started. This way I can be sure that the dishes come out together. These two features will increase my organization and reduce my mistakes.

b. Novelties in our Solution

Our system is not like the other restaurant systems. Our aim is not to automate the whole restaurant process or cut anyone out of a job. Two of our members work as waitresses in local restaurants so our group has a grasp on how a restaurant works. Customers don't come out to eat just for the food. Customers come out to eat for the experience, which includes the food and the service. They want to discuss what looks good on the menu and ask their waiter what his favorite dish is. Our system is going to make each employee's life easier, including the owner's. By providing each employee with a tablet, their tasks will be simplified. This will make each employee less stressed and more customer friendly. Previous systems have taken out the

intermediate workers, like hosts and waiters, and replaced them with tablets and podiums. Our solution is unique because speed is not our only goal, we also aim to increase organization and communication within the restaurant. In this case we will not just be increasing the speed of service to the customer but also raising customer satisfaction by taking some of the stress off of the staff.

Host

Promptly Bussing Tables

- The way we go about notifying the host that a table is almost ready to be cleaned (on check/leaving) is different than previous methods.
 - Group 1 in 2013 had an on check method where they would update the corresponding person once the server printed the check.
 - In our method, we have the waiter press a “paid” button after he has taken that table’s payment. After this the host is signaled that the table will be leaving soon as it is on check.
 - Our way is more efficient because we cut out the variable wait time between the table receiving the check and the table paying for the check.
- The way we go about displaying the host screen is also different than previous projects.
 - Group 1 in 2013 showed all of the tables under one list under the host screen.
 - In our solution, the host sees two different lists at once. One list is the list of ready tables (empty and clean tables) and another list for seated or on check tables.
 - This is more efficient as the host does not need to scroll or switch screens to view which tables are ready and which ones need to be bussed.

Waiter

Taking an order

- The way we go about having the waiter take an order is similar to the implementation of previous groups.
 - Group 4 of Spring 2014 had the waiter place an order on a tablet and send it to the chef.
 - In our method, we have a similar implementation.

Being notified when I get sat/when hot food is ready

- The way we notify a waiter when he has recently been sat is different than previous projects.
 - Group 3 of Spring 2015 had the host send the waiter a message when they have been sat.
 - Our solution is to still have that message sent but have multiple notifications. First the message will be displayed as a hanging notification on the waiter's screen until the waiter clicks the acknowledge. A second notification will be placed on the table button that says "Recently Sat". This would not disappear until the table was opened.
 - Our solution is better because it ensures that the waiter has seen the message and knows he is sat. It will keep the waiter reminded until he approaches the table and open the ticket.
- The way we notify a waiter when he has hot food waiting is different than previous projects.
 - Group 3 of Spring 2013 has the chef send a message to the waiter to notify the server he has hot food waiting.
 - Our solution is to still have that message sent but have the message displayed on that tables button in the waiter's screen. On that button there will be a "Hot food" message under the table number. The text field will disappear upon clicking on the table, since opening the ticket implies that the waiter is preparing to take out the hot food.
 - Our solution is better because it ensures that the waiter has acknowledged that he has hot food and is preparing to take it out.

Notifying a manager about unhappy customer

- Our method for handling unsatisfied customers is different than previous groups'.
 - While many groups did discuss customer satisfaction in terms of wait time but not follow up with the rest of the customer's experience. They assumed that with the application the customers would always be happy with what they ordered.
 - In our solution, we have a "notify manager" button which allows the waiter to send a message to the manager about an unsatisfied customer.

- Our way is better because it allows for quick and easy communication between the waiter and the manger, reducing the amount of time the unhappy customer has to wait.

Manager

Notifying all employees

- Our application for the manager was different than previous projects.
 - Group 3 of Spring 2015 gave the manager the ability to communicate with one waiter at a time.
 - Our solution supplies a broader messaging method. The manager will be allowed to broadcast a message to all employees (host, servers, and chef). Our solution will also provide quick interactions between all employees and the manager by having a button on their screens to call the manager.
 - Our solution is better because it allows for easier communication between employees.

Head Chef

Keeping track of inventory

- The way we go about keeping track of inventory is similar to the implementation of previous groups.
 - Group 3 of Spring 2013 kept track of the amount of ingredients in the inventory so the chef could determine what needed to be restocked.
 - In our method, we have a similar implementation. We decrement the quantity of the ingredients used in the dish when the chef changes the status to started.

Keeping track of ticket orders

- The way we go about keeping track of ticket orders is different from the implementation of previous groups.
 - Group 4 of Spring 2014 have organized their tickets by chronological order.
 - Our solution uses more states to categorize the progress of a ticket. We break down the organization into two factors:
 - First is separation by ticket progress (Finished, started, semi-started, unstarted)
 - Then by chronological order (based on when the chef received the ticket).

- Our solution is more organized because having multiple states within the first separation allows the chef to more easily keep track of what dishes need to be prepared next.

c. Glossary

- ❖ **Head Chef:** Chef that handles delegating all of the orders. Keeps track of what food needs to come out next and tells his chefs what they need to be making. Also in charge of inventory.
- ❖ **Waiter:** Waiter or waitress, responsible for good customer service and taking the orders of customers. Also should run out food.
- ❖ **Running food/ Running out food:** Taking dishes that are ready to be served from the kitchen to the table. Delivering food to a table.
- ❖ **Party:** Group of customers who are dining together (can be party of 1,2,...n)
- ❖ **Ticket:** A list of orders for a specific table. This is what the waiter gives to the chef to tell them what they need to be cooked for a certain table.
- ❖ **Owner:** Owns the restaurant, has the highest stake in the company. Can also act as a manager.
- ❖ **Manager:** Runs the show, makes sure customers are happy and that everything is running smoothly.
- ❖ **Host:** Greets the customers and gives them the wait time. Handles bussing and seating tables.
- ❖ **Head Host:** Handles what table each party should be sat at.
- ❖ **Bussing a table:** Cleaning a table
- ❖ **Comps:** Complimentary dishes. You would say a manger has comped off the meal if they buy the meal for the customer.
- ❖ **Hot Food:** An order that is done and ready to be taken out to a table.
- ❖ **Got Sat:** Use this term to describe the scenario where a party sits down at a table in a waiter's section. Example: "You just got sat at table 32" means that a group of customers has just sat down at table 32 and you are their waiter.
- ❖ **Database:** Holds all of the information about the restaurant on it.(Restaurant name, Employee information, menu information, inventory information) Will be run on a desktop computer. All tablets/clients will be able to read and write to database.

Systems Requirements

a. Functional and On-Screen Appearance Requirements

i. User Stories

Priority is in descending order where the most important user story is at the top.

Size is how much effort goes into implementing that story.

For the ID of these user stories, the format is:

ST - position initial - requirement type (functional: F, or on screen: O) - requirement #

For example: ST-GE-O-1 stands for Story - General Employee - On Screen - 1

General Employee:

ID	User Story	Size
ST-GE-F-1	As an employee, I should be able to pick up any tablet and log in using my employee ID.	4
ST-GE-F-2	As an employee, I should be directed to the correct screen depending on my position after logging in.	2
ST-GE-O-3	As an employee, I should be able to logout from my screen using a logout button.	3
ST-GE-O-4	As an employee, I should be able to see who I am logged in as on the screen.	3
ST-GE-O-5	As an employee, when typing in my ID, I would like to see a number pad.	2

Head Chef:

ID	User Story	Size
ST-HC-F-1	As a head chef, I should be able to view the order of the tickets and update the status of the ticket.	6
ST-HC-F-2	As a head chef, I should be able to see when a new order is placed.	7
ST-HC-F-3	As a head chef, I should be able to notify the waiter when the order is complete.	6
ST-HC-F-4	As a head chef, I should be updated when the number of items in the inventory for an ingredient is low.	3
ST-HC-O-5	As head chef, I should be able to update the dish within a ticket by pressing a button.	4
ST-HC-O-6	As head chef, I should be able to view the list of ticket orders on my screen.	2
ST-HC-O-	As head chef, I should be able to view the contents of the ticket and the	3

7	status of each dish on my screen.	
ST-HC-O-8	As head chef, I should be able to change the status of a meal within a ticket with by clicking on that dish's status.	3
ST-HC-F-9	As a head chef, I should be able to see the orders I haven't attended yet.	4
ST-HC-O-10	As a head chef, when I view my ticket screen, tickets should be ordered and color coded based on their completeness.	3

Host:

ID	User Story	Size
ST-H-F-1	As a host, I should be able to view all tables.	4
ST-H-F-2	As a host, I should be able to view the status of any table.	5
ST-H-O-3	As a host, I should see a list of the tables with the ones available to be sat shown in an obvious way.	5
ST-H-F-4	As a host, I should be notified when a specific table is on check.	3
ST-H-O-5	As a host, I should be able to update the status of a table from empty to occupied by clicking on that table's button.	2
ST-H-O-6	As a host, I should be able to view the maximum occupancy of any table.	2

Manager:

ID	User Story	Size
ST-M-F-1	As a manager, I can send the waiters notifications about food items that are not available.	4
ST-M-F-2	As a manager, I can view all notifications from the hosts, waiters, and the kitchen.	3
ST-M-O-3	As a manager, I would like to be able to create a new message with the push of a button.	2
ST-M-O-4	As a manager, I should be alerted on screen whenever I receive a new message.	2
ST-M-F-5	As a manager, I can pick who I want to send the messages too.	4

