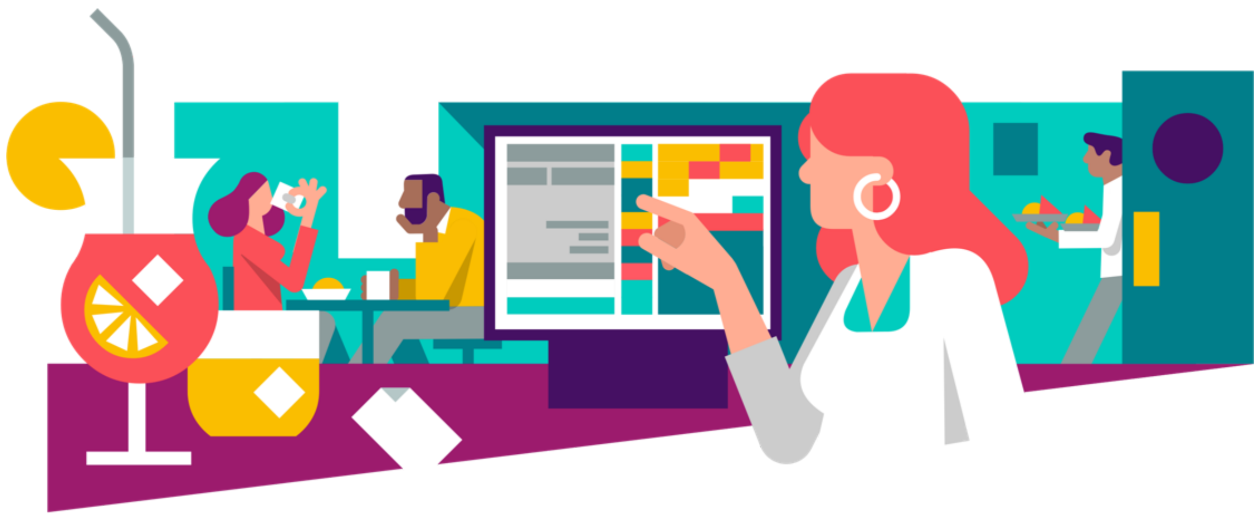


Robo Ramsay

(Name pending)

Software Engineering

Team 6
2/23/2021

https://dpmon1.github.io/Robo_Ramsey

Akira Brown, Esteban Salazar, Alan Chacko, Suryansh Singh, Adrian Mah, Parth Vora, Shrey Joshi, Raghav Gopalakrishnan, Kunj Desai, Ray Chau

Individual Contributions

Topic	Akira	Esteban	Alan	Suryansh	Adrian	Parth	Shrey	Raghav	Kunj	Ray
Customer Problem Statement	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Decomposition into Sub-problems	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Glossary of Terms	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Business Goals	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Enumerated Functional Requirements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Enumerated Nonfunctional Requirements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
User Interface Requirements	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Use Cases	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
User Interface Specification	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
System Architecture	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%
Project Management	10%	10%	10%	10%	10%	10%	10%	10%	10%	10%

Table of Contents

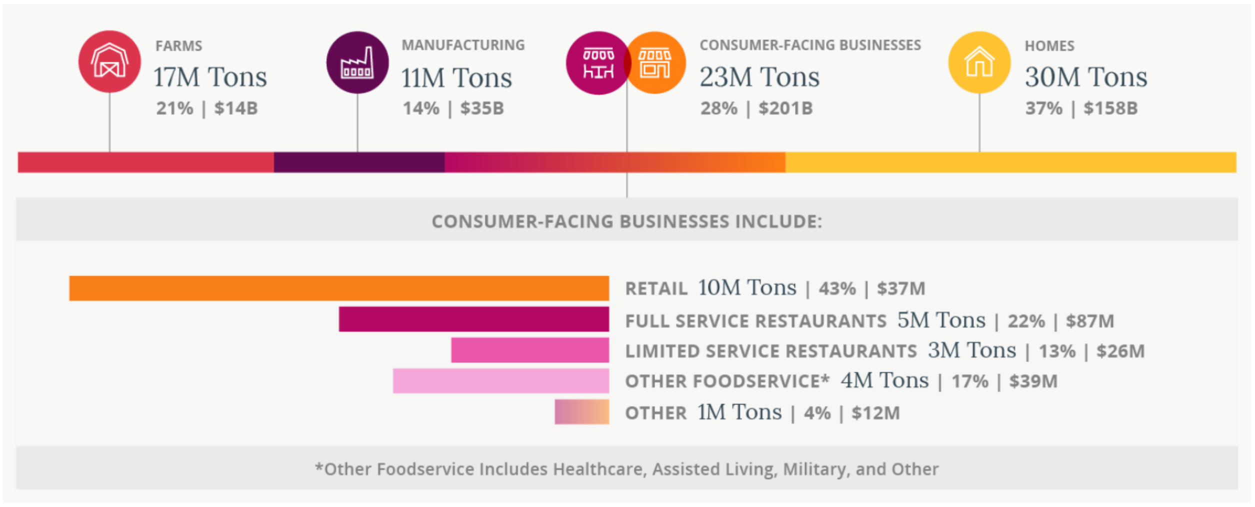
Section 1: Customer Problem Statement

Problem Diagnosis

Most restaurants have leftover food by the day's end, and sometimes there are menu items that have not been ordered even once throughout the day. The Green Restaurant Association found that the average restaurant may produce up to 100,000 pounds of food waste in a year. Ingredients that could have been consumed are instead disposed of because they expire. This results in most of the food waste ending up in landfills, where it rots or ferments into greenhouse gases. Additionally, this wastage also contributes to the proliferation of chemicals or undesirable bacteria that are a danger to public health and the environment if left unchecked. The food waste also represents lost revenue. Money that could have been spent to make revenue-generating dishes is instead spent on unused ingredients for less popular dishes. It is even possible that state regulations issue fines for food waste violations.

Furthermore, items that are less popular are often kept on a restaurant's menu, resulting in funds being wasted on buying ingredients for these items. These funds could have been instead allocated to ingredients for more popular items. An application that could automatically identify these less popular items would be desirable.

Another problem. There tends to be a disconnect between customers and restaurants on what dishes are available, especially when dining in. A method of being able to convey this information almost instantaneously would help streamline the link between customer, restaurant, and menu item so that the customer will not get their hopes up on ordering something that is unavailable.



Source: *Food waste challenge*. (n.d.). Food Waste Recycling Analysis, Reduce Food Waste & Food Recovery - ReFED. https://refed.com/food-waste/the-challenge/?sort=economic-value-per-ton

All these issues could be avoided if there was proper analysis of the restaurant's clientele and the market for ingredients. This can be easily done by a computer with simple sorting algorithms and the use of a database, and then graphed to help the manager make better informed choices.

1.1: Problem Statements

Restaurant Customer Statement

Way back before COVID dining restrictions were put in place, I loved going out to eat at different restaurants with my friends. It was usually an enjoyable time, but I found that I often did not know what their recommendations were, especially if they do not signal any popular dishes on the menu. Due to that, I often take too long looking at every section of the menu, and my friends often tease me about it but I just never know what to get! This only gets worse when the restaurant has such a large menu that is separated by multiple pages with little to no pictures. This can lead me to ordering something and then not liking how it tastes, which makes me feel bad because growing up I was told to not waste food, and even now I don’t like wasting food because it’s my wallet that suffers! When I eventually order what I want, the waiter sometimes comes back telling me that due to specific ingredients having run out, the specific dish I wanted can not be made. This becomes a big hassle as the waiter offers me something to replace it that would not be the same or if I wanted to order something different. What’s worse is if some of my friends and I order the same dish, we can get greeted by the waiter by them mentioning how there is not enough ingredients to fulfill our order, which some of my friends would go “How do you not know that there isn’t enough of [the dish’s name or ingredients to make it]?!”. This makes the dining experience less enjoyable because the information is not readily available right away upon ordering, because at least we’d be able to quickly order something different and have all the food come at roughly the same time, and not have one or more people wait even longer.

I also live in a part of the United States where COVID has been having declining numbers of cases for a while, so my friends and I have been going out to restaurants occasionally, but following guidelines of using masks. However, I am wary of going to restaurants now because of any potential exposure through the waiter coming to our table even when they have a mask on, since I have family members that can be at risk. I feel like if anything were to help with any of these issues I am experiencing, it would be a virtual or online version of a restaurant’s menu that lets me order food to my table when dining in. Not only would it solve my woes of potentially getting COVID and spreading it to family members, but it can probably do so much more! For example, I tend to have trouble ordering something so maybe if that same menu can list recommendations or include images of the actual dishes that can be made, it would save me some trouble and time. Also, with a menu being through an electronic medium, it could also let my friends or I know what can’t be made due to the restaurant running out of something instantly.

Restaurant Chef Statement

I enjoyed working with my team in the kitchen, it was lively, with a bit of casual banter. As a team of cooks, good communication was of the utmost importance. We would communicate what orders we were working on as well as keeping check of stock so that we could notify the manager later on so that they can order more, it's never a good thing to run out of ingredients during a busy day. Ever since COVID struck, our restaurant restricted the amount of kitchen staff allowed in the building at any given time. This generally made work go slightly slower since more people makes prep work and cooking easier especially if there are multiple stations to be monitored. Due to the restrictions, there are times I couldn't be in the kitchen and as a result, some of my fellow yet less experienced cooks would either underestimate or overestimate the current stock of ingredients. This leads to running out of ingredients while we still have to put out orders, or ending up with too much of an ingredient that we have come in fresh. If we keep a very high amount of fresh ingredients in the fridge and constantly put in more stock of it, eventually we could have the whole bunch of fruits or vegetables wilting or rotting. Leaving perfectly edible ingredients in the fridge or freezer until it goes bad wastes the restaurant's resources and storage space; efficiency and freshness are the virtues of a chef, not being able to have both sucks. It's a big pain when in the middle of work, we receive an order for a dish that we later find out that we don't have enough ingredients to fulfill the recipe's measurements. Even worse is the dilemma of "Do we try to finish it with what we have and have the waiter bring it out? Or do we call in the waiter so that they share the news with the customers?". Both situations can end up badly especially if the customers in question are not in the mood to be civil. There have been times where they demand a free dessert or have that specific dish free on the check, and that never ends well with the manager.

A way to potentially fix the issue of inventory management would be to have a virtual or electronic list of it, an ingredient tracker one could call it. This would greatly improve efficiency between different shifts of work, and solves the potential issue of less experienced cooks miscounting or misjudging the amount of ingredients left. Not only will it help with inventory management, the process of the manager buying ingredients to maintain stock will be easier. Not only that, but we could have another part of the system that works with the ingredient tracker to feature dishes and the measurements required to make them, so that stock could be updated as we are making the dishes that are being ordered.

Restaurant Manager Statement

As a manager working in a restaurant setting, the roles and responsibilities that I handle include purchasing items, keeping inventory, and making sure all components of the restaurant are functioning efficiently. Sometimes I have cooks telling me that we need more of an ingredient, but oftentimes giving a higher estimate than what would be needed, which leads to wasting funds as well as wasting perfectly good ingredients. Using an application that would tell me when the kitchen is running low on certain items and when they need to be ordered, I would be able to ensure that the customers are given the restaurant experience they should be expecting while also keeping the kitchen in working order. I would also be able to track usage of each ingredient in each dish to maximize product output with less waste. This helps me make necessary cuts to certain dishes and re-evaluate which dishes might be a better fit for the restaurant.

The biggest issue a customer runs into at a restaurant is the wait times. Having an application track the time an order is placed, the time it takes for a chef to prepare the dish, and the time it takes for the wait staff to deliver the dish, customers will be able to have a very accurate picture of how long it should take for their food to arrive. I can also pinpoint where the staff is working the most efficiently and where they might be able to improve. For example, if I see that the same dish takes 20 minutes longer to prepare during the day vs. night, I am able to see whether it might be an issue of staffing, lack of prepared ingredients, etc. This ensures that I am tending to the staff's needs, so that in turn, they can tend to the customers' needs.

Restaurant Waiter Statement

The biggest responsibility I have as a waiter is to provide excellent service to the customers by keeping them happy and satisfied. My job description includes taking customer orders, delivering food and beverages, and making menu recommendations.

As I take customer orders, I run into problems with stock and inventory all the time, and it's a terrible situation to be in. The customer orders an item that they see on the menu. I go into the kitchen to convey the message. The kitchen tells me that they've run out of stock to make the item. I, as the messenger, have to take that news and explain it to the customer as best I can. Hearing something like that wouldn't make anyone happy, so as a waiter I have already failed in one of my biggest responsibilities: keeping customers happy and satisfied. I'm put in an embarrassing situation, the customer is disappointed, and the restaurant risks losing business. The smart menu fixes this problem by "greying out" (making unavailable for selection) items on the menu that do not have enough ingredients for an order to be filled. As a waiter, I know that I won't ever have to be put in a situation where a customer might order an item that isn't available at that moment, which helps me do my job better and keeps the customer happy.