ST-M-O-6	As a manager, I can delete notifications after I have dealt with them with the push of a button.	2
----------	--	---

Waiter:

ID	User Story	Size
ST-W-F-1	As a waiter, I should be able to start a new ticket for a newly seated table.	2
ST-W-F-2	As a waiter, I should be able to reopen an existing ticket for a table.	2
ST-W-O-3	As a waiter, I would like to be able to click on a button with a table number on it and have it open up the ticket for the corresponding table.	3
ST-W-F-4	As a waiter, I should be able to enter in any menu items under a table.	4
ST-W-F-5	As a waiter, I should be able to modify dishes when placing an order.	7
ST-W-F-6	As a waiter, I should be able to easily send a ticket to the kitchen when I'm done taking it.	3
ST-W-F-7	As a waiter, I should be able to print out my check.	4
ST-W-O-8	As a waiter, I should be able to see all the active tables I have on my home screen.	3
ST-W-F-9	As a waiter, I should be alerted when I get sat.	3
ST-W-F-10	As a waiter, I should be alerted when one of my table's food is ready.	3
ST-W-F-11	As a waiter, I should be able to notify a manager that something is wrong with a table.	3
ST-W-F-12	As a waiter, I should be able to easily go back and edit an order that I have taken but not placed yet.	3

ST-W-O-1 3	As a waiter, I would like to have some sort of color coordination on the screen, green for good things that are ready, and red and orange for bad things.	4
---------------	---	---

Owner:

ID	User Story	Size
ST-O-F-1	As the owner, I should be able to add and remove employees on the database.	3
ST-O-F-2	As the owner, I should be able to modify employee information on the database.	3
ST-O-F-3	As the owner, I should be able to modify menu item data on the database.	5
ST-O-F-4	As the owner, I should be able to add and remove menu items on the database.	5
ST-O-F-5	As the owner, I should have exclusive access to these features.	3
ST-O-O-6	As the owner, when I log on I should be able to pick whether I want to change the menu, the inventory, or the employee data.	2
ST-O-O-7	As the owner, when I decide what section I wish to change (menu, employee, or inventory) I should have buttons to add, remove, or modify a current item or employee.	4
ST-O-O-8	As the owner, when I am looking at a list of employees or items, I should have a scroll bar to access all of them.	3

b. Non-Functional Requirements

ID	Priority	Description
NF-1	2	The GUI for the staff should be easy to use and not require much time to understand.

NF-2	4	The system must allow for quick communication between devices.
NF-3	4	The system should be able to back up information and read from backed up information.
NF-4	5	The system shall not allow unknown devices to connect to it
NF-5	2	The database system should be scalable in regards to amount of employees and menu items it can store.
NF-6	5	The system should have an authorization subsystem that determines a certain client's permission to the database.
NF-7	4	The system should only allow the computer with the waiter process running on it to modify the database.

Functional Requirements Specifications

a. Stakeholders

The users are the most invested stakeholders in the success of this software. The staff that will use this will include the head chef, manager, host and a few waiters. Our system will benefit each of these members by making their roles within the restaurant easier. Another stakeholder are the customers of the restaurant. Their dining experience will be enhanced because of the increase in the speed of service of the restaurant due to the implementation of our system. Owner is a big stake holder of this system as it will affect their business and their profits. The developers of the system also are stakeholders. The result of the software is a direct reflection of the developers work.

b. Actors and Goals

i. Initiating Actors

Waiter:

- **Role:** The staff member who greets, takes orders, and brings out hot food to customers. Also in charge of taking payments for tables.
- **Goal:** Keep all of their tables happy and satisfied by providing friendly and fast service.

Head Chef:

- **Role:** The staff member who is delegates tasks to other chefs. Must keep the incoming orders under control and organized.
- **Goal:** Make sure all orders go out correctly and quickly.

Head Host:

- **Role:** The staff member who greets customers when they first walk into the restaurant. They are in charge of selecting which table to sit a party at and relaying this information to the waiter at that table. They are also in charge of the busboys.
- **Goal:** Minimize the time a party must wait in order to be sat. Notify the waiter of the newly sat table.

Manager:

- **Role:** The employee who oversees the whole restaurant's operation. They deal with problems among the staff or from dissatisfied customers.
- **Goal:** Make sure customers leave satisfied and that each part of the restaurant is running smoothly.

ii. Participating Actors

Message System:

- **Role:** This system processes requests it gets from different users of the application and handles them appropriately, either by sending a message to another employee or updating an interface.
- **Goal:** To get a message or form of data from one staff member to another.

Busboy:

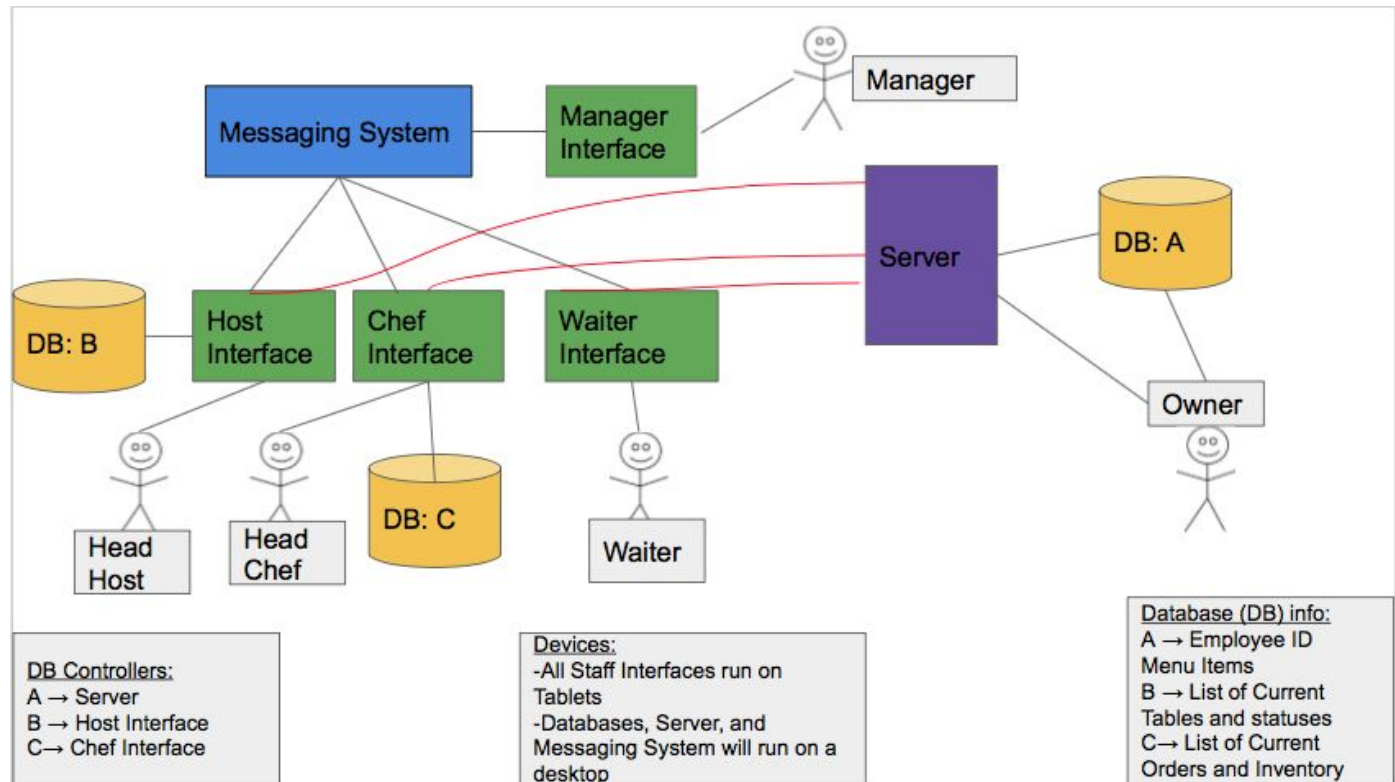
- **Role:** The staff member who is in charge of cleaning off dirty tables.
- **Goal:** Reduce the customer's wait time by cleaning tables as soon as their occupants leave.

c. Use Cases

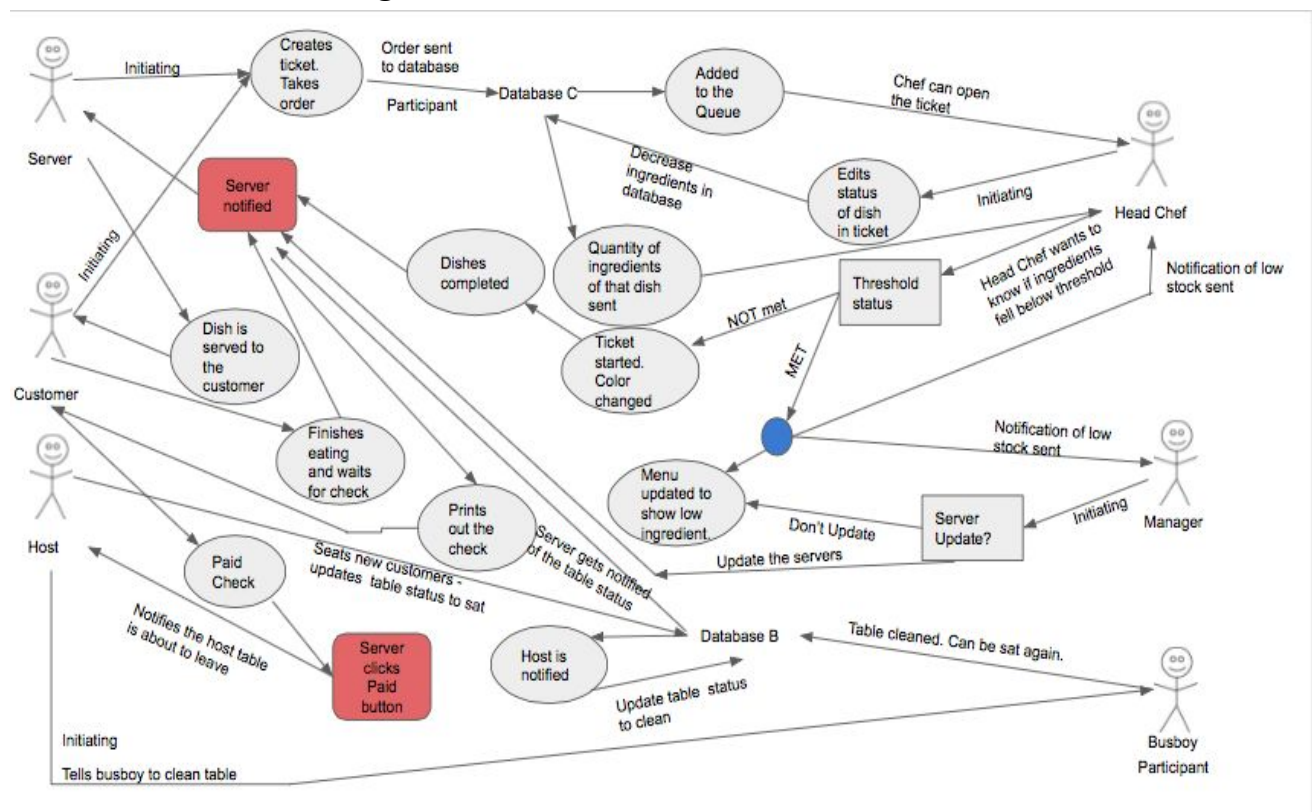
i. Casual Descriptions

The casual descriptions are expanded in individual use cases below.

ii. Internal Structure Diagram



iii. Use Case Diagram



iv. Traceability Matrix

	Use Case 1 - Closing a Table	Use Case 2 - Low Inventory	Use Case 3 - Placing An Order
ST-GE-F-1			
ST-GE-F-2			
ST-GE-O-3			
ST-GE-O-4			
ST-GE-O-5			
ST-HC-F-1			
ST-HC-F-2			
ST-HC-F-3			
ST-HC-F-4			
ST-HC-O-5			
ST-HC-O-6			
ST-HC-O-7			
ST-HC-O-8			
ST-HC-F-9			
ST-HC-O-10			
ST-H-F-1			
ST-H-F-2			
ST-H-O-3			
ST-H-F-4			
ST-H-O-5			
ST-H-O-6			
ST-M-F-1			
ST-M-F-2			
ST-M-O-3			
ST-M-O-4			
ST-M-F-5			
ST-M-O-6			
ST-W-F-1			
ST-W-F-2			
ST-W-O-3			
ST-W-F-4			
ST-W-F-5			
ST-W-F-6			
ST-W-F-7			

ST-W-O-8			
ST-W-F-9			
ST-W-F-10			
ST-W-F-11			
ST-W-F-12			
ST-W-O-13			
ST-O-F-1			
ST-O-F-2			
ST-O-F-3			
ST-O-F-4			
ST-O-F-5			
ST-O-O-6			
ST-O-O-7			
ST-O-O-8			

v. Fully Dressed Description - Specific Use Cases

1. UC1 : Closing a table

→ What user does

← what system does in response

a. UC1 Casual Description

This use case describes the processes that the different staff members must go through when a table is done eating and ready for the check. It starts when the waiter prints out the check and continues all the way through the host re sitting that table with a new party.

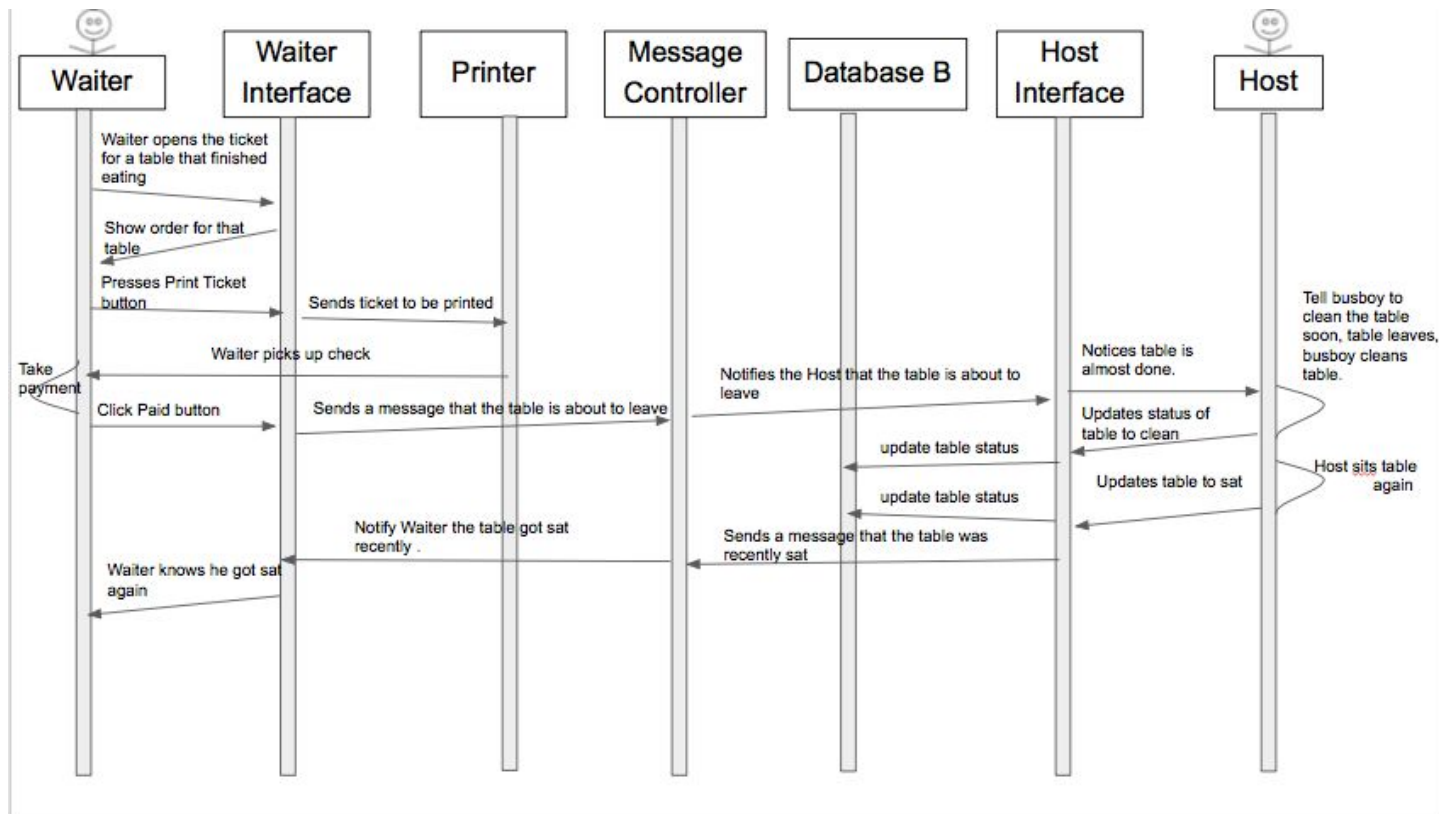
b. UC1 Fully Dressed Description

Related User Stories	ST-H-F-1, ST-H-F-2, ST-H-F-4, ST-H-O-5, ST-W-F-2, ST-W-O-3, ST-W-F-7, ST-W-O-8, ST-W-F-9, ST-W-F-12, ST-W-O-13
Initiating Actor	Waiter
Actor's Goal	Print out the check to give to the table, and take the payment for that check so the party can leave.
Participating Actors	Host, Busboy
Pre-Conditions	Check is viewable on-screen and there is a print check button on the bottom of the screen that will print the check at the printer.
Post-Conditions	The table should no longer be shown as active(green) on the waiter screen. The host's screen should be updated to include a on check status for the departing table.