The restaurant welcomes in new customers almost on a daily basis, most of which have very little idea on what's hot on the menu. Popular menu items can consistently change, and it's often difficult for me to keep track of what most customers are ordering, especially when there are multiple waiters. Beginner waiters at the restaurant also have very little experience, and may not be able to answer customer questions and give them perfect recommendations. Noting down how many orders of a menu item was done each day with pen and paper can get messy and difficult to handle for both the waiters and the manager. A smart menu system where the menu item orders are easily counted and automatically arranged according to the popularity of the dish easily fixes all of these issues. Communication between all of the waiters and the manager in this regard doesn't need to be a major priority anymore, saving valuable time for the restaurant. Novice waiters don't need to be embarrassed when asked for recommendations anymore. Overall, the restaurant is put in a much better position.

1.2: Decomposition into Sub-problems

Based on our problem statement, the two overarching problems we are trying to solve are decreasing food waste and increasing the efficiency of restaurants. These two problems can be further decomposed into sub-problems.

Food Waste

The first problem we are trying to solve is decreasing the amount of food/ingredients that is wasted by restaurants. This problem can further be divided into subproblems, each of which is a potential source of food waste. The first subproblem relates to the fact that dishes that are unpopular take up valuable space in the ingredients storage. These ingredients are wasted and must be resupplied in case the customer orders the dish, sometimes perpetuating the wasting of ingredients. Additionally, the second subproblem is that without knowledge of the inventory, ingredients are also wasted from popular dishes when management is unaware of the exact amount of ingredients currently in possession. These two subproblems will be directly addressed by features in our application.

Increasing Efficiency

In addition to decreasing food waste, the other problem that we are trying to solve is increasing the efficiency of the restaurant. Increasing efficiency can be broken down into several subproblems, each of which will be addressed by a particular feature in our application. The first of these subproblems is the fact that currently, customers at a restaurant often do not know when a dish is not available due to lack of ingredients unless explicitly told so by a waiter. This can result in longer wait times for customers. Another subproblem is increasing the speed of ordering. The average wait time in a restaurant is approximately 23 minutes (Full Service Restaurant News, 2013). Reducing this time would be desirable. The third subproblem relating to efficiency is reducing the time required to address ingredient shortages in the restaurant pantry. The fourth subproblem relating to efficiency is allowing the manager to easily have a bird's eye view of all orders in the restaurant to allow them to easily detect any bottlenecks in service. Each of these subproblems will be addressed by a particular feature or functionality in our application.

The Solution

In order to fix the issues revolving around the availability of a certain menu item, we have proposed to introduce a Smart Menu which will end up helping both the restaurant and the customer. The Smart Menu will, like a normal menu, keep note of the daily specials as well as highlight new dishes. Along with this, it will also keep track of the restaurant's ingredient stocks. The Smart Menu will also keep track of the popularity of all the dishes, using it to provide customer recommendations and rearrange the menu to be relevant. Another function will be to provide the manager with insight on long term trends. This will help the restaurant greatly in figuring out how many of each ingredient they should order, and hence reduce the unnecessary wastage of food. This will save the restaurant a good deal of money as well as make them more efficient.

Such an automated version of the restaurant process will benefit the customers, employees and the management at the same time. The inventory would provide the restaurant staff with live usage data. This will allow the restaurant management to more accurately order quantities and lessen the amount of food wasted at the end of a business period. In addition, the tracking of inventory using an automated software - as opposed to manually keeping inventory - not only helps lessen the employees' workload, but also maintains the data virtually. This allows for our application to use it to predict trends. This way it can better curate menu specials and generally refine the menu. The refined and updated menu will in turn benefit the customer as they will be offered more relevant food that's guaranteed to be in stock.

We can measure these changes using concise metrics based on data gathered from our software. The economic impact of our application can be measured by calculating the difference between the value of the amount of food thrown out prior to our automation and the value of the amount tossed afterwards. In addition, the time saved by the employees who would manually check inventory and track usage at the end of the day could be used elsewhere. We could measure how relevant the menu is by finding the distribution of different plates ordered. Menus with a wider distribution leave dishes at the end that receive nearly no orders indicating it's a poor menu choice. On the other hand, menus with a more even distribution - where all plates get at least some traction - would indicate a more relevant menu. The change in that distribution would be our menu relevance metric, which a business can use to decide whether to replace a dish or set of dishes.

Suppose a customer ordered a meal that requires a particular ingredient and is not currently in the restaurant's inventory. In a restaurant without a Smart Menu, the customer would order the meal and be told several minutes later that the ingredient is not available. The customer would then have to order something else or be offered a replacement ingredient. These minor annoyances could negatively affect the customer's opinion on the restaurant. On the contrary, a restaurant with a Smart Menu would keep a live inventory that provides updates on the availability of these ingredients. Management would be notified about the shortages of certain ingredients well ahead of time and potentially never have customers or employees experience shortages.

Suppose that in another situation, a new customer is unsure of what to order on their first visit. The Smart Menu would provide the customer with an updated list of popular items as well as the availability of items to customers and employees alike. Incoming orders would result in updates to the availability of ingredients, which would in turn result in updates to the smart menu.

So, we can build an app that will incorporate all these individual solutions. It can be a convenient tool to gather, analyse and understand the relevant data points, as well as provide an interactive menu. It will keep track of ingredients, and thus assist with minimizing waste. Large restaurant chains already have apps which are used to order food, review orders, and collect data. Our application will allow smaller restaurants and restaurant chains to easily achieve similar granular functionality, along with better utility towards the restaurant staff.

Plan of Work

The app will consist of four main business functionalities. (Note: some of these functionalities are groupings of highly related functionalities.)

1. Ordering: The customer will have the ability to select and order food from the menu, pay for it, and receive a receipt. Upon ordering, they will be able to view the status of each dish (not prepared, being prepared, prepared) in their order and will receive a notification when dish status changes. This feature can be used alone.

2. Dish analysis: The manager will be able to view the popularity of each dish. Popularity is measured as the amount of times a dish is ordered within a particular time interval that can be selected by the user - week, month, 3 months, or year. The customer will receive recommendations based on which dishes are the most popular. This feature cannot be used alone and has to work in combination with 1.

3. Ingredient tracking: The manager will be able to input the quantities of each ingredient available into an ingredient management system. This system will update the amount of each ingredient available based on food cooked throughout the day. This will be done by communicating with the ordering and smart menu features. The manager will also be able to add new ingredients, search for ingredients based on keywords, and filter the ingredients list based on availability and type. The manager will also be notified when there is not enough of a particular ingredient to cook a dish This feature can be used alone.

4. Smart Menu: The menu of the restaurant will be a “smart menu” that updates according to ingredient availability. Items that do not have enough ingredients to make at least 6 of them will be “greyed out” (unavailable for selection) on the menu. The menu will also include sort, search, and recommendation features to allow customers to quickly select a dish to order.. This feature cannot be used alone and has to work in combination with 3.

The following sub-teams will develop each functionality:

1. Ordering process - (Esteban, Akira)
2. Dish analysis - (Alan, Suryansh)
3. Ingredient tracking and search - (Adrian, Parth, Shrey)
4. Smart menu - (Raghav, Kunj, Ray)

In order to implement the project, the following features will be reused from the TurboYums app produced by a previous team:

1. Customer Interface - We will reuse the order, payment, and receipt functionalities for the customer. To this we will be adding an order status feature that allows the customer to view and be notified of changes in the status of each dish. We will also add a “smart menu” functionality that “grays out” dishes that are not available and recommends dishes based on popularity.
2. Manager and Waiter Interfaces - We will reuse the order-queue feature that allows the manager and waiters to view incoming orders as a queue. For the manager, we will reuse the menu editing (add, remove, and edit item) feature. To this we will add an ingredient tracking system that lets the manager keep track of and search for ingredients and updates based on food cooked throughout the day. The manager will also be able to view the popularity of each dish and be notified when there are not enough ingredients to produce a certain dish.
3. Chef Interface - We will reuse the order-queue feature for the chef. To this we will add the ability for the chef to update the status of each item in the queue of incoming orders as they cook. The chef will also be able to view and edit the ingredient-tracking system.

The project itself will be software-only. We won't need any specialised data or specific hardware. We will only need some computer that meets the minimum requirements (suitable processing power, RAM/ROM, OS and a network connection to user devices) to run the server application, and each user will need a suitable device as well for the user application.

We will be using JavaScript mainly for the application. We will be using a few libraries and frameworks in this application:

1. PostGreSQL: This is a relational database management system. That means it is a system that manages a database. The "relational" part of its name means that stored data must all be in the same structure. Using this, we will create and use a database of accounts, roles, store preferences, collect logs and analytic data, manage an inventory of stock, and also manage a database of recipes.
2. Express: This is a framework for javascript that was built specifically to facilitate development of applications that use Node.js. It is used to create and manage servers. Using this, we will make code that allows us to host a server on the machine that runs the application.
3. React Native: This is a cross-platform mobile application framework. It incorporates the React.js library. It is built to allow supported platforms (mobile phones - both iOS and Android, tablets, desktops, etc.) to render the same React code easily. Using this, we will

code an application that resides on the user's device, which the user can use to interact with the server.
4. Node.js: This is a JavaScript Runtime Environment. This means that Node.js is software that Javascript can use to run. This is common in JavaScript applications. Using this, we will run our code.
5. Stripe API: Given enough time, we may use this API to process customer payments. However, as this app is a POC, it does not need to actually process payments.

Team Description and Strengths:

1. Akira Brown - JS
2. Esteban Salazar - JS
3. Alan Chacko - JS, Node.js/Express, React Native, SQL
4. Suryansh Singh - JS, SQL
5. Adrian Mah - JS
6. Parth Vora - JS
7. Shrey Joshi - JS, React Native, AWS, SQL
8. Raghav Gopalakrishnan - React Native, AWS, JS,
9. Kunj Desai - JS
10. Ray Chau - JS

1.3: Glossary of Terms

Menu - A list of food available to be prepared, cooked, and presented, including pictures and ratings.

Smart Menu - Menu available on phone/website that automatically “greys out” dishes that are unavailable for selection by communicating with the ingredient tracker.

Dish - An item on the smart menu.

Grey Out - To make an item unavailable for selection on the Smart Menu

Ingredient - Any food or substance, excluding water, that is combined to make a particular dish

Chef - Makes food that is requested by the customers.

Customer - Someone that comes to the restaurant as a guest to be waited on and served food.

Manager - Supervises restaurant to make sure everything is going smoothly. The person who is ultimately responsible for both the customer experience and the restaurant inventory.

Waiter - Delivers food to customers. Can take orders if the customer does not have a phone.

Order status - The status of each order (not complete, complete).

Dish status - The status of each dish within an order (not prepared, being prepared, prepared).

Order queue - A feature that displays a queue that shows the orders and corresponding order status of all customers in a restaurant.

Ingredient tracker - A feature that lets the manager/chef enter the amount of ingredients required for a particular recipe, update the quantity of each ingredient available, and search for ingredients. This tracker updates the amount of ingredients available throughout the day as food is ordered.

Dish popularity - The number of times a dish is ordered within a particular time interval (week, month, three months, or year).

Menu Editor - A feature that allows the user to add, remove, or edit items on the menu. Also displays the popularity of each dish.

Manager Account - An account that has access to the order queue, ingredient tracker, and menu editor.

Chef Account - An account that has access to order queue and can change the status of each dish within an order and ingredient tracker.

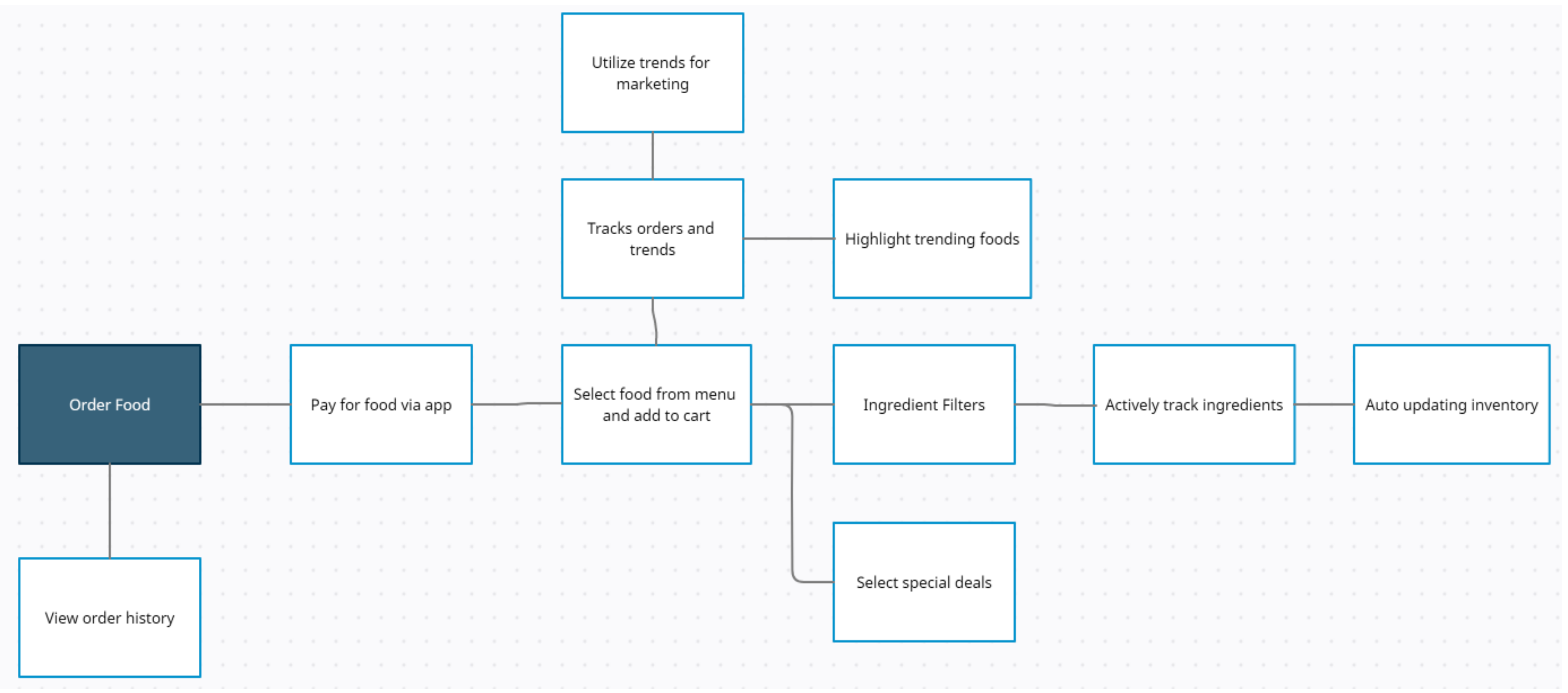
Customer/Waiter Account - An account that can order food, pay for the order, and view the status of each dish within an order.