<p>Flow of Events for Main Success Scenario</p>	<p>1→ Waiter opens up the table's ticket by clicking the button on the screen with the corresponding table number on it.</p> <p>1.1←The system shows the current ticket on screen as well as a print check button on the bottom left.</p> <p>2→ Waiter clicks print check button on the screen.</p> <p>2.1←The system sends the check to the restaurants printer to be printed.</p> <p>→ Waiter picks up the check from the printer and delivers it to the table.</p> <p>3→ Waiter takes the payment for the table. He clicks the paid button under that table as that table is done. Then clicks the yes button to the "Are you sure" question.</p> <p>3.1←The system updates the waiter's software so that the button for that table number is now inactive(white) on the waiter's screen.</p> <p>3.2←The system sends a message to the host application notifying the application that this table is on check and leaving soon. This is an indirect message sent first to the message controller who then forwards it to the host.</p> <p>3.2.1←The system updates the host's screen by changing the status of the table to on check.</p> <p>→ Host views his screen with the list of tables and realizes that there is a table on check. He notifies a busboy to keep prepare to bus that table in the near future.</p> <p>→ The party leaves and the Busboy cleans the table.</p> <p>4→ Host acknowledges that table is cleaned and changes the status of that table from on check to empty.</p> <p>4.1←The system shows the updated list of table statuses on the host's screen with the newly cleaned table near the top of the list.</p> <p>5→ Host sits the same table again with a different party and changes that table's status to occupied.</p> <p>5.1←The system shows the table as occupied on the host's screen and that table moves down the list as it can no longer be sat and so shouldn't be with the clean tables.</p> <p>5.2←The system sends a message to the corresponding waiter's application to notify them they have gotten sat at that table and change the color of that table to yellow(recently sat/new). This is an indirect message sent first to the message controller who then forwards it to the waiter.</p> <p>5.3→ Waiter sees that he has gotten sat at that table number on his screen. He goes to greet it.</p>
<p>Flow of Events for Extension (Alternate Scenarios)</p>	<p>1→ Waiter opens up the table's ticket by clicking the button on the screen with the corresponding table number on it.</p> <p>1.1←The system shows the current ticket on screen as well as a print check button on the bottom left.</p> <p>2→ Waiter clicks print check button on the screen.</p> <p>2.1←The system sends the check to the restaurants printer to be printed.</p> <p>→ Waiter picks up the check from the printer and delivers it to the table.</p> <p>A) Waiter loses the check/ check gets dirty/ reprinting check</p> <p>3→ Waiter opens up the same table's ticket by clicking the button on the screen with</p>

	<p>the corresponding table number on it.</p> <p>3.1←The system shows the current ticket on screen as well as a print check button on the bottom left.</p> <p>4→ Waiter clicks print check button on his screen.</p> <p>4.1←The system sends the check to the restaurants printer to be printed.</p> <p>→ Waiter picks up the check from the printer and delivers it to the table.</p> <p>Progression continues as Main Success Scenario.</p> <p>B) Waiter hits the print button by accident/needs to add something to ticket</p> <p>3→ Waiter opens up the same table's ticket.</p> <p>3.1←The system shows the current ticket on screen as well as options of menu items to add to the list</p> <p>4→ Waiter orders whatever he had to order by clicking appropriate button</p> <p>4.1←The system shows the updates the current order on waiter's screen.</p> <p>5→ Waiter clicks send order button.</p> <p>5.1←The system sends the order to the head chef's application. This is an indirect message sent first to the message controller who then forwards it to the head chef.</p> <p>6→Later, the waiter prints the new check by pressing print check button.</p> <p>6.1←The system sends the check to the restaurants printer to be printed.</p> <p>→ Waiter picks up the check from the printer and delivers it to the table.</p> <p>Progression continues as Main Success Scenario.</p> <p>C) Host changes the status of table to clean before it's cleaned</p> <p>3→ Waiter takes the payment for the table. He clicks the close table button under that table as that table is done.</p> <p>3.1←The system updates the waiter's software so that table number will now have a new ticket when clicked on.</p> <p>3.2←The system sends a message to the host application notifying the application that this table is on check and leaving soon. This is an indirect message sent first to the message controller who then forwards it to the host.</p> <p>3.2.1←The system updates the host's screen by changing the status of the table to on check.</p> <p>→ Host views his screen and realizes that there is a table on check. He notifies a busboy to keep prepare to bus that table in the near future.</p> <p>4→ The party leaves and Host mistakenly changes the status of a dirty table to empty.</p> <p>4.1←The system updates the host's screen by changing the status of the table to empty.</p>
--	--

c. UC1 System Sequence Diagram



d. UC1 User Effort Estimation

Mouse Clicks:

➤ Waiter:

- 1 click to print check.
- 2 clicks to tell application the table has paid. First click to press paid button, then click yes to “Are you sure?”.

➤ Host:

- 2 clicks to update status of a table. First click update status, then click which status to be updated to.

2. UC2 : Low Inventory

a. UC2 Casual Description

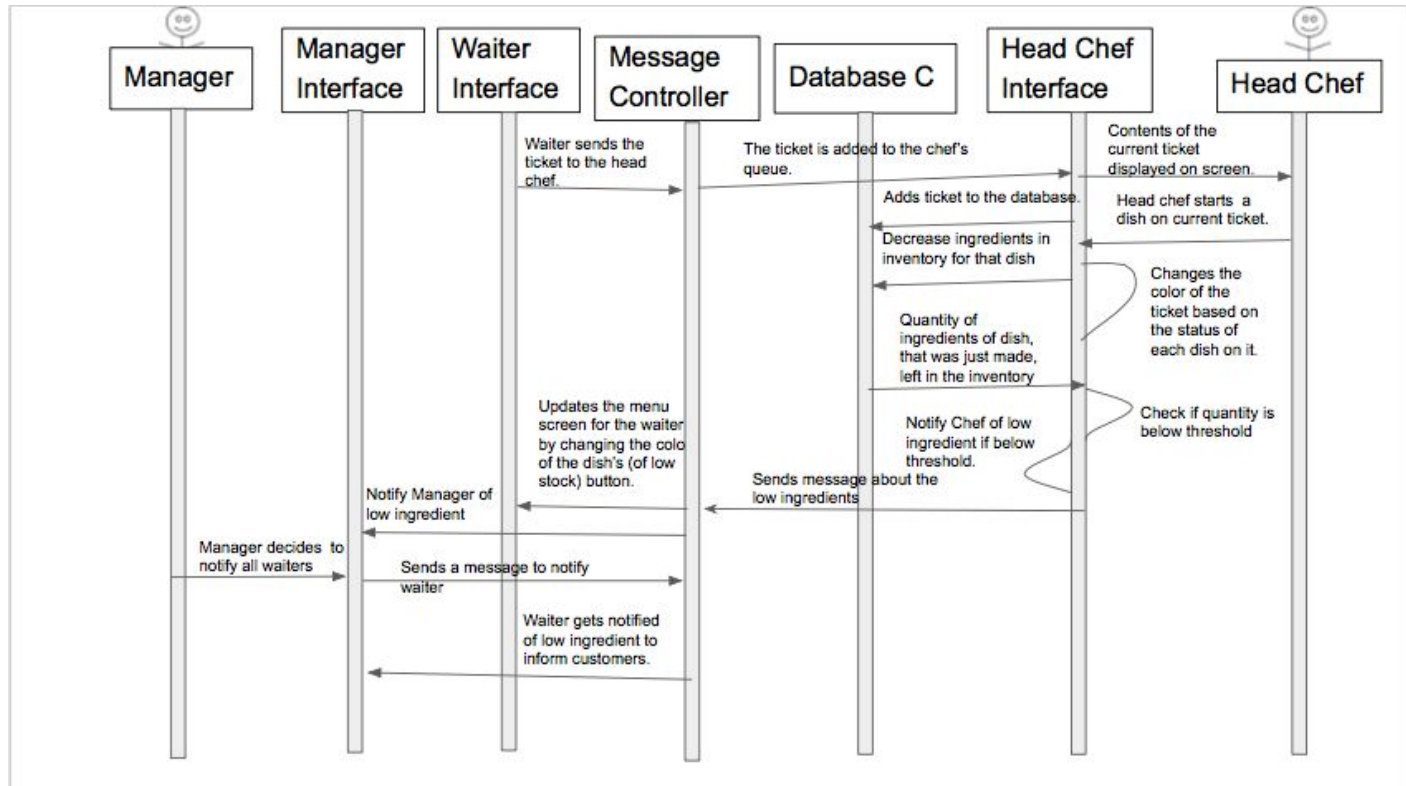
This use case describes the processes that the different staff members must go through when the inventory of ingredients at the restaurant begins to run low. It starts from the chef starting a ticket to alerting the manager and continues all the way to the waiter receiving a message from the manager.

b. UC2 Fully Dressed Description

Related User Stories	ST-HC-F-1, ST-HC-F-2, ST-HC-F-4, ST-HC-O-5, ST-HC-O-7, ST-HC-O-8, ST-HC-O-10, ST-M-F-1, ST-M-F-2, ST-W-F-11
Initiating Actors	Head Chef, Manager
Actor's Goal	To maintain a record of the stock of ingredients and inform staff when low on one. Keep an organized queue of tickets that are in the processes of being made.
Participating Actors	Waiters
Pre-Conditions	The steady database would have the information about the quantity of items in the inventory. At the start of each day, the head chef will load the data from that database into the inventory/order database. This will allow him/her to have the most updated information about the inventory and accordingly have accurate responses for stock.
Post-Conditions	Each time the head chef starts a dish on a ticket, the system should be able to deduct the correct amount of ingredients that the dish uses from the number listed in the inventory (inventory/order database). If the items of the inventory fall below a specified threshold, the system should alert both the manager and the chef and give a color coding on the waiter's menu screen that is different for dishes that use those ingredients. The manager can send a message to all the waiters about the low amount of ingredients based on his/her decision.
Flow of Events for Main Success Scenario	<p>1→ Head Chef receives a ticket from the waiter.</p> <p>1.1← The system shows the new ticket on the chef's screen.</p> <p>2→ Head Chef clicks on the ticket.</p> <p>2.1→ The system shows the dishes ordered on that ticket.</p> <p>3→ Head Chef clicks on a dish to start it.</p> <p>3.1← The system deducts the corresponding quantity of involved ingredients from the inventory.</p> <p>4→ Head Chef changes the status of each dish on the ticket.</p> <p>4.1← The system changes the color of the ticket based on the status of each dish.</p> <p>5→ Threshold of low stock is met.</p> <p>5.1← The system sends an alert in the form of a non-interrupting banner to inform the head chef.</p> <p>5.1.1→ Chef can view the item(s) and quantity left of the ingredient(s) causing the low stock through a button on their screen.</p> <p>5.2← The system updates the menu screen to reflect the change visually for the waiter (most likely by changing the color of the item).</p> <p>5.3← The system sends an alert in the form of a notification to inform</p>

	<p>the manager. This is an indirect message sent first to the message controller who then forwards it to the manager.</p> <p>→ Manager can choose whether to notify the waiters or not.</p> <p>6→ Manager chooses to alert the waiters of low stock.</p> <p>6.1←The system sends a message to the waiter alerting them of the ingredient that is running low. This is an indirect message sent first to the message controller who then forwards it to the waiters.</p> <p>7→ Waiter receives the message and can accordingly alert the parties who order dishes which use the corresponding ingredient.</p>
Flow of Events for Extension (Alternate Scenarios)	<p>1→ Head Chef receives a ticket from the waiter.</p> <p>1.1← The system shows the new ticket on the chef's screen.</p> <p>2→ Head Chef clicks on the ticket.</p> <p>2.1→ The system shows the dishes ordered on that ticket.</p> <p>3→ Head Chef clicks on a dish to start it.</p> <p>3.1←The system deducts the corresponding quantity of involved ingredients from the inventory.</p> <p>4→ Head Chef updates the statuses of the dishes on the ticket.</p> <p>4.1← The system changes the color of the ticket.</p> <p>5→ Threshold of low stock is met.</p> <p>5.1←The system sends an alert in the form of a non-interrupting banner to inform the head chef.</p> <p>→ Chef can view the item(s) and quantity left of the ingredient(s) causing the low stock through a button on their screen.</p> <p>5.2←The system updates the menu screen to reflect the change visually for the waiter (most likely by changing the color of the item).</p> <p>5.3←The system sends an alert in the form of a notification to inform the manager. This is an indirect message sent first to the message controller who then forwards it to the manager.</p> <p>→ Manager can choose whether to notify the waiters or not.</p> <p>A) The manager does not send an alert to all waiters.</p> <p>6→ Manager chooses not to alert the waiters of low stock.</p>

c. UC2 System Sequence Diagram



d. UC2 User Effort Estimation

Mouse Clicks:

- Head Chef:
 - 1 click to open ticket.
 - 1 click to start a dish.
 - 1 click to view the low inventory.
- Manager:
 - 1 clicks to update the waiters of low ingredient.
- Waiter:
 - No effort. The waiter only receives an update.

3. UC3 : Placing an order

a. UC3 Casual Description

This use case describes the processes that the different staff members must go through while placing a customer's order. It starts from the waiter opening a new ticket to sending it to the cooks and continues all the way until the waiter brings the food to the customers.

b. UC3 Fully Dressed Description

Related User Stories	ST-S-F-1, ST-S-F-2, ST-S-O-3, ST-S-F-4, ST-S-F-5, ST-S-F-6, ST-S-O-8, ST-S-F-10, ST-S-F-11, ST-S-F-12, ST-S-O-13, ST-HC-F-1, ST-HC-F-2, ST-HC-F-3, ST-HC-O-5, ST-HC-O-6, ST-HC-O-7, ST-HC-O-8, ST-HC-O-10
Initiating Actor	Waiter
Actor's Goal	To place and complete an order
Participating Actors	Head Chef, Cooks
Pre-Conditions	There must be a button on the server's screen to create a new ticket. Also, the customer must place an order.
Post-Conditions	The order has been successfully completed and delivered to the customer.
Flow of Events for Main Success Scenario	<p>1 → Waiter opens new ticket for seated table.</p> <p>1.1 ← The system shows that the new ticket has been created.</p> <p>2 → Waiter fills in the information of the ticket.</p> <p>2.1 ← The system updates the ticket as the information is entered.</p> <p>3 → Waiter clicks send button.</p> <p>3.1 ← The system sends a message indirectly to the head chef (first to the message controller who will forward it to the head chef).</p> <p>3.1 ← The ticket is added to the chef's queue.</p> <p>3.1.2 ← The ticket gets added to the database.</p> <p>3.2 ← The system displays the contents of the current ticket on Head Chef's screen.</p> <p>4 → Head Chef starts cooking the dishes on the current ticket and changes the statuses to started.</p> <p>4.1 ← The system changes the color of the ticket based on the statuses of the dish on it.</p> <p>4.2 ← The ingredients needed for the started dish get subtracted from the inventory in the database.</p> <p>5 → Head Chef updates the status of each dish to finished.</p> <p>5.1 ← The system changes the color of the finished ticket as all dishes are finished.</p> <p>5.2 ← The system sends the notification "hot food" to the waiter when all the dishes for their ticket is done. This is an indirect message sent first to the message controller who then sends it to the waiter.</p>

	6 → Waiter serves the dishes to the table.
Flow of Events for Extension (Alternate Scenarios)	<p>1 → Waiter opens new ticket for seated table. 1.1 ← The system shows that the new ticket has been created. 2 → Waiter fills in the information of the ticket. 2.1 ← The system updates the ticket as the information is entered.</p> <p>A) Waiter changes order. 3 → Waiter clicks on the table to reopen the order. 3.1 ← The system shows the current order and allows more items to be added. 4 → Waiter clicks on items he wishes to add and hits send. 4.1 ← The system confirms and updates the changes in the order by sending a message to the head chef (through the message controller) that will update the head chef's ticket for that table. Progression continues as Main Success Scenario.</p> <p>B) Waiter calls manager for issue. 3 → Waiter notifies manager by pressing a button. 3.1 ← The system sends a message to the message controller who forwards it to the manager. 4 → Manager sees the notification and resolves issue. Progression continues as Main Success Scenario.</p> <p>C) Waiter accidentally presses wrong button. 3 → Waiter clicks the item he just added and then clicks remove button. 3.1 ← The system takes off the item that was highlighted from the ticket. Progression continues as Main Success Scenario.</p> <p>D) Customer is dissatisfied with order. 3 → Waiter notifies manager by pressing a button. 3.1 ← The system sends a notification to the manager about the issue. 4 → Manager resolves the issue with the customer. Progression continues as Main Success Scenario.</p> <p>E) Waiter drops the food. 3 → Waiter apologizes and notifies manager by pressing a button.</p>

3.1 ← The system sends a notification to the manager about the issue through a message (that is sent indirectly to the message controller that forwards it to the manager).

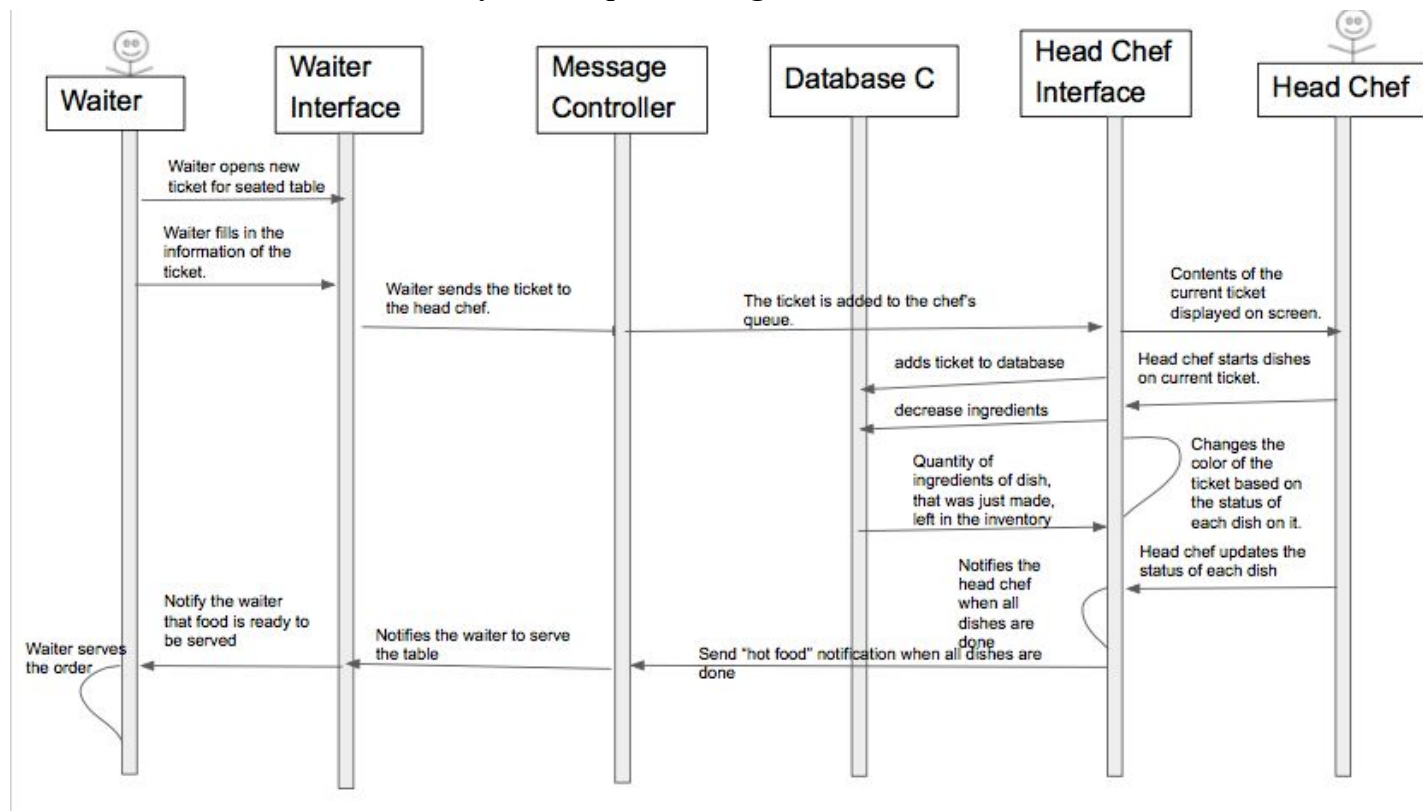
4 → Manager gets the notification and resolves the issue.