Restaurant Automation - Process of automating restaurant tasks and functionalities that would otherwise require employees to do.

Section 2: Goals, Requirements, and Analysis

2.1: Business Goals

In order to successfully implement our proposed system to automate restaurants the team would have to first create a “smart menu” which would link the automated process from the consumer all the way to management. This smart menu would, at the least, allow patrons to browse, order, and pay from the smart menu system. From there, the team gets the goal of giving the smart menu the ability to maintain a database of the restaurant’s inventory. This database would then need to be able to be edited by manual input or changed automatically based on orders placed so that we can have the information to keep inventory. We also want to be able to keep track of the estimated quantity of ingredients used as well as how much of a dish is used in an order. With the database of ingredients patrons should also be able to filter the menu by ingredient in various ways such as “preferred” or “allergic”. After the implementation of the database has been successfully completed, the smart menu can begin tracking trends based on orders to find what people like to roder, when they order it, how they want it cooked, and how many ingredients are used by the restaurant in one period of time. With this data tracking, the smart menu can then be used as a marketing tool which can boost specials on the menu, help target ads, and provide a “popular foods” page to help new customers order with ease.
Tree diagram:



2.2: Enumerated Functional Requirements

Note: Priority 5 is highest, 1 is lowest.

A. Ordering

Identifier	Priority	Requirement
REQ-1	5	The application's customer interface will be a menu that lists and facilitates ordering available dishes.
REQ-2	2	The application's customer interface should be able to request a waiter.
REQ-3	4	The application's customer interface should display current order status.
REQ-4	3	The application's waiter interface should display all orders.
REQ-5	4	The application's waiter interface should display the customer menu.
REQ-6	3	The application's waiter interface should allow them to order on behalf of customers
REQ-7	4	The application's manager interface should allow them to view and edit the status of all orders in the restaurant.
REQ-8	3	The application's chef interface should allow them to view and edit the status of all orders in the restaurant.
REQ-9	1	The application should show estimated time for meal to arrive

B. Smart Menu (Note: These are functionalities that extend the menu beyond simply displaying and allowing selection of items, i.e. a dumb menu)

Identifier	Priority	Requirement
REQ-10	4	The application's customer interface will not allow customers to select items that are not available or possible to make.
REQ-11	3	The application's customer interface should have a category for popular dishes in the menu. (as measured by order frequency)
REQ-12	5	The application's manager interface should allow them to edit the digital menu.

REQ-13	1	The application’s manager interface should display data on popularity and trends.

C. Ingredient Tracking (Inventory)

Identifier	Priority	Requirement
REQ-14	5	The application’s manager interface will display an alert if it’s not possible to make a dish due to insufficient ingredients.
REQ-15	3	The application’s manager interface will display an alert if a kitchen ingredient is below a certain threshold.
REQ-16	5	The application’s manager interface should allow them to view and edit the ingredient count manually.
REQ-17	4	The application’s chef interface should allow them to view and edit the ingredient count manually.
REQ-18	4	The application should maintain and allow the editing of an inventory of the quantity of each ingredient available.

D. Miscellaneous

Identifier	Priority	Requirement
REQ-19	5	The application’s chef interface should allow them to summon a specific waiter or manager.
REQ-20	4	The application should log usage data.

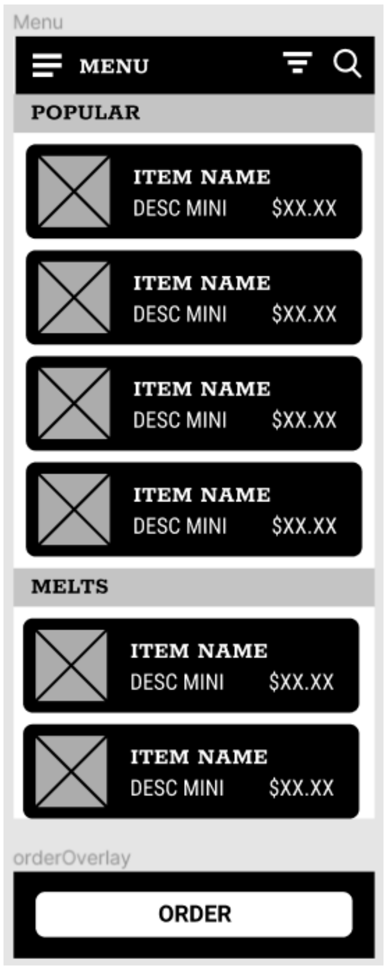
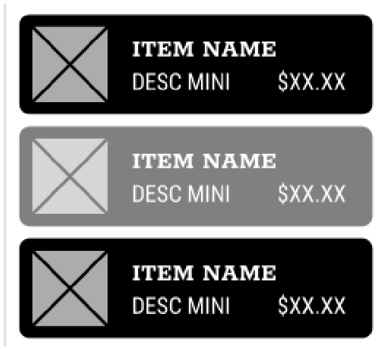
2.3: Enumerated Nonfunctional Requirements

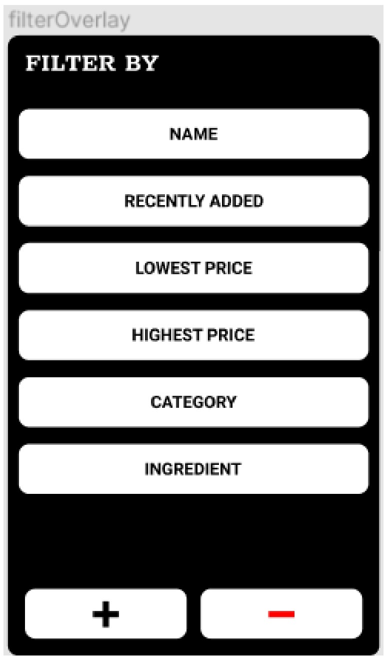
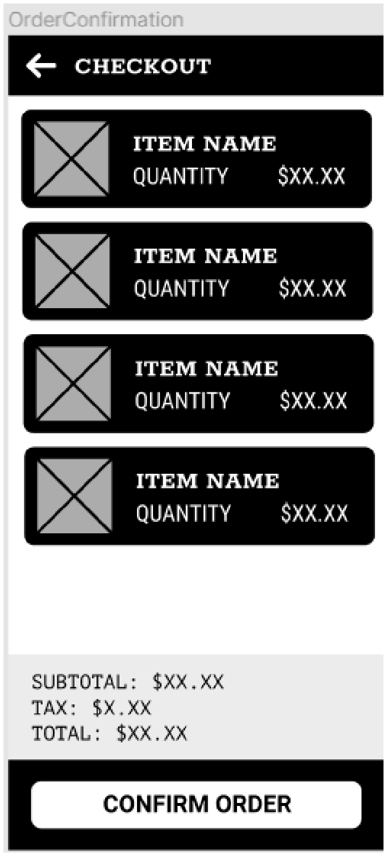
Note: Priority 5 is highest, 1 is lowest.

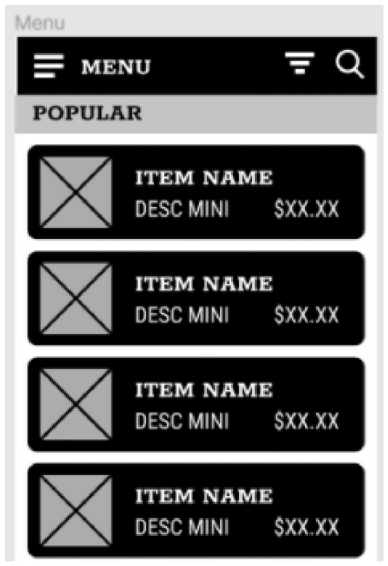
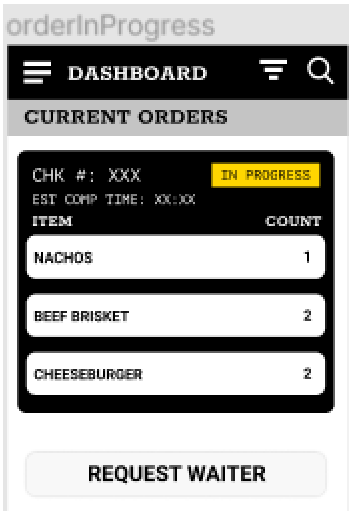
Identifier	Priority	Requirement
REQ-21	4	The application interface should be capable of functioning on older devices.
REQ-22	4	The application should authenticate all orders using the OAuth authentication protocol and also implement AES encryption in between and within subsystems..
REQ-23	3	The application should be accessible from any device with at least 1 mbps internet access.
REQ-24	2	The application should be able to efficiently use varying levels of processing and/or storage, to service varying levels of loads, and hence be able to scale in the cloud.
REQ-25	5	The application should be accessible enough to satisfy WCAG 2.
REQ-26	3	The application should provide a large text option and be tested with common text-to-speech providers to assist visually impaired users.
REQ-27	4	The application should be responsive, being able to respond to user input within 50 milliseconds or less.
REQ-28	5	The application should be able to provide a user with an easy and non-labor-intensive way of getting their job done, within 5 clicks.
REQ-29	3	The application should be aesthetically pleasing with eye-catching graphics without being overwhelming.

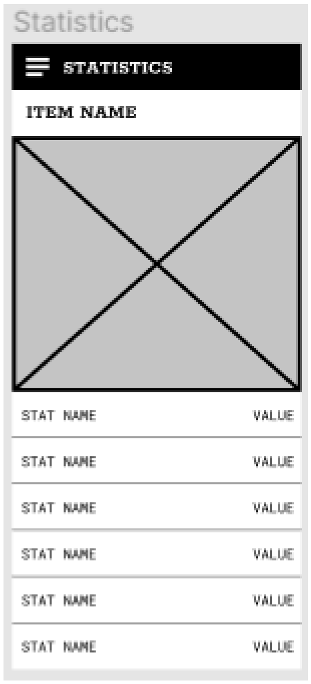
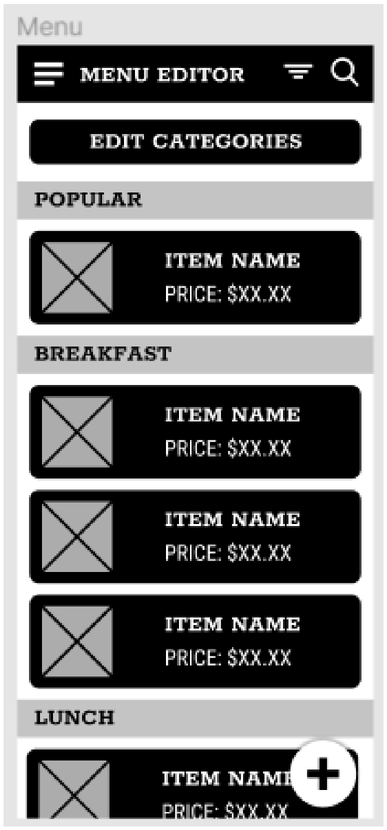
2.4: User Interface Requirements

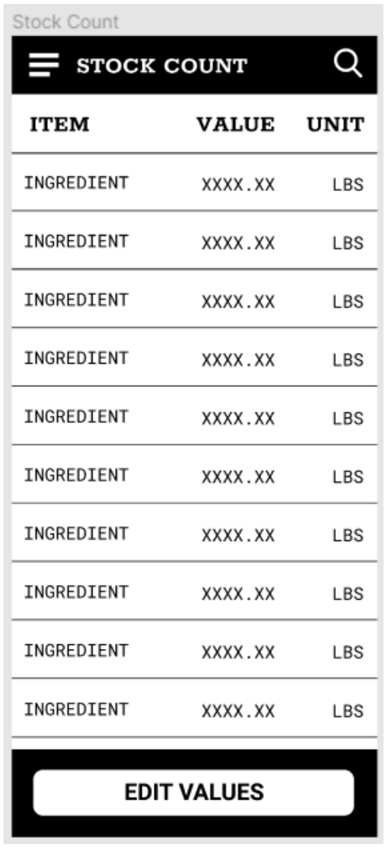
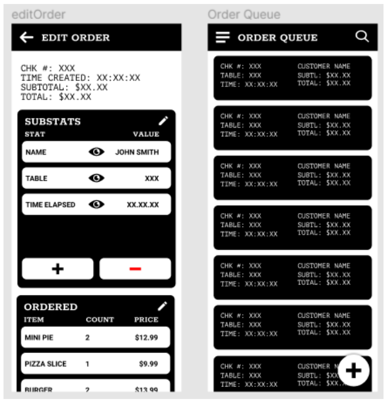
Note: Priority 5 is highest, 1 is lowest.