5 → Waiter resends the food that was dropped to Head Chef with a message.

5.1 → The system sends the order to Head Chef by using the message controller.

Progression continues as Main Success Scenario.

c. UC3 System Sequence Diagram



d. UC3 User Effort Estimation

Mouse Clicks:

➤ Waiter:

- 1 click to open a ticket.
- 1 click to open a new order within a ticket.
- 1 click to modify an order.
- 1 click to edit an existing order.

- 1 click to undo an order.
- 2 clicks to notify Manager. First click to contact Manager, then another click to send notification.
- 1 click to close ticket.
- 1 click to send order to Head Chef.
- 1 click to confirm delivering food.
- Head Chef:
 - 1 click to add a ticket to active queue.
 - 1 click to view a ticket.
 - 1 click to view an order.
 - 1 click to confirm an order is done.
 - 1 click to confirm a ticket is complete and ready to be delivered to customer.
- Manager:
 - 1 click to open notification from waiter.

User Interface Specification

a. Preliminary Design

i. General Employee Screens

Restaurant Name

Please enter Employee ID:

1	2 <small>ABC</small>	3 <small>DEF</small>
4 <small>GHI</small>	5 <small>JKL</small>	6 <small>MNO</small>
7 <small>PQRS</small>	8 <small>TUV</small>	9 <small>WXYZ</small>
0	Image may be subject to copyright. Le	

ii. Head Chef Screens

Below is the screen the head chef will see when viewing the ticket queue and status.

The diagram illustrates a restaurant ticketing system interface. At the top, a header bar contains 'Logged In as: Kristen Collins' on the left and a 'Logout' button on the right. Below the header, a 'Head Chef' role is indicated. The main area displays a grid of table status cards. On the left, a button '<<See more Tickets' is positioned above the first two columns. On the right, a button 'See more Tickets >>' is positioned above the last two columns. The table status cards are arranged in two rows of three columns each. Each card displays a table number at the top and its current status in the center. The status categories are: 'Finished' (Table 34), 'Finished: All dishes finished' (Table 14), 'Started: All dishes started or finished' (Table 3), 'Semi started: At least one dish unstated' (Table 4), 'Semi started' (Table 5), and 'Unstarted: All dishes are unstated' (Table 6). At the bottom, a navigation bar contains three buttons: '<<<Front of List', 'Notify Manager', and 'End of List>>>'.

Logged In as: Kristen Collins				Logout	
Head Chef					
<<See more Tickets			See more Tickets >>		
Table 34	Table 14	Table 3	Table 4	Table 5	Table 6
Finished	Finished: All dishes finished	Started: All dishes started or finished	Semi started: At least one dish unstated	Semi started	Unstarted: All dishes are unstated
<<<Front of List		Notify Manager		End of List>>>	

Below is the screen the head chef will see when clicking on a ticket.

Logged In as: Kristen Collins		Back
Head Chef	Table 4	
<div>1.Chicken Parmesan Spg w/ Pomo</div> <div>Finished</div>		
<div>2.Chicken Bryan mashed potatoes</div> <div>Started</div>		
<div>3.Steak med Mixed Vegies</div> <div>Unstarted</div>		

<Prev Ticket in the List

Next Ticket in the List>

iii. Host Screens

Below is the screen the host will see when viewing the tables.

Logged In as: John Host

Logout

Ready

Not Ready

Table #	Status	Seats	Server
46	ready	4	Perry
98	ready	6	Tyler
49	ready	4	Annie
14	ready	6	Tyler
47	ready	4	Pedro
15	ready	6	Chris
36	ready	4	Pedro
78	ready	6	Perry

Table #	Status	Seats	Server
52	on check	4	Chris
10	on check	6	Tyler
42	on check	4	Annie
12	on check	6	Nishy
32	on check	4	Pedro
18	seated	6	Athira
35	seated	4	Emma
19	seated	6	Perry

Notify Manager

iv. Waiter Screens

Below is the screen the waiter will observe when selecting a table.

Logged In as: John Deer

Logout

#38

#63
Recently sat

#40

#57

Below is the waiter's view of the screen where they can place an order is shown.

Logged In as: John Deer		Drinks	Apps	Entrees	Desserts	Logout											
Check: Table #63 <table border="1"> <tr> <td>Coke</td> <td>\$1.99</td> </tr> <tr> <td colspan="2">Total: \$1.99</td> </tr> </table>		Coke	\$1.99	Total: \$1.99		<table border="1"> <tr><td>Coke</td></tr> <tr><td>Diet Coke</td></tr> <tr><td>Sprite</td></tr> </table>	Coke	Diet Coke	Sprite	<table border="1"> <tr><td>Apple Juice</td></tr> <tr><td>Iced Tea</td></tr> <tr><td>Lemonade</td></tr> </table>	Apple Juice	Iced Tea	Lemonade	<table border="1"> <tr><td>Beer List</td></tr> <tr><td>Wine List</td></tr> </table>	Beer List	Wine List	
Coke	\$1.99																
Total: \$1.99																	
Coke																	
Diet Coke																	
Sprite																	
Apple Juice																	
Iced Tea																	
Lemonade																	
Beer List																	
Wine List																	
		<table border="1"> <tr><td>Water</td></tr> </table>	Water														
Water																	
Send	Print	Remove	Modify	Notify Manager @63	Back												

Below is the screen the waiter sees when selecting a meal screen/clicking modify button:

Logged In as: John Deer		Drinks	Apps	Entrees	Desserts	Logout																	
Check: Table #63 <table border="1"> <tr> <td>Coke</td> <td>\$1.99</td> </tr> <tr> <td>Chicken Parm</td> <td>\$20.00</td> </tr> <tr> <td>-side</td> <td></td> </tr> <tr> <td>Spg & Pomo</td> <td></td> </tr> <tr> <td colspan="2">Total: \$21.99</td> </tr> </table>		Coke	\$1.99	Chicken Parm	\$20.00	-side		Spg & Pomo		Total: \$21.99		Sides: <table border="1"> <tr><td>Mashed Potatoes</td></tr> <tr><td>Pasta</td></tr> <tr><td>Vegies</td></tr> </table>	Mashed Potatoes	Pasta	Vegies	Pasta: <table border="1"> <tr><td>Ling</td></tr> <tr><td>Penne</td></tr> <tr><td>Spaghetti</td></tr> </table>	Ling	Penne	Spaghetti	Sauce: <table border="1"> <tr><td>Alfredo</td></tr> <tr><td>Pomo</td></tr> </table>	Alfredo	Pomo	
Coke	\$1.99																						
Chicken Parm	\$20.00																						
-side																							
Spg & Pomo																							
Total: \$21.99																							
Mashed Potatoes																							
Pasta																							
Vegies																							
Ling																							
Penne																							
Spaghetti																							
Alfredo																							
Pomo																							
Send	Print	Remove	Modify	Notify Manager @63	Back																		

v. Manager Screens

Below is the screen that the manager sees after they log in.

Logged In as: John Manager

Logout

Create New Message

From	Message	Time
Perry	Table 4 unsatisfied	4:51pm
Dan	Table 3 unsatisfied	4:21pm
Annie	Out of Onions	4:15pm
Nishy	Need help @ host stand	3:55pm
Bob	Table 2 unsatisfied	3:51pm


Reply

Delete

Below is the screen that the manager sees when they want to send a new message.

Logged In as: John Manager

Logout



Send

Back

b. User Effort Estimation

i. Scenario 1: Waiter Places an Order

1. Select table number of new table to open a new ticket

2. Select tab for item type (drinks, appetizer, entree, dessert):
3. For each dish:
 - a. Select dish under tab.
 - b. Select any specifications/modifications that are needed to make the dish properly
4. Select send to send the order to the kitchen.
5. Wait for hot food notification from chef.

Total clicks by Waiter: at least 5 clicks (dependent on size of party)

ii. Scenario 2: Starting a Dish on a Ticket

1. Select a ticket on the screen
2. Select a dish in ticket.
3. Select started for status of dish.
4. Ingredients deducted and threshold checked.
5. Select to view low ingredients (upon notification of low threshold).

Total clicks by Chef: at least 4 clicks (dependent on threshold limit).

iii. Scenario 3: Closing a Table

1. Select table number of table to open ticket for.
2. Select print check on bottom of screen.
3. Pick up the check at the printer, take it to table, and take payment.
4. Select paid button on bottom of screen.
5. Select yes button to “Are you sure?”.

Total clicks by Waiter: at least 4 clicks

Domain Analysis

a. Domain Model

i. Concept Definitions

Responsibility Description	Type	Concept
R-01: Container for the multiple dishes of the ticket. Contains the table number, price, status, and information of dishes on the ticket.	K	Ticket(Ti)
R-02: Sends the ticket from the waiter interface to the printer.	D	WaiterTicketSender (WTS)
R-03: Recognize that a message needs to be sent and sends that to the message controller (notifying host of payment)	D	WaiterMessageSender (WMS)
R-04: Sends and receives messages between different subsystems. It decodes a message that it receives to determine who to send it to.	D	MessageController (MC)

R-05: Listens for any messages sent to the host from the message controller and calls the appropriate function for the message. Listens for host action (pressing a button) which may update a table's information appropriately.	D	HostEventListener (HEL)
R-06: Holds the information for the table, including the maximum occupancy, the time the table was last sat, the status of the table, and the waiter who is serving it.	K	Table (Ta)
R-07: Listens for any messages sent to the waiter from the message controller and calls the appropriate function for the message. Listens for any waiter action (pressing a button) which may update tickets or send messages.	D	WaiterEventListener (WEL)
R-08: Sends the ticket from the waiter interface to the head chef.	D	WaiterTicketSender (WTS)
R-09: Listens for any messages sent to the head chef from the message controller and calls the appropriate function for the message. Updates the queue of tickets.	D	ChefEventListener (CEL)
R-10: Holds list of tickets that are in process.	K	TicketQueue(TQ)
R-11: Sorts the list of tickets based on the status of the ticket first (finished, not finished, in progress) and then by the order in which it was received by the chef.	D	TicketQueueSorter (TQS)
R-12: Listens for any actions from the chef and sends it to the message controller to perform the appropriate function (may be a color change on the chef's screen or sending a message to the database).	D	ChefEventListener (CEL)
R-13: Decrement the ingredients that were used for a dish that was recently started.	D	InventoryUpdater (IU)
R-14: Determines if the threshold has been met upon receipt of a starting dish message from the chef and follows the appropriate actions.	D	LowInventoryChecker (LIC)
R-15: Listens for any messages sent to the manager from the message controller and can create messages to send out to waiters (low inventory).	D	ManagerEventListener (MEL)
R-16: Recognize that a message needs to be sent and sends that to the message controller (low inventory broadcast)	D	ManagerMessageSender (MMS)
R-17: Recognize that a message needs to be sent and sends that to the message controller (message of seated	D	HostMessageSender (HMS)

table to waiter)		
R-18: Recognize that a message needs to be sent and sends that to the message controller (message that food is ready)	D	ChefMessageSender (CMS)
R-19: The container for messages used for communication between subsystems.	K	Message(Mes)

ii. Association Definitions

Concept Pair	Association Description	Association Name
Ticket ⇔ Message	A ticket is a type of message.	Child
Ticket ⇔ WaiterTicketSender	Send ticket from waiter interface to printer.	Provides Data
WaiterMessageSender ⇔ MessageContoller	MessageController receives message from waiter and sends it to corresponding subsystem	Provides Data
ManagerMessageSender ⇔ MessageContoller	MessageController receives message from Manager and sends it to corresponding subsystem	Provides Data
HostMessageSender ⇔ MessageContoller	MessageController receives message from Host and sends it to corresponding subsystem	Provides Data
MessageController ⇔ HostEventListener	MessageController forwards messages to the Host	Forwards
ChefMessageSender ⇔ MessageContoller	MessageController receives message from Chef and sends it to corresponding subsystem	Provides Data
MessageController ⇔ WaiterEventListener	MessageController forwards messages to the Waiter	Forwards
MessageController ⇔ ChefEventListener	MessageController forwards messages to the Chef	Forwards
MessageController ⇔ ManagerEventListener	MessageController forwards messages to the Manager	Forwards

ChefEventListener ⇔ TicketQueue	ChefEventListener will add a new ticket to the TicketQueue.	Updates Information
ChefEventListener ⇔ TicketQueueSorter	ChefEventListener will call the TicketQueueSorter after it modifies the Queue.	Conveys Request
TicketQueueSorter ⇔ TicketQueue	TicketQueueSorter sorts the Ticket Queue	Sorts
ChefEventListener ⇔ InventoryUpdater	ChefEventListener calls Inventory Updater to update the inventory of the ingredients of the dish that was just started.	Conveys Request
InventoryUpdater ⇔ LowInventoryChecker	InventoryUpdater calls LowInventoryChecker to check if the inventory of a recently decremented ingredient is below the threshold.	Convey Request
LowInventoryChecker ⇔ MessageController	LowInventoryChecker sends a message to the MessageController that an ingredient's inventory is low. The messageController will then forward this to the corresponding devices.	Sends Notification
MessageController ⇔ Message	Message Controller uses the message type to convey information.	Convey Information

iii. Attribute Definitions

Concept	Attributes	Attribute Description
Message (Mes)	Sender	Who sent the message
	Recipients	Who is receiving the message
	Data	Contents of message
Ticket(Ti)	Price	Price of the ticket
	Status	Updates according to status

		of each dish on it
	List of Dishes	Items in each dish
	Table Number	The table that ordered the ticket
	Time Stamp	Time of when the ticket was received by the Chef.
WaiterTicketSender (WTS)	Ticket	Ticket to be printed by the printer.
WaiterMessageSender (WMS)	Message	Holds the message object to be sent.
MessageController (MC)	List of Message	List of processed messages
	List of Message	List of messages to be processed
HostEventListener (HEL)	Message	Current message it is reading.
	List of messages	List of messages to be processed
	Message Decoder	This will take the current message, decode it and call the appropriate function.
Table (Ta)	Max Occupancy	How many people can be seated at that table.
	Status	Tells you the status of the table (seated, oncheck, ready)
	Table Number	The table's number that will be used to identify it.
	Waiter Name	Name of the waiter who is serving the table
	Time	Time the table was last sat.
WaiterEventListener (WEL)	Message	Current message it is reading.

	List of messages	List of messages to be processed
	Message Decoder	This will take the current message, decode it and call the appropriate function.
ChefEventListener (CEL)	Message	Current message it is reading.
	List of messages	List of messages to be processed
	Message Decoder	This will take the current message, decode it and call the appropriate function.
TicketQueue (TQ)	List of Tickets	Lists of ticket objects
TicketQueueSorter (TQS)	TicketQueue	Holds the list of the ticket objects
	SorterFunction	Sorts the Ticket Queue based on the status of the ticket and the chronological order it came in.
ChefEventListener (CEL)	Message	Current message it is reading.
	List of messages	List of messages to be processed
	Message Decoder	This will take the current message, decode it and call the appropriate function.
InventoryUpdater (IU)	Ingredient	Name of the ingredient you wish to update.
	Current Quantity	The current quantity of a certain ingredient
LowInventoryChecker (LIC)	threshold	the limit at which the ingredient is considered low in stock
	current quantity	The current quantity of a certain ingredient

ManagerEventListener (MEL)	Message	Current message it is reading.
	List of messages	List of messages to be processed
	Message Decoder	This will take the current message, decode it and call the appropriate function.
ManagerMessageSender (MMS)	Message	Holds the message that the manager wishes to send.
HostMessageSender (HMS)	Message	Holds the message that the host wishes to send.
ChefMessageSender (CMS)	Message	Holds the message that the chef wishes to send.

iv. Traceability Matrix

Use Case	MES	TI	WTS	WMS	MC	HEL	Ta	WEL	CEL	TQ	TQS	IU	LIC	MEL	MMS	HMS	CMS
UC - 1	x	x	x	x	x	x	x	x								x	
UC - 2	x	x	x		x			x	x	x	x	x	x	x	x		x
UC - 3	x	x	x		x			x	x	x	x	x		x			x

b. System Operation Contracts

Name	Closing a Table
Responsibilities	Print out the check to give to the table, and take the payment for that check so the party can leave.
Use Cases	UC-1
Exceptions	None
Pre-Conditions	Check is viewable on-screen and there is a print check button on the bottom of the screen that will print the check at the printer.
Post-Conditions	The table should no longer be shown as active(green) on the Waiter's screen. The Host's screen should be updated to include a on check status for the

	departing table.
--	------------------

Name	Low Inventory
Responsibilities	To maintain a record of the stock of ingredients and inform staff when low on one. Keep an organized queue of tickets that are in the processes of being made.
Use Cases	UC-2
Exceptions	None
Pre-Conditions	The steady database would have the information about the quantity of items in the inventory. At the start of each day, the head chef will load the data from that database into the inventory/order database. This will allow him/her to have the most updated information about the inventory and accordingly have accurate responses for stock.
Post-Conditions	Each time the head chef starts a dish on a ticket, the system should be able to deduct the correct amount of ingredients that the dish uses from the number listed in the inventory (inventory/order database). If the items of the inventory fall below a specified threshold, the system should alert both the Manager and the chef and give a color coding on the Waiter's menu screen that is different for dishes that use those ingredients. The Manager can send a message to all of the Waiters about the low amount of ingredients based on his/her decision