Identifier	Priority	Requirement	Image
REQ-30	5	The customer interface should have a menu that lists all the items that the restaurant offers.	 Menu MENU POPULAR ITEM NAME DESC MINI $XX.XX MELTS orderOverlay ORDER
REQ-31	2	The customer interface should obscure items that are no longer available.	

REQ-32	3	The customer interface should have a section in which the user can filter the results from	 filterOverlay FILTER BY NAME RECENTLY ADDED LOWEST PRICE HIGHEST PRICE CATEGORY INGREDIENT + –
REQ-33	5	The customer interface should have an order page listing the customer's selections and permitting payment	 OrderConfirmation CHECKOUT ITEM NAME QUANTITY $XX.XX ITEM NAME QUANTITY $XX.XX ITEM NAME QUANTITY $XX.XX ITEM NAME QUANTITY $XX.XX SUBTOTAL: $XX.XX TAX: $X.XX TOTAL: $XX.XX CONFIRM ORDER

REQ-34	4	In the customer interface, there should be a popular item list which lists items that help customers see which items are popular according to the analysis on the popularity of food items.	 Menu MENU POPULAR ITEM NAME DESC MINI $XX.XX ITEM NAME DESC MINI $XX.XX ITEM NAME DESC MINI $XX.XX ITEM NAME DESC MINI $XX.XX
REQ-35	3	In the customer interface there should be a tab where a customer can request an additional waiter, utensils, and how long it will take their order, etc.	 orderInProgress DASHBOARD CURRENT ORDERS CHK #: XXX IN PROGRESS EST COMP TIME: XX:XX ITEM COUNT NACHOS 1 BEEF BRISKET 2 CHEESEBURGER 2 REQUEST WAITER

REQ-36	5	The manager interface should visualize the popularity and trends of the items. (The X image represents a chart)	 Statistics STATISTICS ITEM NAME STAT NAME VALUE STAT NAME VALUE STAT NAME VALUE STAT NAME VALUE STAT NAME VALUE STAT NAME VALUE
REQ-37	5	The manager interface should be able to view and edit the menu.	 Menu MENU EDITOR EDIT CATEGORIES POPULAR ITEM NAME PRICE: $XX.XX BREAKFAST ITEM NAME PRICE: $XX.XX ITEM NAME PRICE: $XX.XX ITEM NAME PRICE: $XX.XX LUNCH ITEM NAME PRICE: $XX.XX

REQ-38	4	The manager interface should be able to view and edit ingredient count.	
REQ-39	4	The manager interface should be able to view and edit past and present orders.	

Stock Count

STOCK COUNT

ITEM	VALUE	UNIT
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS

EDIT VALUES

editOrder

EDIT ORDER

CHK #: XXX
TIME CREATED: XX:XX:XX
SUBTOTAL: $XX.XX
TOTAL: $XX.XX

SUBSTATS

STAT	VALUE
NAME	JOHN SMITH
TABLE	XXX
TIME ELAPSED	XX.XX.XX

ORDERED

ITEM	COUNT	PRICE
MINI PIE	2	$12.99
PIZZA SLICE	1	$9.99
BURGER	2	$13.99

Order Queue

ORDER QUEUE

CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX

CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX

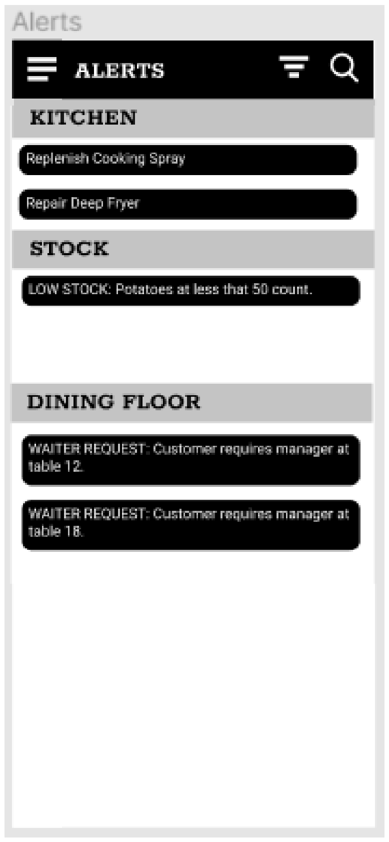
CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX

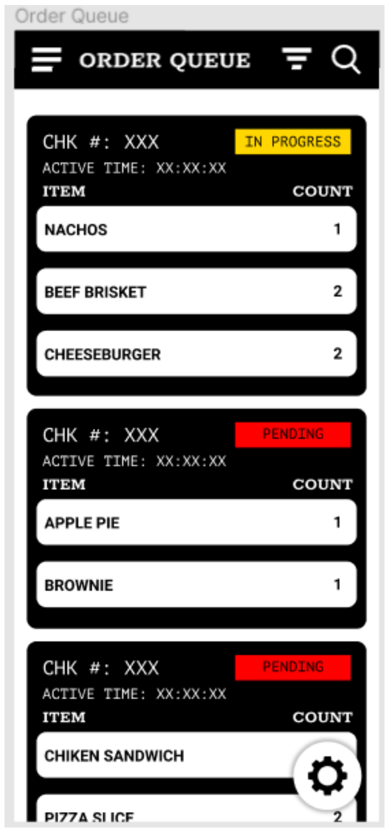
CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX

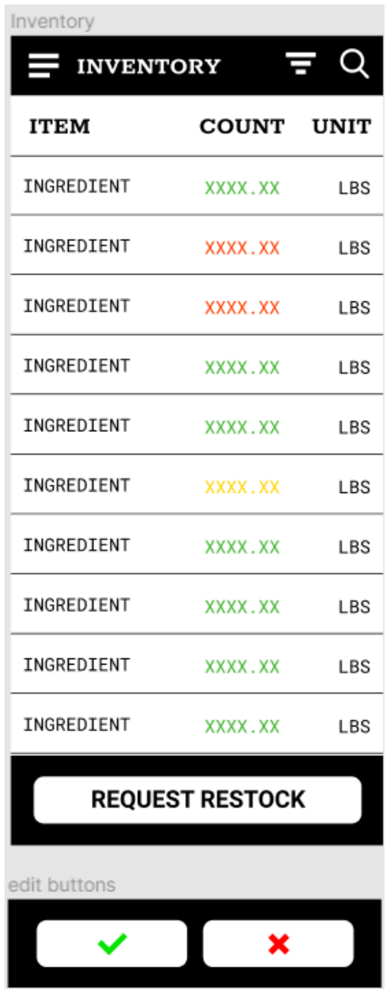
CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX

CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX

CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER N SUBTL: $XX TOTAL: $XX.

REQ-40	5	The manager interface should be alerted if any ingredient is below a certain threshold or insufficient to make a dish that uses it.	 Alerts ALERTS KITCHEN Replenish Cooking Spray Repair Deep Fryer STOCK LOW STOCK: Potatoes at less that 50 count. DINING FLOOR WAITER REQUEST: Customer requires manager at table 12. WAITER REQUEST: Customer requires manager at table 18.

REQ-41	4	The chef interface should be able to view and edit current orders.	 Order Queue ORDER QUEUE CHK #: XXX IN PROGRESS ACTIVE TIME: XX:XX:XX ITEM COUNT NACHOS 1 BEEF BRISKET 2 CHEESEBURGER 2 CHK #: XXX PENDING ACTIVE TIME: XX:XX:XX ITEM COUNT APPLE PIE 1 BROWNIE 1 CHK #: XXX PENDING ACTIVE TIME: XX:XX:XX ITEM COUNT CHIKEN SANDWICH PIZZA SLICE 2

REQ-42	4	The chef interface should be able to view and edit ingredient count.	

Inventory

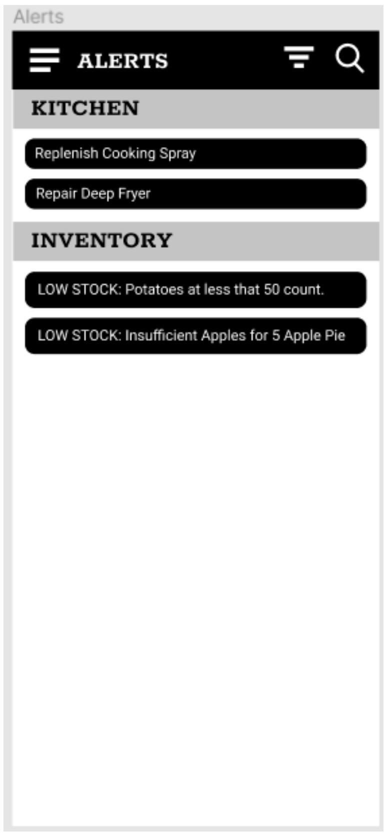
INVENTORY

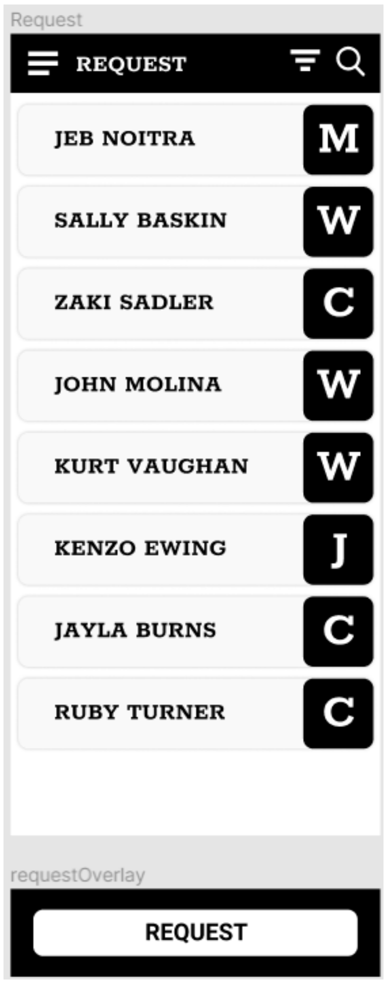
ITEM	COUNT	UNIT
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS
INGREDIENT	XXXX.XX	LBS

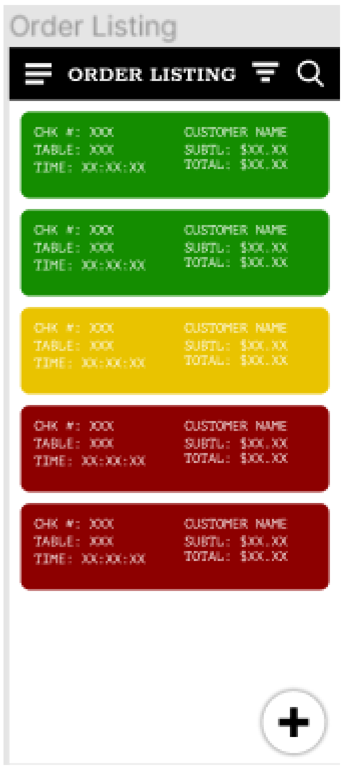
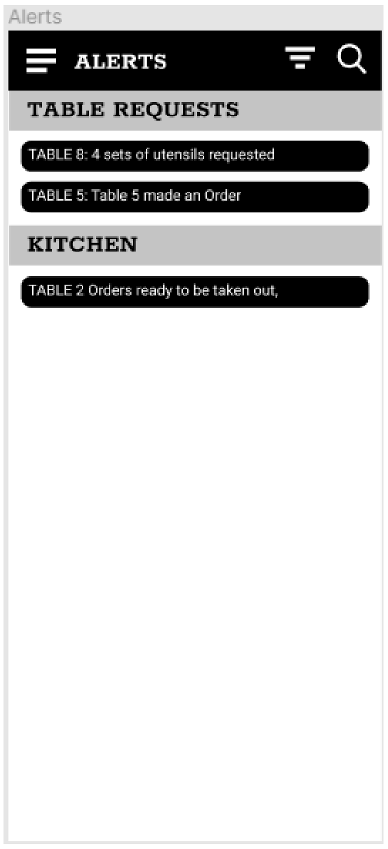
REQUEST RESTOCK

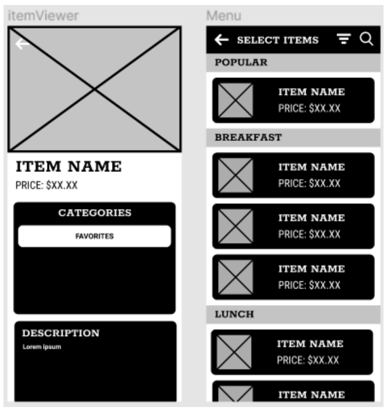
edit buttons

✓ ✗

REQ-43	4	The chef interface should be alerted if any ingredient is below a certain threshold or insufficient to make a dish that uses it.	 Replenish Cooking Spray Repair Deep Fryer LOW STOCK: Potatoes at less that 50 count. LOW STOCK: Insufficient Apples for 5 Apple Pie

REQ-44	4	The chef interface should be able to summon a specific waiter or manager.	 Request REQUEST JEB NOITRA M SALLY BASKIN W ZAKI SADLER C JOHN MOLINA W KURT VAUGHAN W KENZO EWING J JAYLA BURNS C RUBY TURNER C requestOverlay REQUEST

REQ-45	3	The waiter interface should display the orders made and the progress on each order.	 CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX CHK #: XXX TABLE: XXX TIME: XX:XX:XX CUSTOMER NAME SUBTL: $XX.XX TOTAL: $XX.XX
REQ-46	3	The waiter interface should display alerts from chefs or customers	 TABLE 8: 4 sets of utensils requested TABLE 5: Table 5 made an Order TABLE 2 Orders ready to be taken out,

REQ-47	3	The waiter interface should display the menu as the customer sees it	 itemViewer Menu SELECT ITEMS POPULAR ITEM NAME PRICE: $XX.XX BREAKFAST ITEM NAME PRICE: $XX.XX ITEM NAME PRICE: $XX.XX ITEM NAME PRICE: $XX.XX LUNCH ITEM NAME PRICE: $XX.XX ITEM NAME ITEM NAME PRICE: $XX.XX CATEGORIES FAVORITES DESCRIPTION Lorem ipsum

Section 3: Use Cases

3.1: Stakeholders

1. Restaurant Owner - Have an interest in improving worker efficiency so that better service will enable better word-of-mouth which in turn can cause an increase in revenue.
2. Managers - Keeping track of operations and maintaining quality will allow smoother operation between the wait staff and the kitchen.

3.2: Actors and Goals

Definitions: Item status (not prepared, being prepared, prepared)

i. Customers: Initiating, Order food, satisfactory dining experience, dish attribute comprehension (attributes include: description, allergens/ingredients)
ii. Waiters: Initiating, Deliver food, order food if the customer cannot, customer satisfaction, order tracking, keeps track of complete dishes (async).
iii. Chef: Initiating, Cook food, track ingredient stock, minimize waste, order tracking (only output no input)
iv. Managers: Initiating, Track ingredient stock, minimize waste, item trend tracking,
v. Application: Participating, Restaurant automation

3.3: Use Cases

i. Casual Description

Ordering

UC-1: A customer will be able to use the system to place an order through their phone in addition to directly via the waiter by requesting they come to their table. **REQ-2, REQ-3, REQ-4, REQ-6, REQ-9, REQ-33, REQ-47.**

UC-2: After an order is placed, they should be able to view progress of their order from order placed to delivered. This will correspond to **REQ-6, REQ-9, REQ-45,**

UC-3: The chef should be able to view and update the status of each item in the current order queue. This will correspond to **REQ-8, REQ-9, REQ-41.**

UC-4: The customer should be able to request a waiter to come to their table, as well as request extra utensils, or other miscellaneous requests through the app. This will correspond to **REQ-2, REQ-35, REQ-46.**

UC-5: The manager will be able to view and edit the current orders placed, as well as archived orders placed. This will correspond to **REQ-7, REQ-20, REQ-39**.

Smart Menu

UC-6: A customer will be able to see all available items, which will be kept up to date by the manager. The customer should also see recommended items based on the popularity of dishes. The waiter should also be able to view the menu. This will correspond to **REQ-1, REQ-10, REQ-11, REQ-5, REQ-12, REQ-30, REQ-31, REQ-32, REQ-34, REQ-37, REQ-47.**

UC-7: A waiter will be able to see the current status of any order placed and track it at any time. This will correspond to **REQ-4, REQ-9, REQ-45.**

UC-8: The manager will be able to see all available items and edit these items. The manager can hide items that should not be seen by customers or add new ones. This will correspond to **REQ-12, REQ-37.**

Ingredient Tracking

UC-9: The manager and chef will be able to view and edit the ingredient stock count and units it is measured in. This will correspond to **REQ-16, REQ-17, REQ-18, REQ-38, REQ-42.**

UC-10: The manager will be able to view statistics regarding the preparation of all meals including: ingredients used, most popular dishes, and most common time a meal is ordered. This will correspond to **REQ-13, REQ-20, REQ-36.**

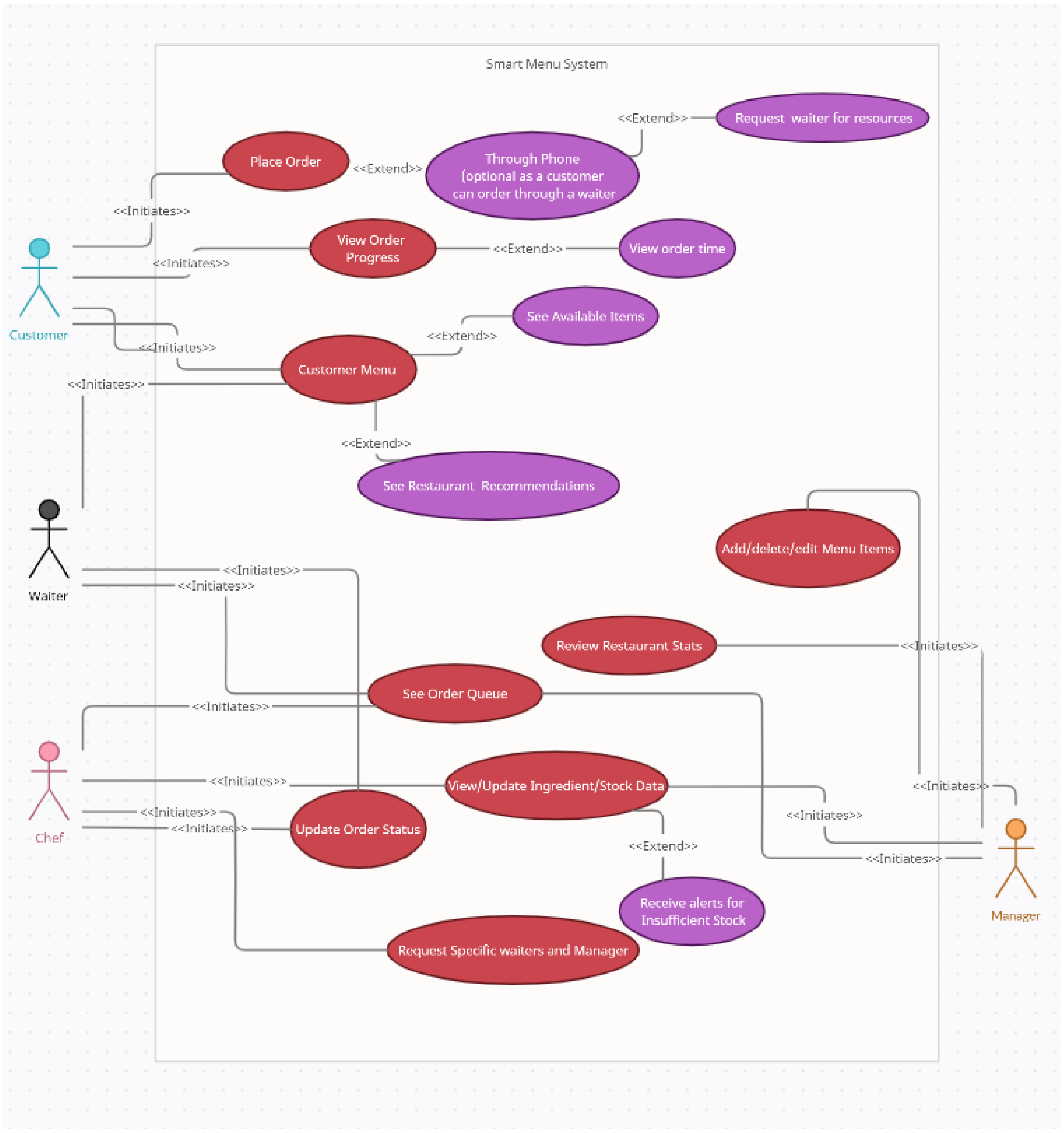
UC-11: The manager and chef should receive an alert when a dish cannot be made due to insufficient stock and when an ingredient is below a certain threshold. This will correspond to **REQ-14, REQ-15, REQ-18, REQ-40, REQ-43.**

UC-12: The availability of ingredients will update as orders are placed throughout the day.

Miscellaneous

UC-13: The chef should be able to summon a specific waiter or manager. This will correspond to **REQ-19, REQ-44.**

ii. Use Case Diagram


Smart Menu System
Customer
Waiter
Chef
Manager
Place Order
Through Phone (optional as a customer can order through a waiter
Request waiter for resources
View Order Progress
View order time
See Available Items
Customer Menu
See Restaurant Recommendations
Add/delete/edit Menu Items
Review Restaurant Stats
See Order Queue
View/Update Ingredient/Stock Data
Update Order Status
Receive alerts for Insufficient Stock
Request Specific waiters and Manager
<<Initiates>>
<<Extend>>

iii. Traceability Matrix

REQ = Requirement

UC = use case

	UC	1	2	3	4	5	6	7	8	9	10	11	12
REQ	Weight												
1	5						X						
2	4	X			X								
3	3	X											
4	2	X						X					
5	4						X						
6	3	X	X										
7	4					X							
8	3			X									
9	5	X	X	X				X					
10	3						X						
11	1						X						
12	5						X		X				
13	5										X		
14	4											X	
15	4											X	
16	3									X			
17	5									X			
18	4									X		X	
19	4												X
20	1					X					X		
30	5						X						

31	2						X						
32	3						X						
33	5	X											
34	4						X						
35	3				X								
36	5										X		
37	5						X		X				
38	4									X			
39	4					X							
40	5											X	
41	4			X									
42	4									X			
43	4											X	
44	4												X
45	3		X					X					
46	3				X								
47	3	X					X						

iv. Fully-Dressed Description

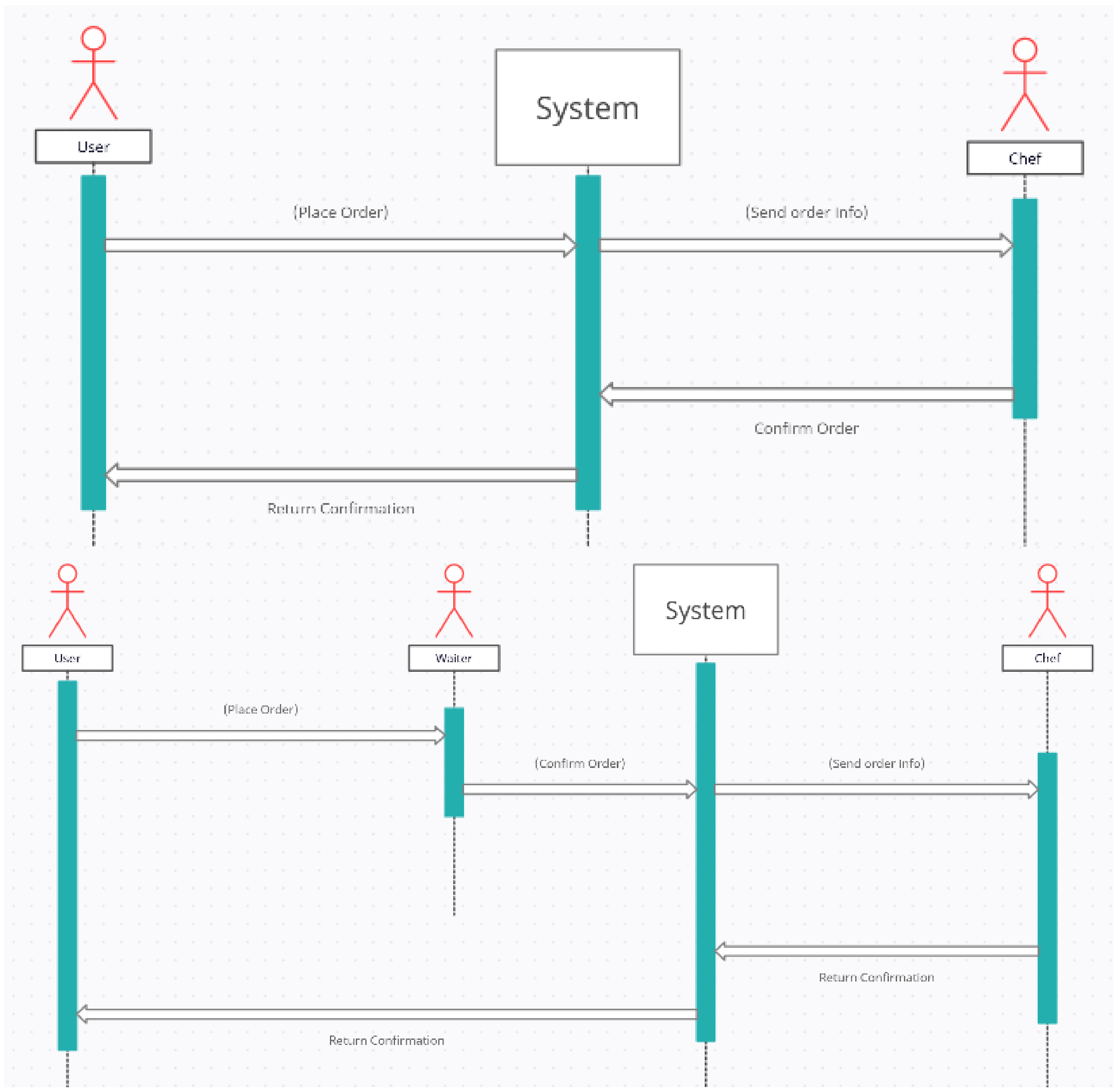
Use Case UC-1: Place Order
Related Requirements: **REQ-2, REQ-3, REQ-4, REQ-6, REQ-9, REQ-33, REQ-47** Initiating Actor: Any of: Customer, Waiter Actor's Goal: To place an order to be sent to the kitchen to be prepared for the customer, and to view progress of their order. Participating Actors: None Preconditions: 1. There is at least one item in the order Minimal Guarantees: The order will not get sent to Order Queue if systems fails Success Guarantees: Order will be sent to Order Queue Flow of Events for Main Success Scenario: 1. Customer is seated at a table, and the customer selects the items they would like to order. 2. Customer selects menu item "Place Order" 3. Customer receives order confirmation message and button to go to order progress menu 4. System goes to the Order Progress menu if Order placement was successful.

Use Case UC-9: Edit Kitchen Stock
Related Requirements: **REQ-16, REQ-17, REQ-18, REQ-38, REQ-42.** Initiating Actor: Chef and Manager Actor's Goal: Update Kitchen Stock so system displays accurate figures of supplies Participating Actors: None Preconditions: 1. There is at least one ingredient added to stock list 2. User must change at least one value or menu won't update. Minimal Guarantees: Kitchen stock information will not be updated if no changes were made. Success Guarantees: Kitchen stock get updated according to users additions Flow of Events for Main Success Scenario: 1. User selects the Kitchen stock menu. 2. User then enters the respective values depending on physical ingredients or can add/remove ingredients. 3. User selects "Update" when finished.

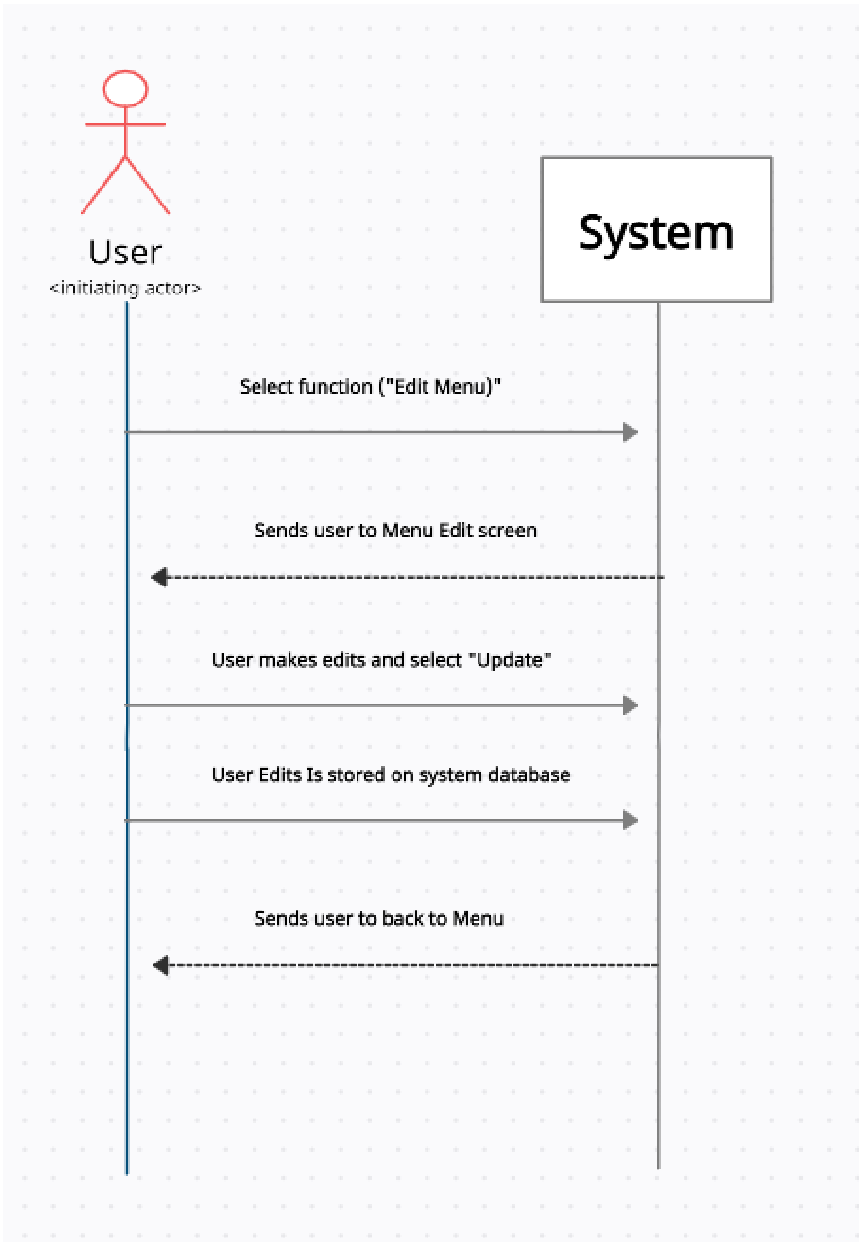
Use Case UC-8: Edit Menu
Related Requirements: **REQ-12, REQ-37.** Initiating Actor: Manager Actor's Goal: To add/remove or edit item(s) in menu Participating Actors: None Preconditions: 1. The must be at least one menu item before user can apply their menu changes Minimal Guarantees: Menu changes will be applied if the result leave menu blank Success Guarantees: Menu gets update according to user changes Flow of Events for Main Success Scenario: 1. User selects the Smart Menu option. 2. User selects the "Edit" option. 3. User can then adjust the item name, category, price, ingredient cost, picture, description, and add or remove items from the menu. 4. User selects "Update" when finished.

3.4: System Sequence Diagrams

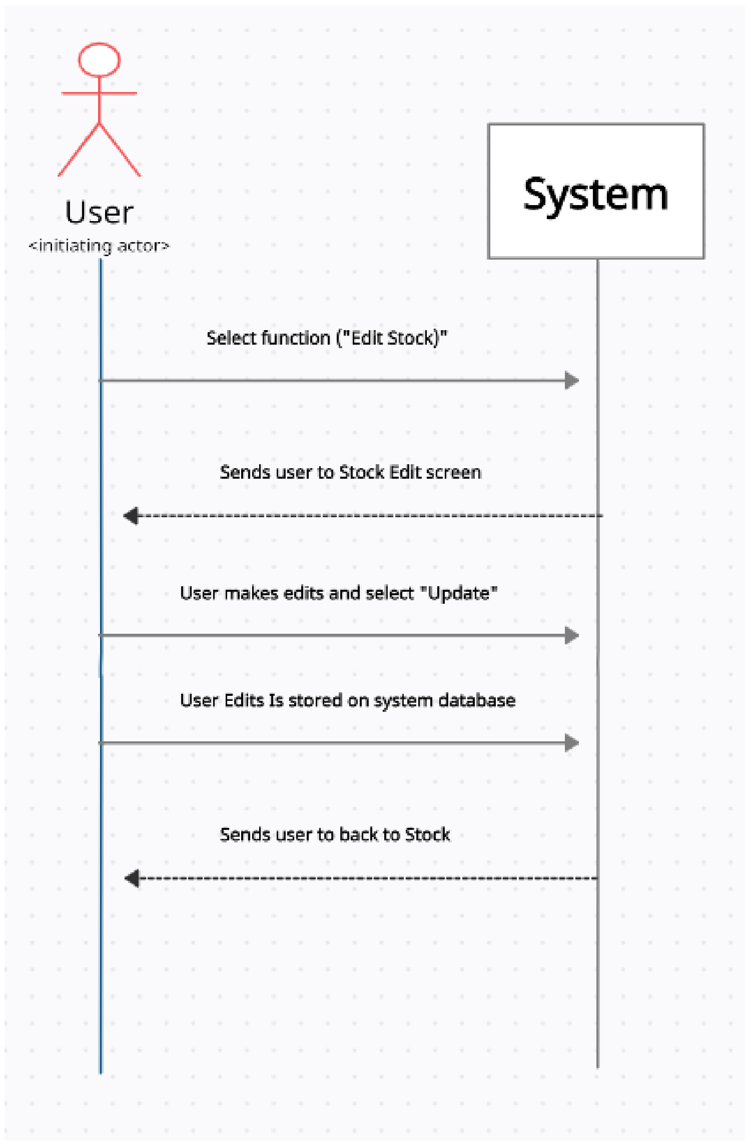
UC-1: Place Order



UC-8: Edit menu


User
<initiating actor>
System
Select function ("Edit Menu)"
Sends user to Menu Edit screen
User makes edits and select "Update"
User Edits Is stored on system database
Sends user to back to Menu

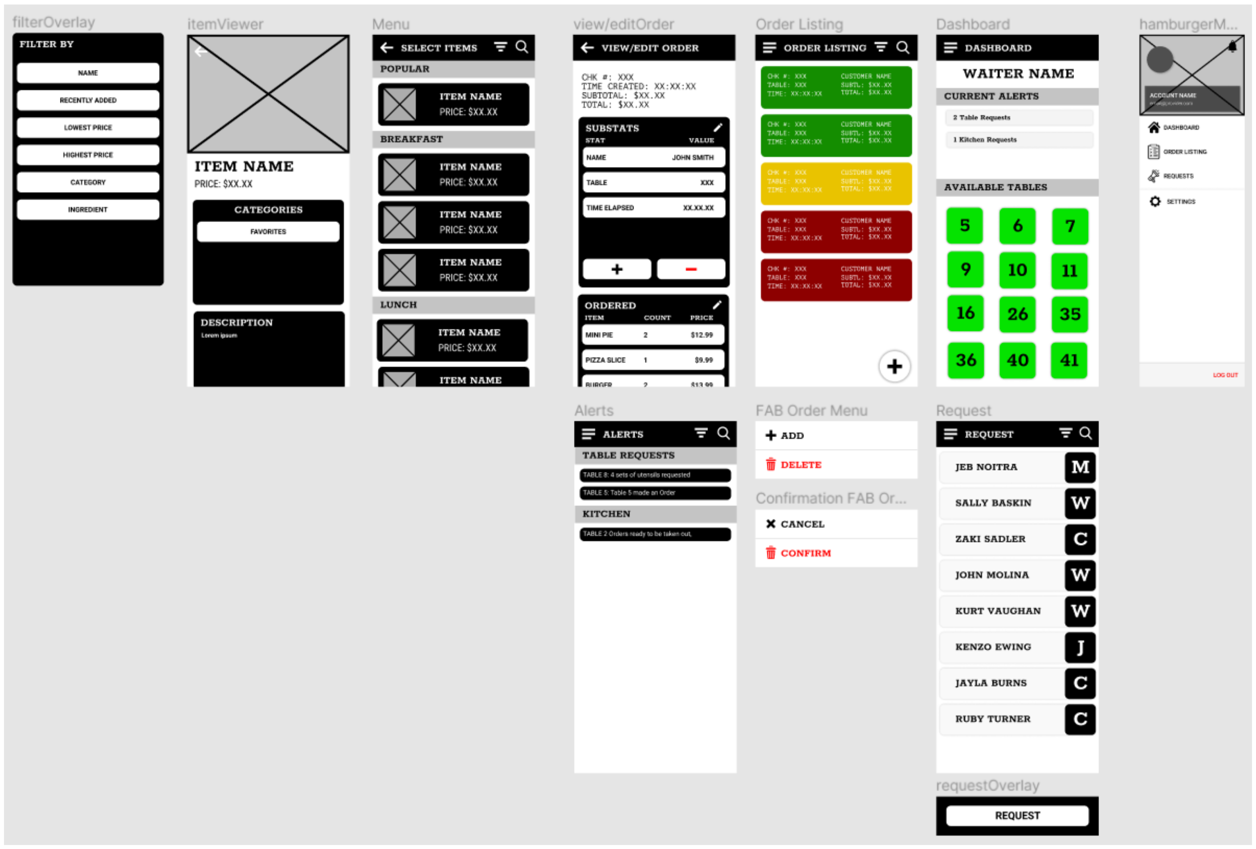
UC-9: Edit Ingredient Stock


User
<initiating actor>
System
Select function ("Edit Stock)"
Sends user to Stock Edit screen
User makes edits and select "Update"
User Edits Is stored on system database
Sends user to back to Stock

Section 4: User Interface Specification

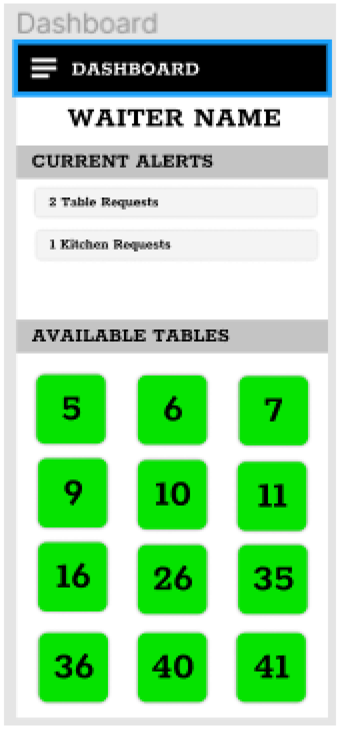
4.1: Preliminary Design

Waiter Main Interface

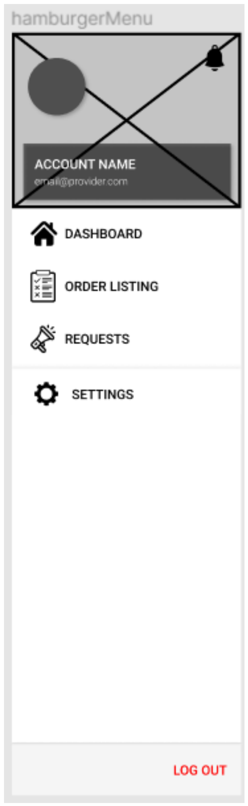


For Use Case 7 (Waiter can view current orders):

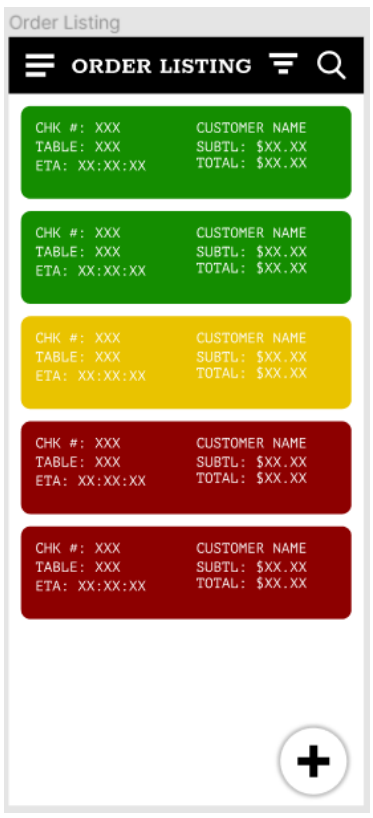
1. Start at home screen (dashboard with table statuses)



2. Upon pressing on the hamburger button and selecting the tab Order Listing:



3. It brings the waiter to the page of the current orders along with the status (represented by the color of the order card) and wait time (ETA) associated. Select an order card to open up the View/Edit Order page. This page lists the menu items and the price of each item in the order.



Order Listing

ORDER LISTING

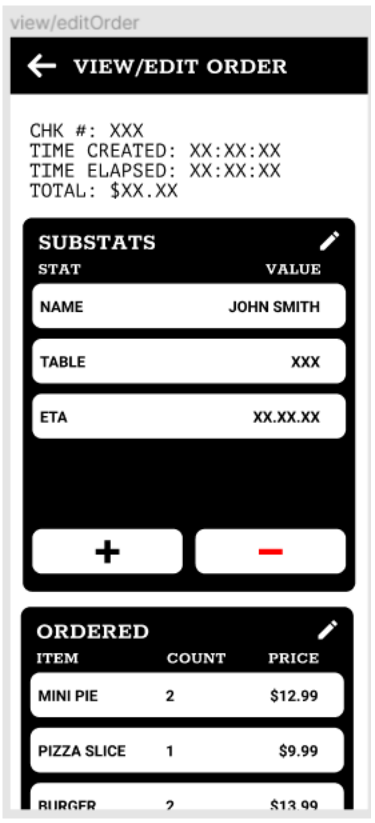
CHK #: XXX	CUSTOMER NAME
TABLE: XXX	SUBTL: $XX.XX
ETA: XX:XX:XX	TOTAL: $XX.XX

CHK #: XXX	CUSTOMER NAME
TABLE: XXX	SUBTL: $XX.XX
ETA: XX:XX:XX	TOTAL: $XX.XX

CHK #: XXX	CUSTOMER NAME
TABLE: XXX	SUBTL: $XX.XX
ETA: XX:XX:XX	TOTAL: $XX.XX

CHK #: XXX	CUSTOMER NAME
TABLE: XXX	SUBTL: $XX.XX
ETA: XX:XX:XX	TOTAL: $XX.XX

CHK #: XXX	CUSTOMER NAME
TABLE: XXX	SUBTL: $XX.XX
ETA: XX:XX:XX	TOTAL: $XX.XX



view/editOrder

VIEW/EDIT ORDER

CHK #: XXX
TIME CREATED: XX:XX:XX
TIME ELAPSED: XX:XX:XX
TOTAL: $XX.XX

SUBSTATS

STAT	VALUE
NAME	JOHN SMITH
TABLE	XXX
ETA	XX.XX.XX

ORDERED

ITEM	COUNT	PRICE
MINI PIE	2	$12.99
PIZZA SLICE	1	$9.99
BURGER	2	$13.99

4.2 User Effort Estimation

Use Case	Navigation	Clerical Data Entry
UC-1	Requires REQ-2, REQ-3, REQ-4, REQ-6, REQ-9, REQ-33, REQ-47. 1. Menu displays upon login 2. Select items to order 3. Press order overlay button. 4. Confirm order at redirected checkout. Total clicks/presses: 3	1. Enter First Name 2. Enter Last Name 3. Enter Card Number 4. Enter Card Exp. Date 5. Enter Card CVV 6. Select "Pay Now"
UC-2	Requires REQ-6, REQ-9, REQ-45, 1. Order status displays upon order submission (assuming customer has already submitted order, the order status will show up on the home screen with no need for further navigation)	
UC-3	Requires REQ-8, REQ-9, REQ-41 1. Select Order Queue tab in bottom menu 2. To change status, select the item and keep tapping it to toggle between the statuses "not prepared", "preparing", and "prepared." Total number of clicks/presses: 2	
UC-4	Requires REQ-2, REQ-35, REQ-46 1. If order can't be placed, the waiter can be requested by pressing the hamburger icon. 2. Selecting the dashboard tab. 3. Pressing the request waiter button.	
UC-5	Requires REQ-7, REQ-20, REQ-39:	

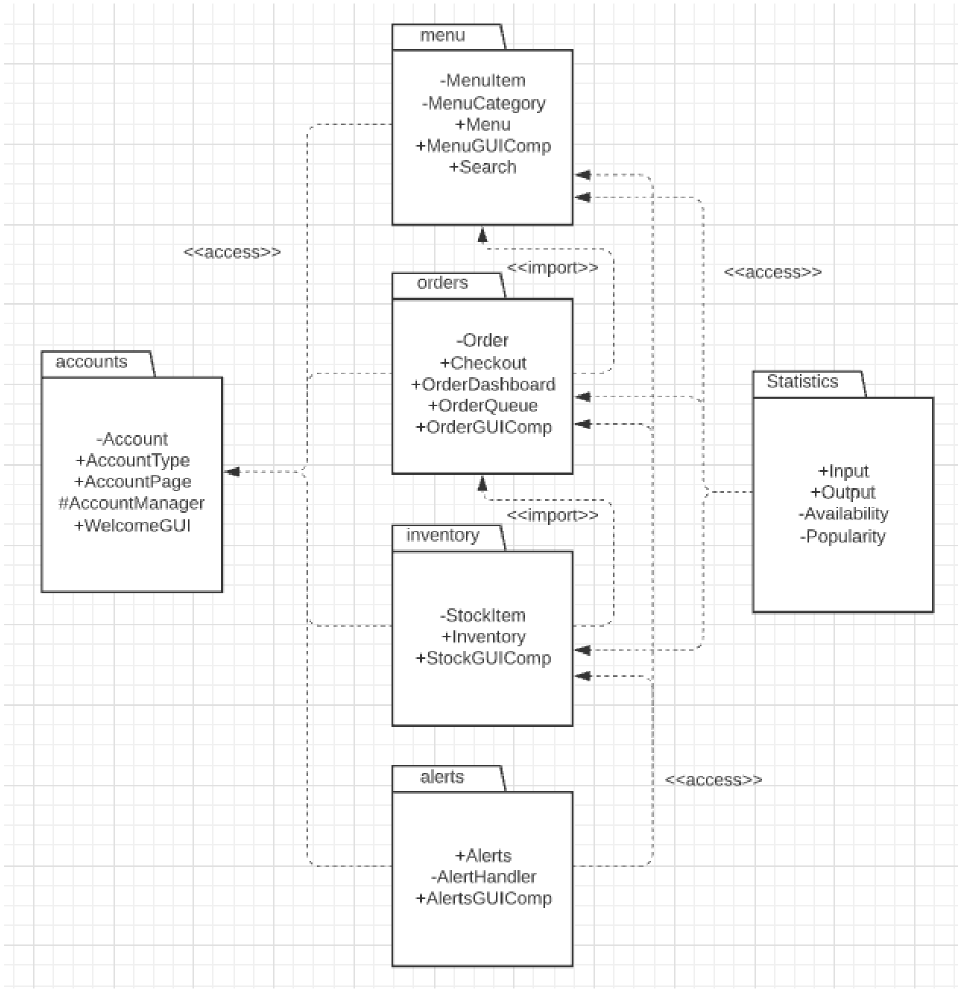
	To view current and past orders: 1. Press the hamburger icon 2. Select “ORDER QUEUE” 3. Press order to view and edit. 4. Edit order cards by pressing the pencil icon. 5. Save changes by pressing the “SAVE CHANGES” button below, or cancel by pressing the “CANCEL” button below or back arrow. Total number of clicks/presses: 5	
UC-6	Requires REQ-1, REQ-10, REQ-11, REQ-5, REQ-12, REQ-30, REQ-31, REQ-32, REQ-34, REQ-37, REQ-47. 1. Menu displays on login 2. Popularity already shown 3. To filter, press the filter button in the header. 4. Press + to add filters to filter modal 5. Tap out of modal to see changes. Total clicks/presses: 3	
UC-7	Requires REQ-4, REQ-9, REQ-45. 1. When logged in, it shows the number of items in an order for a table and the time needed to finish 2. For in-depth information, select the “Orders” tab from the hamburger menu denoted by a checklist Total clicks/presses: 3	
UC-8	Requires REQ-12 and REQ-37 1. Select the hamburger menu icon on top left. 2. Select tab named “MENU EDITOR” 3. A digital menu is shown, and you can edit single menu items in the editor by selecting said item. Can edit item categories by selecting the above CATEGORIES button, which	1. Enter menu item name. 2. Enter ingredients. 3. Enter category (optional) 4. Enter description Total entries: 4

	navigates to CATEGORIES. 4. In the item editor, edit item name by selecting the pencil icon. Edit any UI card by selecting the pencil icon. 5. Pressing a card's plus and minus button group can add/remove data entries from a card. 6. Save changes by pressing the "SAVE CHANGES" button below, or cancel by pressing the "CANCEL" button below or back arrow. Total clicks/presses: 6	
UC-9	Requires REQ-15, Req-19, Req-43: To view ingredient stock count, the chef has to: 1. Select the hamburger icon 2. Select the inventory menu option 3. Scroll to the needed ingredient 4. Press the button under the menu's settings column that says "Adjust" Total number of clicks/presses: 3	
UC-10	Requires REQ-13, REQ-20, REQ-36: To view statistics, the manager from the home page can: 1. Press the hamburger icon. 2. Select "STATISTICS" 3. View an interactable graph Total clicks/presses: 2	
UC-11	Requires this will correspond to REQ-14, REQ-15, REQ-18, REQ-40, REQ-43. . 1. Press the hamburger icon on the top left. 2. Select the bell icon on the top right. 3. The manager can now see current notifications/alerts. Total number of clicks/presses: 2	

UC-12	Requires REQ-19, REQ-44 1. Open application 2. Select Hamburger Menu 3. Select the Alerts menu 4. Press on the “Request” sub-menu 5. Press on the “Request” button corresponding to the staff member to request Total number of clicks/presses: 4	

Section 5: System Architecture

5.1: Identifying Subsystems



There are 6 subsystems:

Menu: Stores and displays the menu. Includes the tools to edit the menu for admins. Contains a database of all menu items.

Accounts: Handles the authentication of users. Manages the permissions of a user. Includes tools to manage accounts for admins.

Alerts: Stores, displays and dismisses alerts for users.

Orders: Stores and displays orders. Has a whitelist of users capable of seeing the order including the customer who made the order (Admins allowed regardless). Includes tools to edit orders status for staff and includes tools to manage orders for admins.

Inventory: Stores and displays the current stock of items. Includes tools to edit stock for chefs and managers.

Statistics: Stores and displays data about item popularity, trends, etc.

How do they tie into each other:

The **Menu** component is used to select items to add to an order for the **Orders** component. **Menu** also sends item availability data and item category data to **Statistics**.

The **Accounts** component will allow users to log in, which will allow them more permissions. With the right permissions, the user will be able to (**Menu**) edit the menu, manage (**Accounts**) user permissions, (**Orders**) see and/or edit orders, (**Inventory**) see the current stock of items, (**Alerts**) see alerts sent to that user and (**Statistics**) see the statistics gathered.

The **Orders** component sends item order statistics and data about item add/remove to/from the order list to **Statistics**. It also sends an alert via **Alerts** to a waiter when a customer asks for assistance or a chef marks an order as completed.

The **Inventory** component will send an alert via **Alerts** when an ingredient is low and sends data to **Statistics** about ingredient usage.

The **Statistics** component allows **Menu** to know which items are popular.

5.2: Architecture Styles

Microservices are modularized subsystems that can communicate with each other. The microservices will send useful information to the subsequent microservices using standard communication protocols like HTTP. Each microservice is typically responsible for a specific task or subproblem and is capable of running autonomously from the rest of the system, sending their data to a shared data server or network. The autonomous nature of each of the microservices allows for them to be modified without affecting the rest of the system. An error occurring in one microservice will not hinder the operation of the following microservices. This allows for each microservice to be tested in isolation while maintaining the functionality of the system as a whole.

The application will implement three main microservices for its three primary business functionalities: smart menu, ordering, and ingredient tracking. The smart menu microservice will handle updates to the menu, including adding/removing items on the menu, updating items on the menu (including updating the amount of ingredients required for each item), and updating the availability of each item based on the ingredients available to the restaurant. This microservice will communicate with ingredient tracker microservice, which will handle changes to the ingredient quantity database. This microservice will handle both updates to ingredient quantity inputted by the manager as well as those resulting from food orders (which result in subtractions to the quantity of an ingredient available). Thus, the ingredient tracker microservice will have to communicate with the ordering microservice. The ordering microservice will handle customer orders and payment.

Communication between microservices will be implemented using synchronous messaging (e.g. client waits for response from service) via the HTTP protocol. Furthermore, each microservice (smart menu, ingredient tracker, ordering) will have its own private database to capture their data and implement its respective business functionality.

5.3: Mapping Subsystems to Hardware

Our application will have a sub-application that runs on the customers' devices (mobile phones, PCs, tablets, laptops, etc.). For managers, waiters and chefs, the application will be cross platform. This section contains our Customer interface, Chef interface, Waiter interface and Manager interface. These interfaces will involve React Native (multiplatform website framework) and use REST API (application communication format). Our application will also have a sub-application that runs on a server device. The server can be an onsite machine provided by the user, or it can be on the cloud. This will include our application logic and our database, which will involve Node.js (essential software for JavaScript), Express (web application framework), Sequelize (intermediary for JavaScript and database), REST API (application communication format), mySQL (database management system), Crypto-Js (security library), and jstat (statistics library).

5.4: Connectors and Network Protocols

In order to create our restaurant automation application, our team is implementing a Express.js API. An Express.js API is the web application framework we will be using for Node.js. It will provide low latency for our servers by removing waiting times for return requests. Node.js will be useful for our application because it operates a single-threaded event loop registered with the system to handle connections, it is very efficient and reduces CPU and memory load on our system. The front-end of the application will be built with React Native, which will handle performing the requests. The user interacts with the application and when want to complete a goal, a request will be sent to a server via HTTP where the information will be processed. HTTP means hypertext transfer protocol, which is how the user's device communicates data through the web. For example, if the user orders food after clicking confirm, the order will be sent via HTTP request to the server which will respond back with status of the order, which is necessary for the client to display to the user. Another Example is when the chef or manager wishes to view the order queue, the request gets sent to the server then responds with the current order queue.

5.5: Global Control Flow

i. Execution orderliness

The application is structured such that there are linear segments that can be initiated in an event-driven manner. For example, for a customer, ordering food is relatively linear, although there will be the option to leave the ordering segment entirely. Logging in is a linear process that can be initiated by an event.

ii. Time dependency

The system is event-driven. All events draw on the real-time clock for logging purposes. Also, all events that have a timeout or a timer measure it in real time. Timeouts are to be user-defined.

5.6: Hardware Requirements

With the system being software-only, there is no strict need for specific hardware. The software will have to be able to run on mobile devices that connect to the internet in some form, with a minimum required 1 MBps connection speed in order to function. It needs to be able to work on devices as old as the iPhone 6 or the Samsung Galaxy S7, meaning there will be support for the current version of iOS and Android. Devices will need a functioning touchscreen to provide the needed input, as well as a minimum screen resolution of 750 x 1334 pixels. In order to be installed, there must be at least 40 MB of device storage remaining. The device should have a minimum clock speed of 1 GHz. For the server, it requires at minimum 2.4 GHz clock speed as well as memory space capacity of 0.5 GiB.

Section 6: Project Size Estimation

Based on the use cases detailed in Section 3.3, the project does not appear to be too big to work on. There are not too many use cases involved

Unadjusted Use Case Weight (UUCW) – the point size of the software that accounts for the number and complexity of use cases.

Unadjusted Actor Weight (UAW) – the point size of the software that accounts for the number and complexity of actors.

Technical Complexity Factor (TCF) – factor that is used to adjust the size based on technical considerations.

Environmental Complexity Factor (ECF) – factor that is used to adjust the size based on environmental considerations.

Use Case Class	Use Cases Involved	No. of Transactions	No. Cases	Weight
Simple	UC-8	1-2 transactions/req	1	5
Average	UC-2,3,4,5,6,7,9,10,12	3-6 transactions/req	9	10
Complex	UC-1,6	7 or more transactions/req	2	15

UUCW = (1 x 5) + (9 x 10) + (2 x 15) = 125

UAW = 4 simple (4 microservices) * 1 + 4 (3 db + 1 HTTP) * 2 + 4 complex (customer, manager, waiter, chef) * 3 = 24

TCF

Factor	Description	Weight	Assigned value	Weighted value
T1	Distributed system	2.0	2	4
T2	Response time/performance objectives	1.0	5	5
T3	End-user efficiency	1.0	4	4
T4	Internal processing complexity	1.0	2	2
T5	Code reusability	1.0	4	4

T6	Easy to install	0.5	4	2
T7	Easy to use	0.5	5	2.5
T8	Portability to other platforms	2.0	5	10
T9	System maintenance	1.0	2	2
T10	Concurrent/parallel processing	1.0	4	4
T11	Security features	1.0	3	3
T12	Access for third parties	1.0	1	1
T13	End user training	1.0	3	3

Sum: 46.5

TCF = 46.5/100 + 0.6 = 1.065

ECF = 1

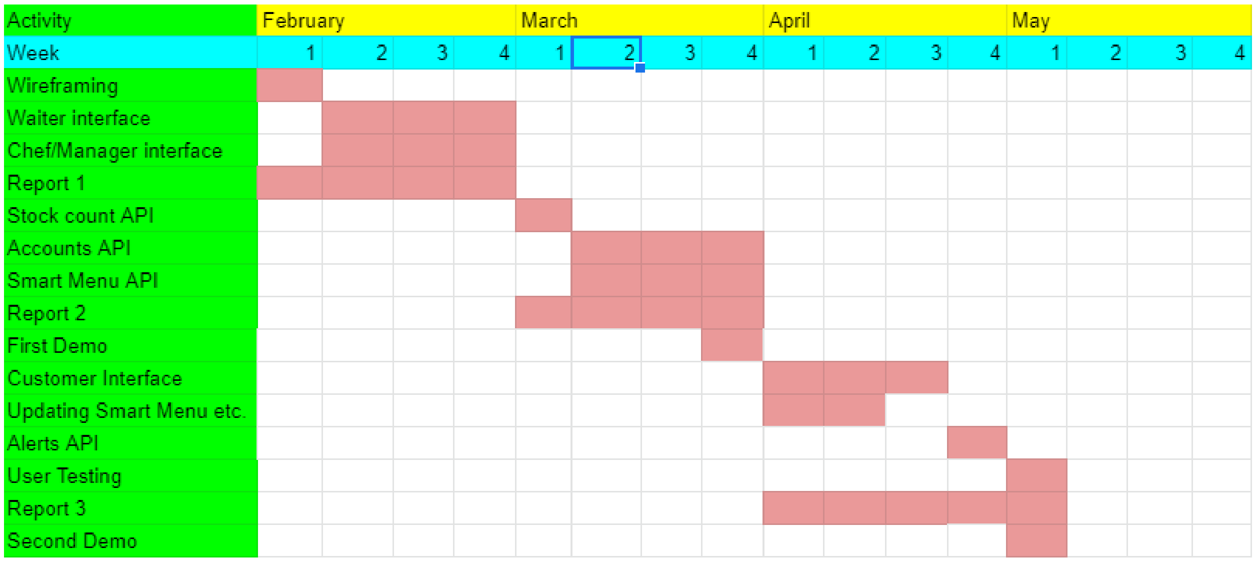
UCP = (UUCW + UAW)*TCF*ECF = 158.6

Estimated Effort = UCP * 1.5 hours per use case point = 237 hours

Section 7: Project Management

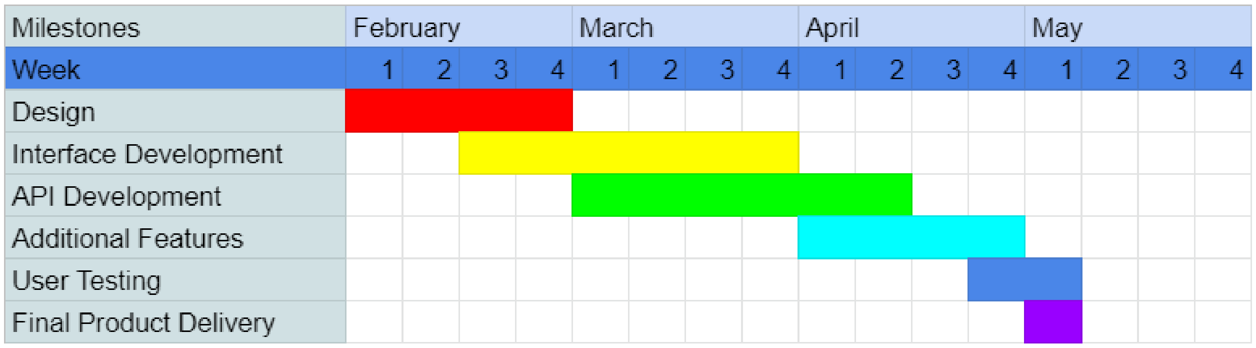
7.1: Plan of Work

Gantt Chart



Activity	February				March				April				May			
Week	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Wireframing	🟥															
Waiter interface		🟥	🟥	🟥												
Chef/Manager interface		🟥	🟥	🟥												
Report 1	🟥	🟥	🟥	🟥												
Stock count API					🟥											
Accounts API						🟥	🟥	🟥								
Smart Menu API						🟥	🟥	🟥								
Report 2					🟥	🟥	🟥	🟥								
First Demo								🟥								
Customer Interface									🟥	🟥	🟥					
Updating Smart Menu etc.									🟥	🟥						
Alerts API												🟥				
User Testing													🟥			
Report 3									🟥	🟥	🟥	🟥	🟥			
Second Demo													🟥			

Project Roadmap



Milestones	February				March				April				May			
Week	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Design	🟥	🟥	🟥	🟥												
Interface Development			🟨	🟨	🟨	🟨	🟨	🟨								
API Development					🟩	🟩	🟩	🟩	🟩	🟩						
Additional Features									🟦	🟦	🟦	🟦				
User Testing												🟦	🟦			
Final Product Delivery													🟪			

Product Ownership

Every individual is assigned 2 functional requirements to tackle.

Names/REQ	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Raghav											🟦	🟦								
Alan										🟦			🟦							
Esteban			🟦																🟦	
Suryansh	🟦														🟦					
Shrey							🟦	🟦												
Kunj									🟦								🟦			
Adrian					🟦									🟦						
Ray		🟦														🟦				
Akira				🟦														🟦		
Parth						🟦														🟦

7.2: Responsibilities

Section Description: what each team member did so far, is currently doing, will do in the future, including management and coordination activities.

NOTE: requirement assignments/responsibilities subject to change in the future

Raghav: Currently schedules weekly meetings, will implement REQ-11 and REQ-12

Alan: Currently divides labor for each requirement, will implement REQ-10 and REQ-13

Esteban: Currently manages business goals, will implement REQ-3 and REQ-19

Suryansh: Currently manages UI requirements, will implement REQ-1 and REQ-15

Shrey: Currently manages team, will implement REQ-7 and REQ-8

Kunj: Currently manages non functional requirements, will implement REQ-9 and REQ-17

Adrian: Currently manages document formatting, will implement REQ-5 and REQ-14

Ray: Currently manages team communication server, will implement REQ-2 and REQ-16

Akira: Currently manages functional requirements, will implement REQ-4 and REQ-18

Parth: Currently manages references, will implement REQ-6 and REQ-20

References

Full Service Restaurant News. (2013, October 8). *Study Released on Average Restaurant Wait Times*. fsrmagazine.com. Retrieved February 9, 2021, from https://www.fsrmagazine.com/content/study-released-average-restaurant-wait-times#:~:text=Restaurants%20certainly%20have%20plenty%20of,wait%20longer%20than%2040%20minutes.

Green Restaurant Association. (n.d.). *Waste Reduction and Recycling*. dinegreen.com. https://www.dinegreen.com/waste

Apple Inc. (2017, October 30). Ios device compatibility reference. Retrieved February 24, 2021, from https://developer.apple.com/library/archive/documentation/DeviceInformation/Reference/iOSDeviceCompatibility/Displays/Displays.html