Name	Placing an Order
Responsibilities	To place and complete an order.
Use Cases	UC-3
Exceptions	None
Pre-Conditions	There must be a button on the Waiter's screen to create a new ticket. Also, the customer must place an order. .
Post-Conditions	The order has been successfully completed and delivered to the customer.

c. Mathematical Model

Generating Host's Table Queue:

Algorithm for how the host list of tables is generated. The host screen will show ready tables (tables that are open) in one list and not ready tables (tables that are seated or dirty) in another, both on one screen. Below is pseduo code for the algorithm we will use for how to order these tables in the appropriate list.

Pseudo Code:

At start of day:

```
allTables = getListofTableNumbersFromDataBase()

readyTables = new TableList();
notReadyTables = new TableList();

for each table in allTables
    table.status=ready;
    readyTables .add(table);
```

Throughout the day:

The host event listener will listen for the host clicking the screen or an on check message from a waiter. The event object will hold the information about that action.

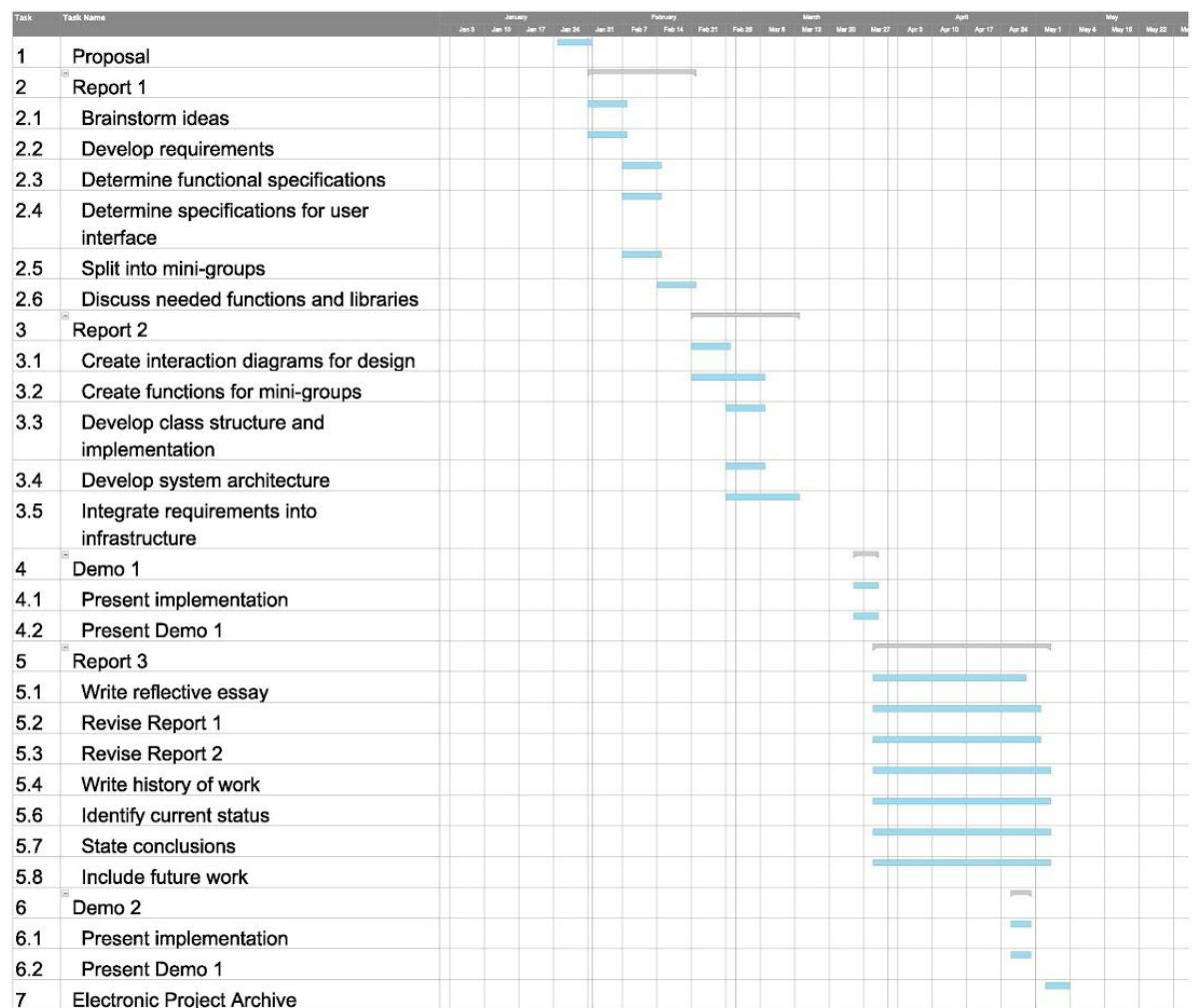
```
hostEventListener( Event e){
    currTable = allTable[e.tableNumber];
    oldStatus = currTable.status;
    if(e.type == click){ //if host is changing status
        if(oldStatus==e.newstatus){
            return;
        }
        if(oldStatus==seated){
            handleError();
            return;
        }
        if(oldStatus == ready){
            readyTables.remove(currTable);
            notReadyTables.add(currTable);
        }
        else{
            readyTables.remove(currTable);
            notReadyTables.add(currTable);
        }
    }
}
```

```
    }
    currTable.status = e.newtablestatus;
}
else if(e.type == waiterMessage){ //if waiter sent a message
    if(oldStatus!=seated){
        handleError();
        return;
    }
    currTable.status = e.newtablestatus;
}
else if(e.type== managerMessage){
    makeNotification(e);
}
//update the screen
redrawHostScreen();
}
```

Plan of Work

a. Project Management

The objectives of our project throughout the duration of the semester have been depicted in the below Gantt Chart. Each task in the Gantt Chart is a milestone in the project, and each decimal numbered task is a subtask of the main task below. The chart is updated to the best of our knowledge in regard to the responsibilities and deadlines we have been given thus far and is subject to change.



b. Product Ownership

We decided to break the larger group up into mini groups of 2 people to improve the efficiency of our group. We chose the mini groups based on team familiarity and skill balance. We still worked as a whole group for certain parts of the project.

c. Breakdown of Responsibilities

Team Head Chef:

Mini-Group Members: Annie Antony and Nishtha Sharma

Completed Tasks: Created a first draft for what the head chef interface will look like.

Current Tasks:

- Understand and plan the organization of database for the ticket orders and inventory.
- Research how to create a GUI for the head chef screen.
- Understand how the interface of the head chef will influence the database linked to it.

Future Tasks:

- Make the Head Chef GUI.
- Create functions to organize the order of the tickets received by our color coordination set up.
- Create an event listener for the tickets which will receive tickets from the waiter and use the above function to sort them.

Team Waiter:

Mini-Group Members: Christina Segerholm and Athira Haridas

Completed Tasks: Created a first draft for what the waiter interface will look like.

Current Tasks:

- Plan out how the menu database is going to look like on the inside.
- Research how to make waiter GUI using java swing library.

Future Tasks:

- Make the Waiter GUI.
- Make functions for sending messages to the message controller.
- Make event listener for waiter which should listen for any buttons on the screen pressed or any messages being sent to it.

Team Host:

Mini-Group Members: Emma Roussos and Christina Parry

Completed Tasks: Created a first draft for what the host interface will look like.

Current Tasks:

- Research how to create a GUI for the host interface.
- Comprehend the interactions between the host and database.

Future Tasks: Add:

- Create Host GUI.
- Implement functions for HostMessageSender.
- Implement functions for HostEventListener.

Whole Team:

Completed Tasks: Created an outline (initial draft) of the overall system.

Current Tasks:

-
- Develop a complete understanding of the interactions between the different subsystems in the software application.
 - Redesign the initial diagram of the overall system with our new understandings of the functionalities.
 - Research how to create a GUI for the manager interface.

Future Tasks:

- Gradually develop the shared infrastructure library piece by piece.
- Develop the databases needed overall.
- Develop controllers to manage data flow into and out of databases.
- Create a message controller to allow communication between all subsystems.
- Create the manager user interface.

References

Previous Group Projects:

- Group 2 Spring 2012
- Group 4 Spring 2014
- Group 3 Spring 2015
- Group 11 Spring 2012

Employee References:

- Emma Roussos: Server/Hostess at Colonial Diner
- Christina Segerholm: Server/Hostess at Carrabba's Italian Grille

http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf