Restaurant Management System Requirements Documentation

Lance Keith

INFO-C451-26578

Dr. Shawn Dai

01 May 2023

Contents

Problem Statement

Restaurants are the sum of many different parts all working together both synchronously and asynchronously. The teams that make the restaurant function can be broken down into bar staff, kitchen staff, wait staff, and hospitality staff. It is crucial that much of the communication between these teams be automated and seamless. Each team needs access to different kinds of information at different times, and it needs to be as current as possible. There are a limited number of seats, a limited number of staff, and a limit to the amount of food that can be prepared in a day. Each member of the different teams has a different share of the tips collected in the restaurant. Menu items are priced based on the cost of their ingredients, and are limited by those ingredients as to their availability. A restaurant management system is designed to make sure that everything has proper pricing, availability is tracked, there is adequate staff to cover each area of the floor, and that each customer is charged properly, and employees paid correctly.

Glossary of Terms

Wait staff – employees that take orders and deliver food to tables

Kitchen staff – employees that prepare the food

Hospitality staff – staff that manages customer experience and reservations

Bar staff - staff that prepares drinks and serves as wait staff for patrons seated at the bar

Managers – the head of each team responsible for overseeing their operation

POS – point of sale. Where orders are placed and customers charged

Waitlist – the digital queue of patrons waiting for an available table

Menu – all available items that can be ordered from the restaurant at a given time

Menu-item – items that are on a given menu, consist of ingredients

Ingredients – the items that have a managed inventory and determine whether kitchen staff can prepare a menu-item

Tip-share – the percentage of the total tips that an employee is entitled to during their shift

Floor map – a visual representation of the tables and seats in the restaurant which includes availability and active wait times

Cook-queue – the list of menu items that need to be prepared by kitchen staff in the order they were called for

Wait time – a time value that represents the amount of time a customer will have to wait to be seated

Cook time – a time value that represents the amount of time it takes to prepare a dish from the time it is ordered

86 – this is a term used by the restaurant industry to describe menu items that are no longer available to be ordered for the day

System Requirements

Functional Requirements

Number	Weight	Description
Req-1	High	The floor map should contain a visual representation of every table and the number of seats at that table.
Req-2	High	A member of hospitality staff should be able to determine wait times based on information on the waitlist
Req-3	High	Wait staff and kitchen staff should be able to determine whether a menu item is 86'd or not (with kitchen staff having the ability to 86 menu-items as needed)
Req-4	High	The system should have inventory information for each ingredient that is used by the restaurant, which updates when shipments arrive or dishes/drinks are prepared
Req-5	Medium	The system should be able to calculate an employee's tip share on a given shift and provide that information live as it changes.
Req-6	Low	The system should allow for multiple ways of sorting inventory information, with the default being alphabetically
Req-7	High	The system should not accept more reservations than are physically possible for a given time frame
Req-8	Medium	Items should be automatically 86 based on ingredient inventory
Req-9	Medium	All aspects of the system should be accessible to team members, with editing capability based on user permissions
Req-10	Low	The system should have capability for online ordering
Req -11	High	The system should have a superuser permission which can grant permissions and change them for each team member

Non-functional Requirements

Number	Weight	Description
Req-1	High	The system should be intuitive and easy to learn/teach
Req-2	High	The system should be secure. Information should never be accessible to anyone other than permitted users.
Req-3	Medium	The system should be visually appealing
Req-4	High	The system should have the ability to backup all inventory information in the event of a system corruption or failure
Req-5	Medium	The system should be able to run on most devices that have been released in the last 10 years
Req-6	Medium	The system should be scalable to allow for growth and expansion
Req-7	Medium	The system should be easily troubleshooted without the need for extensive knowledge of IT
Req-8	High	The system should be bug free
Req-9	Low	The program should not consume large amounts of resources to preserve mobile device battery life

Plan of Work

This week I spoke to a few friends who work in restaurants to see what kind of features bother them as well as which features are good in a restaurant management system. This coming week I will finalize my development environment and begin building the database. It may be the most complex aspect of the project, so I would like to start with that to make things easier going forward. I plan to spend two weeks on the database, and then another two programming the back

end features. Front end will be the last step and will take three weeks, which should leave plenty of time for polishing and debugging.

Stakeholders

- Restaurant owners
- Managers
- Employees
- Customers
- Suppliers
- The developer (me)

Actors and Goals

Primary Actors

- Customers: customers can place online orders by interacting with the menu and the modified self-service POS
- Employees: employees will utilize every aspect of the system based on their permissions
    - Change inventory
    - Add or remove customer from waitlist
    - Add or remove item from cook queue
    - Use POS features

Secondary Actors

- Admins: can add and delete users as well as assign permissions. Can create and change floor map, and add/remove menu items and ingredients

- System: calculates tip shares, updates inventory based on orders, removes items from menu based on availability, notifies staff when waitlist is at capacity

Use Cases

Customer (total: 6)

- Add/delete items from order (2)
- Pay for order (2)
- Request account (1)
- Login/logout (1)

Employee (total: 19)

- Add/delete items from order (2)
- Ring order: to assist customer with paying (2)
- Login/logout (1)
- Add/remove customer from waitlist (2)
- Add/remove items to the cook queue: when customers order, waitstaff adds their items to the queue (2)
- Update inventory: a backup if the system does not properly represent inventory (4)
- Change a table's availability on the floor map (6)

Admin (total: 16)

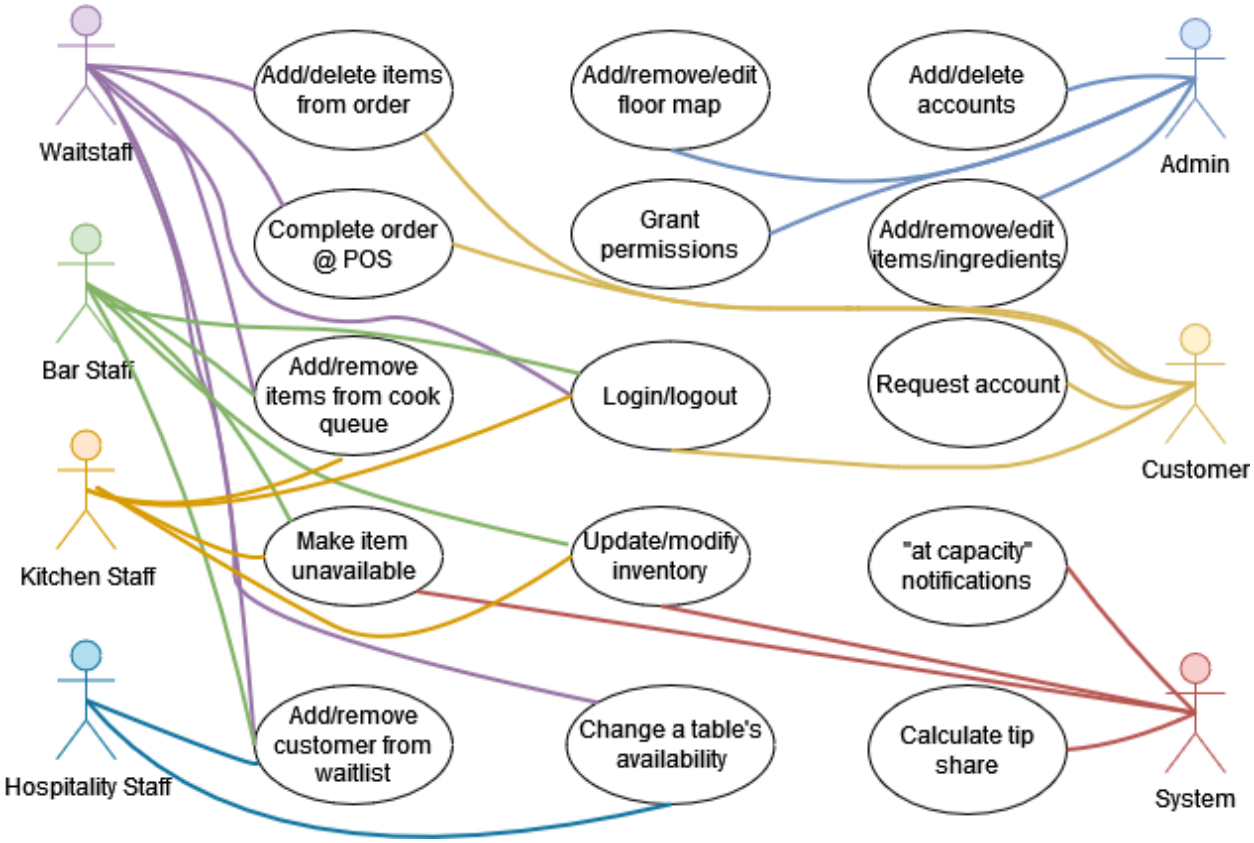
- Add/delete accounts (2)
- Grant permissions: specifies a user's account functionality (4)

- Add/edit floor map: can manipulate an existing floor map or create a new one (6)
- Add/edit/remove items and ingredients: can change what items are available for menus and which ingredients are in stock (4)

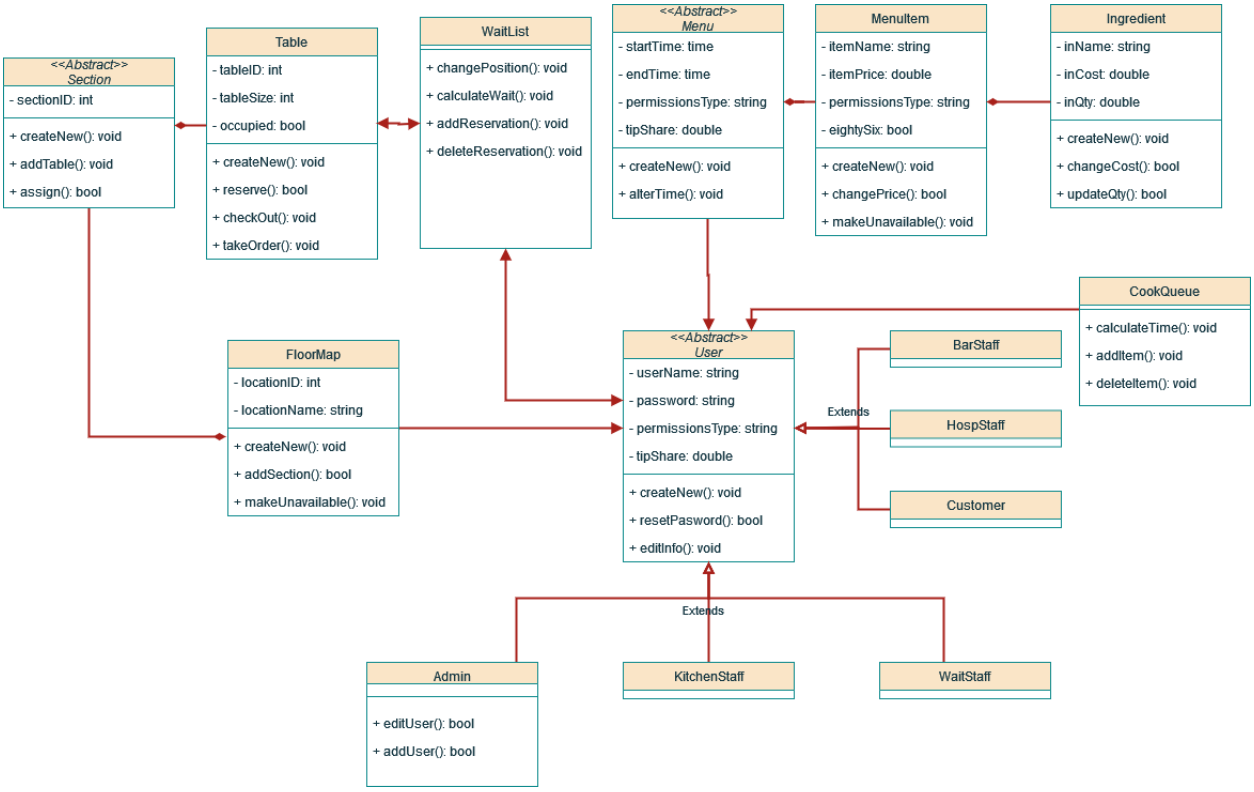
System (total: 20)

- Calculates tip share: uses employee title to calculate their share of tips based on worked hours (2)
- Modify inventory: when an item is ordered, it reduces the inventory quantity of ingredients that were used to make it (4)
- Makes items unavailable: when an ingredient's inventory quantity drops to zero, all items that require it are 86'd from the menu (6)
- Notifies staff when waitlist is at capacity: uses data from the cook queue and waitlist to determine when it is impossible to take further reservations (8)

Diagram



Class Diagram



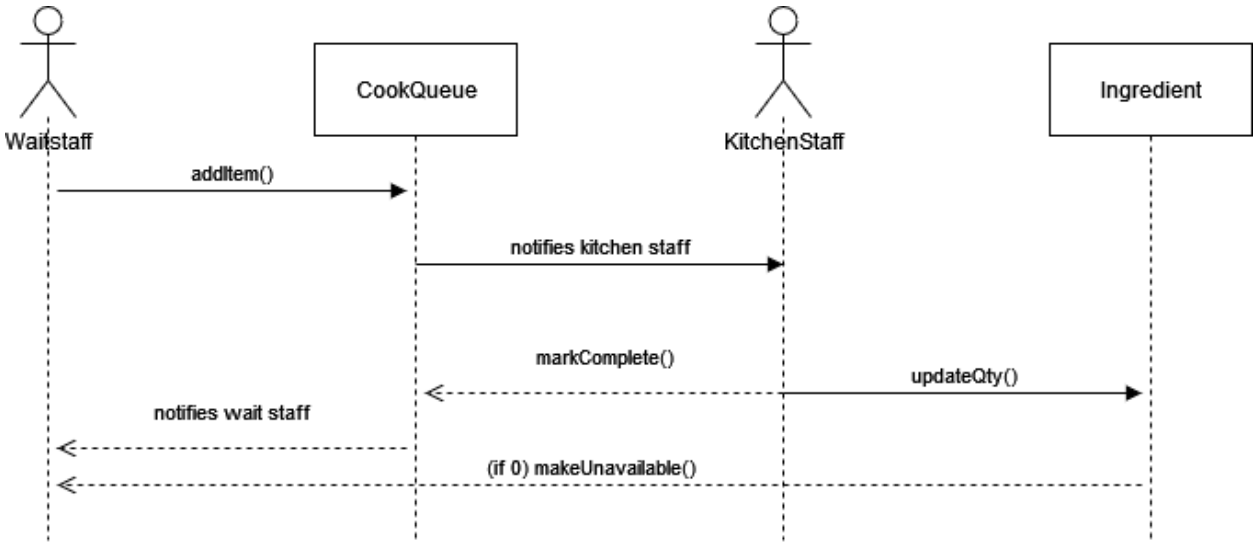
Sequence Diagrams

Actors: WaitStaff and KitchenStaff

Objects: Ingredient and CookQueue

Steps of encounters with the cook queue

1. Waitstaff enters order
2. Item is added to the cook queue
3. Kitchen staff is notified
4. If complete, waitstaff is notified, and ingredient quantities are updated
5. If ingredient is depleted, system notifies kitchen and waitstaff that items are 86

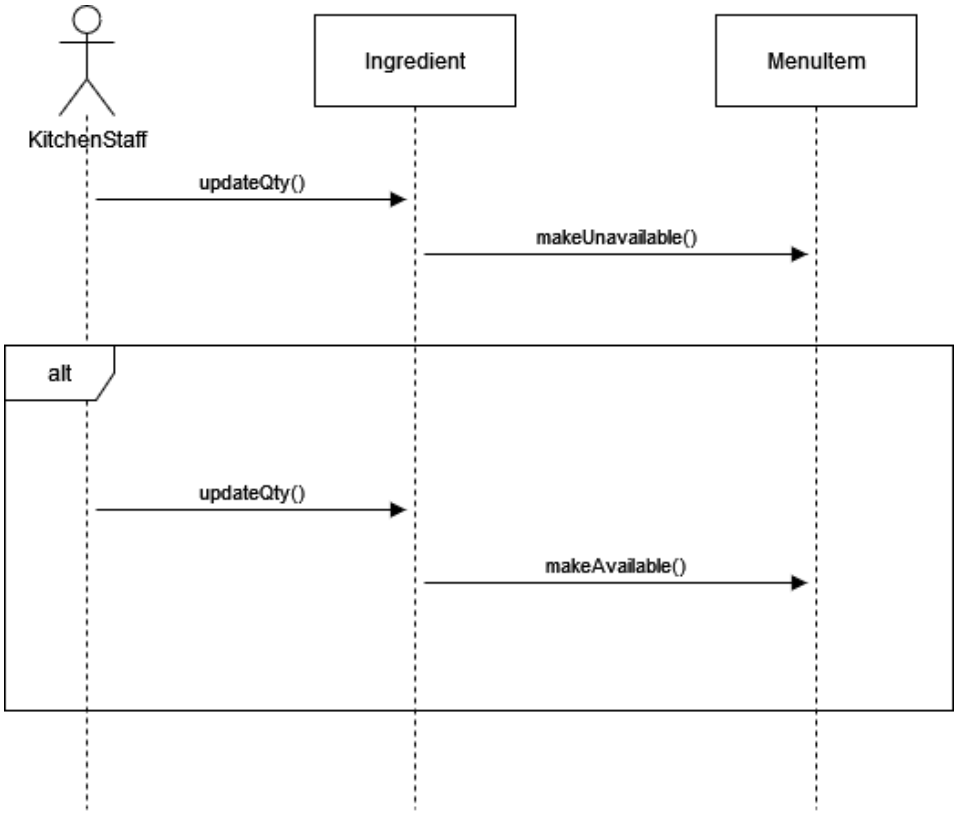


Inventory changes

Actor: KitchenStaff

Objects: Ingredient and MenuItem

1. Kitchen staff updates ingredient quantity
2. A) Ingredient quantity is zero, so MenuItem is made unavailable
   B) Ingredient quantity is changed to above zero, so the item is made available



Activity Diagrams

Waitstaff takes customer order

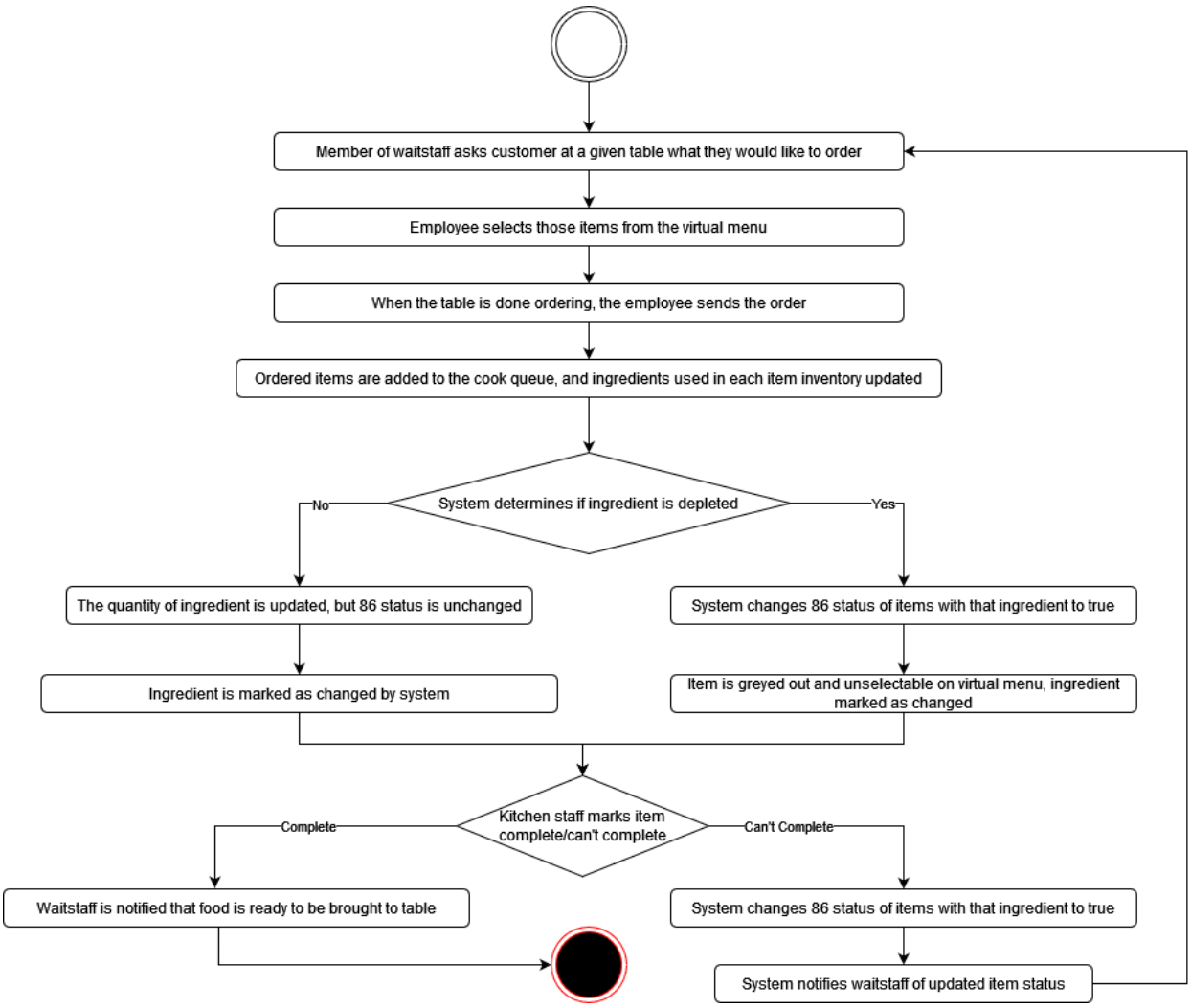
States

Initial: customer is seated at a table

Final: 1. customer receives their order 2. Item is unavailable, so the customer must make another order

Actions

The customer is seated at the table. An employee takes the customer order and enters it into the system. The system notifies the kitchen staff an item has been ordered and updates ingredient inventories. If an item is completed, a member of kitchen staff marks it complete. The system tells the server that it is time to deliver the food. If the item can not be made by the kitchen staff, the kitchen staff enters “unable to complete.” The system then 86’s the item.



Customer is seated or added to the waitlist

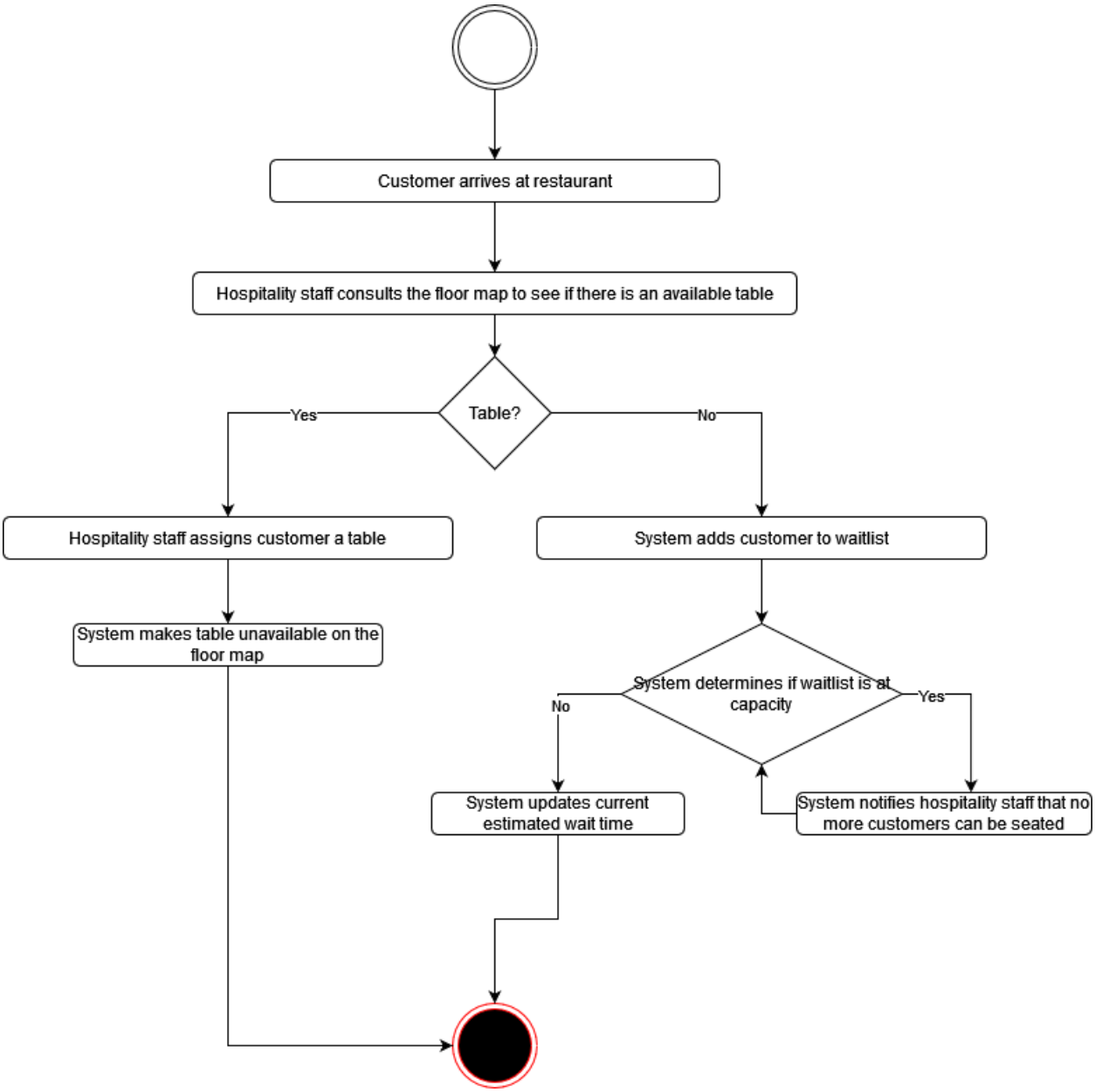
States

Initial: customer arrives at hospitality desk

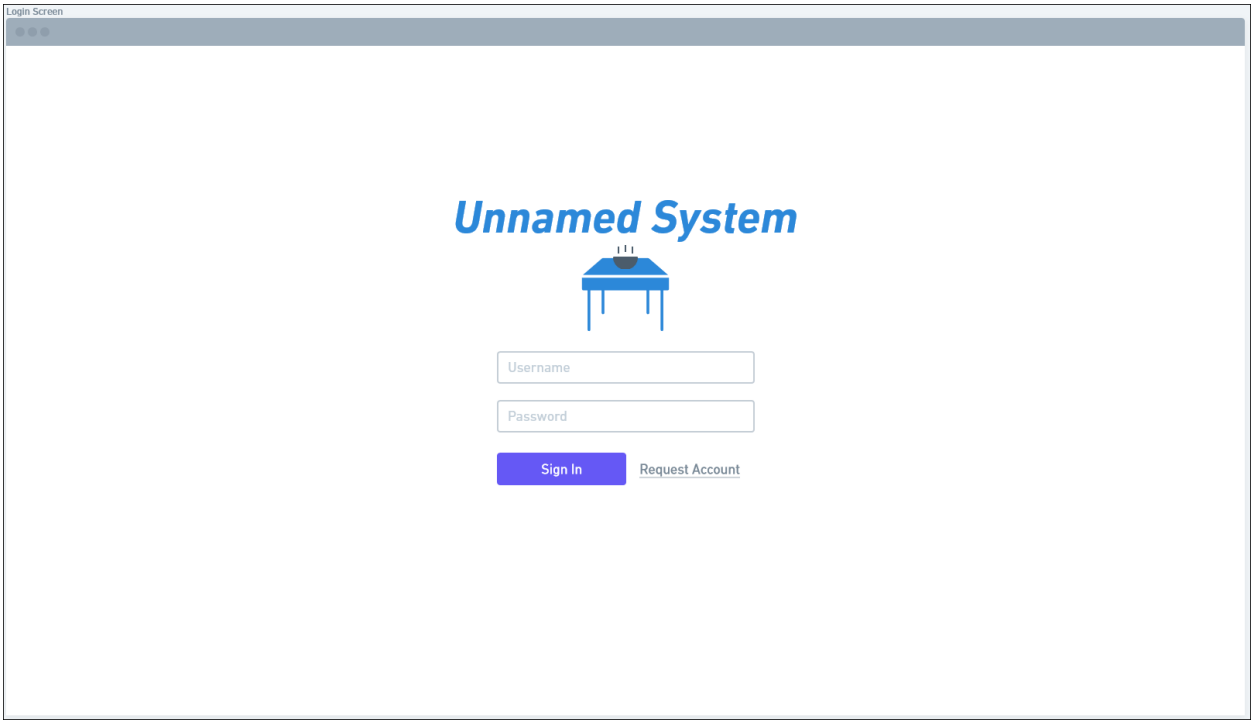
Final: 1. customer is seated 2. Customer is added to the waitlist

Actions

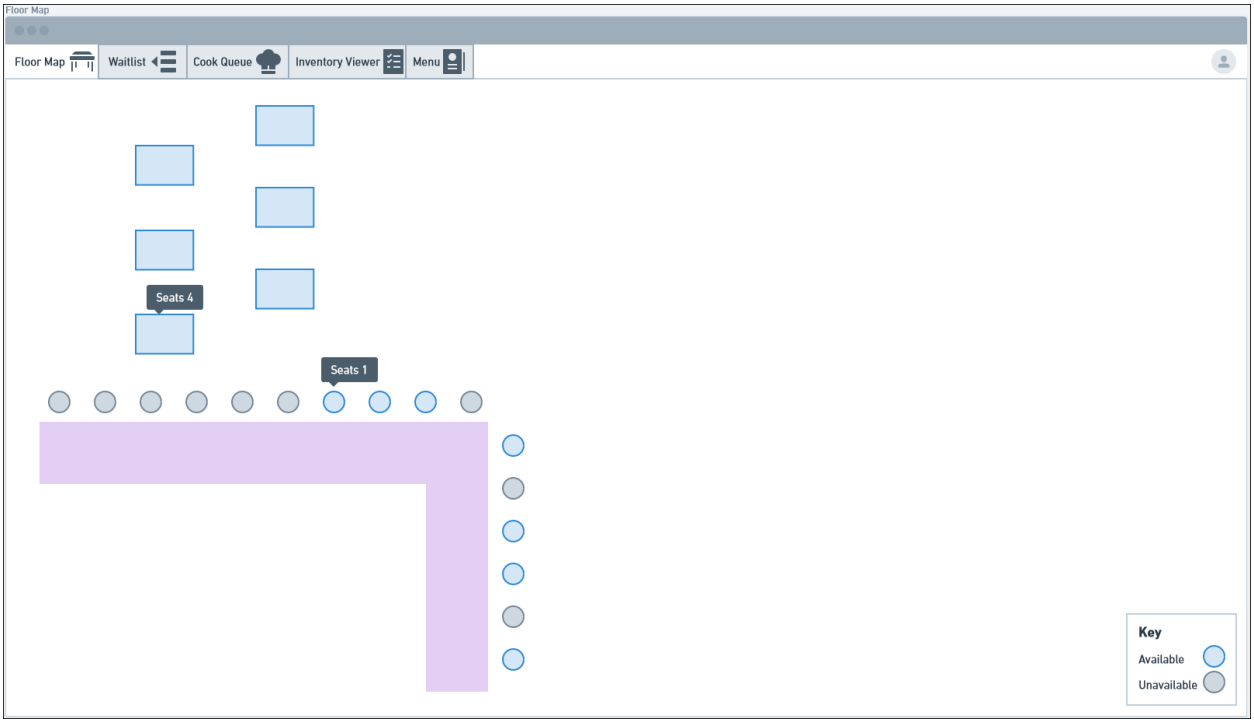
A member of hospitality staff consults floor map to determine availability. If table is available, customer is seated. If not they are added to the waitlist. The system then determines if the waitlist is at capacity, if so hospitality staff is notified. If not, estimated wait time is updated.



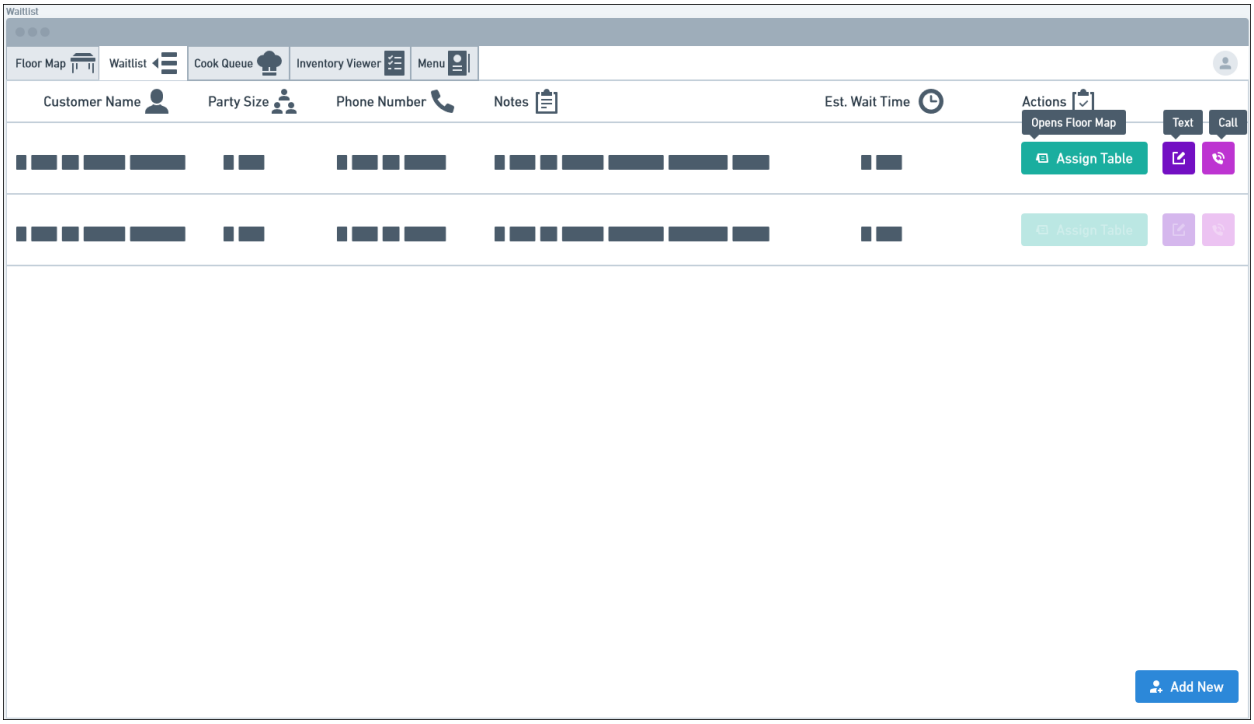
UI Mockups



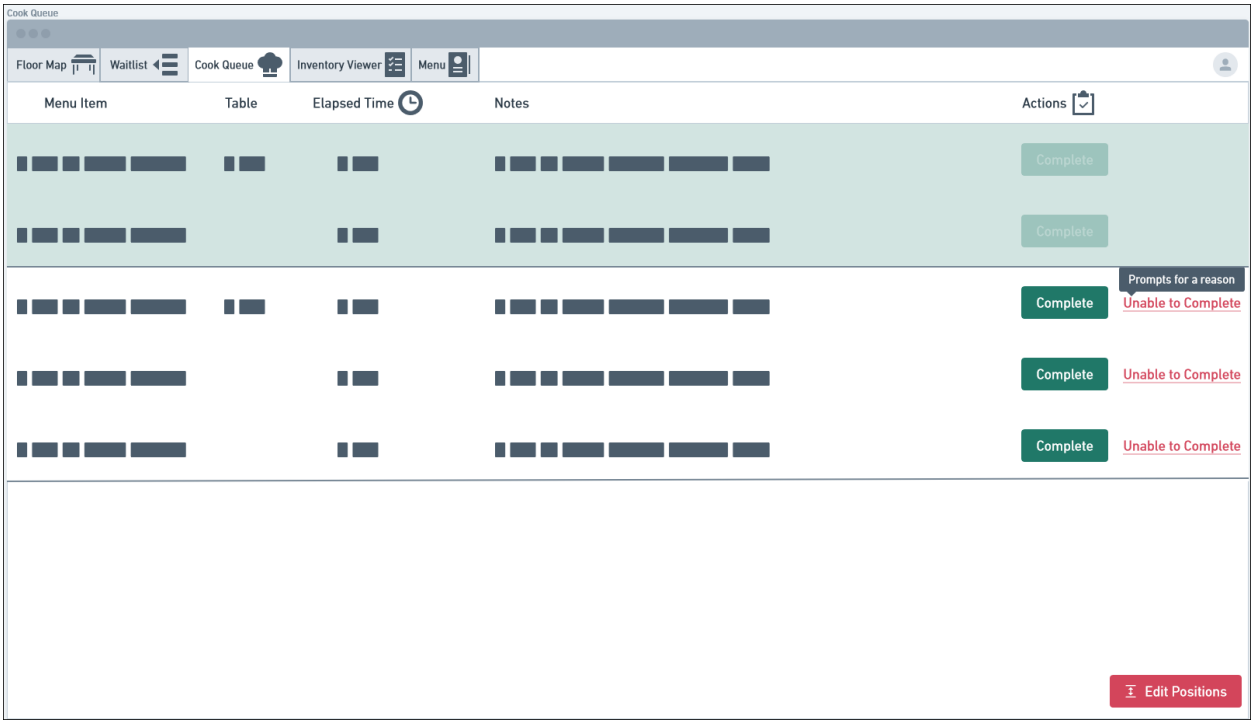
The login screen which will be shown on full sized devices. All users will see this screen and interact with it to login. The request account button will send a notification to users with admin privilege to approve or deny,



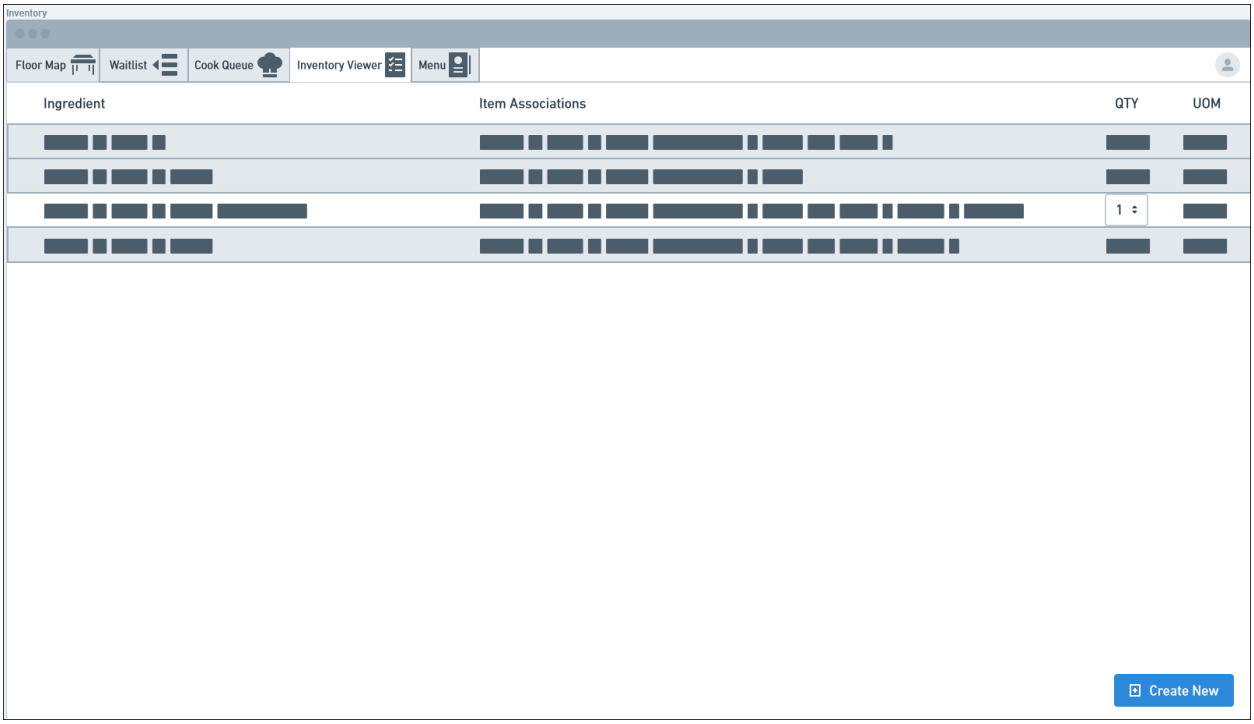
This is what a typical floor map may look like. Upon mouse/hover over a tooltip appears over each selection that explains to the user how many customers may sit at each table. Blue indicates available tables, and grey represents unavailable ones. Hospitality staff may click on a table to assign it to a customer. This is also the screen that they will be taken to if a table becomes available for a customer on the waitlist.



This is what the waitlist will look like. After adding customers, they will be shown as seen above. Actions can be taken when a table becomes available for a given customer, but they may not if there is nothing available. The estimated wait time is updated in real time based on how long seated tables are taking to leave, and the size of the customer's party. When a table becomes available, hospitality staff can assign customers a table, send them a text message to notify them that a table has come available, or call them for the same reason. There are tooltips which tell users what will happen if they click an action.



This is the cook queue. It shows the kitchen staff and serving staff the status of each item, the amount of time since the order was placed, and allows the kitchen staff to complete or not complete each item in the queue. The items are grouped by table so that each customer at a table will receive their food at a similar time. When all items for a given table are complete, that section turns green. If a member of kitchen staff is unable to complete an item, they will be prompted to enter a reason, as indicated by the tooltip. Members of kitchen staff may also move items around on the list as they see fit by clicking “edit positions.”



This is the very basic view of inventory. It shows the ingredient, all of the menu items that it is associated with, and the quantity and unit of measure (UOM). When an ingredient is selected, its quantity can be changed. Certain employees will also have the ability to add new items by clicking “Create New.”

User Effort Estimation

Use Case	Mouse Clicks
Login	3
Assign a table to a customer	1
Add customer to waitlist	5
Complete an item on the cook queue	1
Update an item's inventory quantity	2

System Requirements

Number	Priority Weight (1 - 5: 1: lowest, 5: highest)	Description
Req-1	5	The floor map should contain a visual representation of every table and the number of seats at that table.
Req-2	3	A member of hospitality staff should be able to determine wait times based on information on the waitlist
Req-3	4	Wait staff and kitchen staff should be able to determine whether a menu item is 86'd or not (with kitchen staff having the ability to 86 menu-items as needed)
Req-4	4	The system should have inventory information for each ingredient that is used by the restaurant, which updates when dishes/drinks are prepared
Req-5	2	The system should be able to calculate an employee's tip share on a given shift and provide that information live as it changes.
Req-6	1	The system should allow for multiple ways of sorting inventory information, with the default being alphabetically
Req-7	5	The system should not accept more reservations than are physically possible for a given time frame
Req-8	3	Items should be automatically 86 based on ingredient inventory
Req-9	3	All aspects of the system should be accessible to team members, with editing capability based on user permissions
Req -10	1	The system should have a superuser permission which can grant permissions and change them for each team member

Use Cases

Number	Description
UC1	Add/delete items from an order
UC2	Login/logout
UC3	Add/remove customer from waitlist
UC4	Add/remove items to the cook queue
UC5	Manually update ingredient inventory
UC6	Change a table's availability
UC7	Add/delete accounts
UC8	Grant Permissions
UC9	Change items and ingredients in the inventory
UC10	Calculate tip share
UC11	Notify staff of waitlist capacity
UC12	System makes items unavailable
UC13	System modifies items or ingredients

Traceability Matrix

REQ	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13
Req1	5			X			X							
Req2	3			X			X					X		
Req3	4	X			X								X	
Req4	4	X			X	X				X			X	
Req5	2			X							X			
Req6	1	X				X				X				
Req7	5				X		X					X		
Req8	3	X								X			X	X
Req9	3	X		X			X		X	X				
Req10	1		X					X	X					X
Max PW		5	1	5	5	5	5	1	3	4	2	5	4	3
Total PW		15	1	13	13	9	16	1	4	11	2	8	11	4

Architectural Style

I am taking the three-tiered approach.

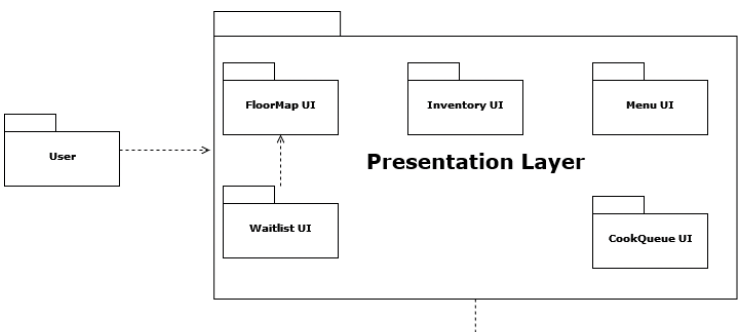
Presentation layer: the UI and all the ways it can be manipulated. This is the top tier.

- I would add that an admin will have considerably more control over the UI, so that could almost be its own layer. However, all interaction with the system will still remain in the UI, so it is not a true "tier."

Application layer: this is where all the logic in the back end code will exist. It is the middle tier, which draws from the data layer, and converts that data into something more digestible for the end user in the presentation layer.

Data layer: this is where the database is. Bottom tier. Manipulation of the presentation layer will filter through the application layer and ultimately update or create data in the data layer.

Subsystems



Sub-systems to Hardware

All data layer and application layer operations will exist on a central server. Presentation layer will be available on any device with an internet connection. The system will be accessible from user devices via HTTP protocol. I am selecting this because of compatibility; most devices will be able to access and use the system.

Control Flow

Execution Orders

The system is event driven. There is some linear path in the instance of waitlist to floormap operations, but for the most part, when a restaurant is not full, each table on the floormap will be in a state of occupied or vacant until changed by a user.

Time dependency

Different users will have access to the same information in real time. As such, the actions of one user must be visible to other users as well. The time estimations on the waitlist and the elapsed time on the cookqueue will be highly dependent on time.

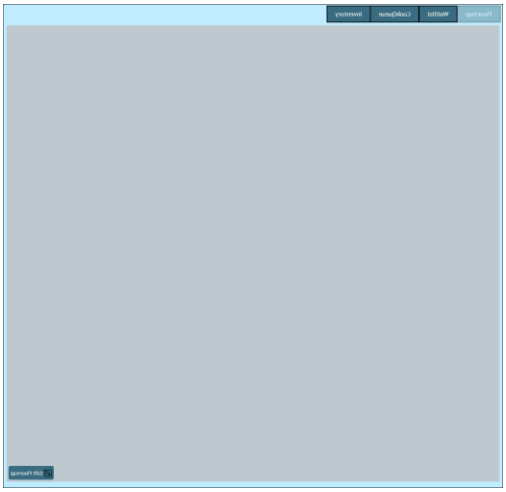
Hardware requirements

To run the program, a device must have a screen (color, 1280x720p preferred), an internet connection, and a way to accept text/mouse/touch inputs from a user. Nothing is very data intensive, so 10Mbps networking should be adequate for most users. Because most of the program will run on a central server, the processing requirements for an end-user device are minimal. Server requirements are unknown at this time because the program is not complete.

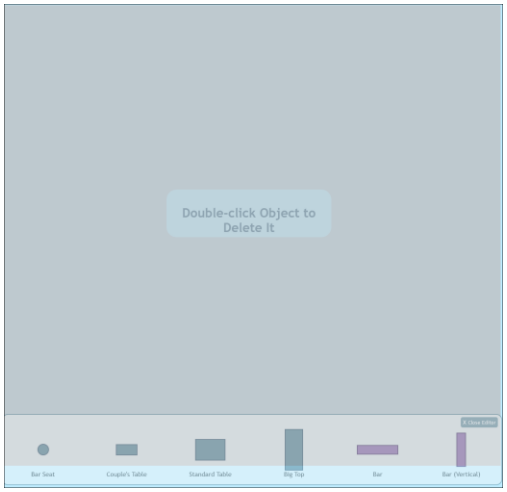
UI Changes

The final product is fairly true to the final product. I was able to design each section roughly the same way. In contrast to where I was with the UI at the midterm, the floor map is where the most progress was made. I was able to make the map entirely editable . In addition, I added some things that were not initially planned to give the user feedback regarding their inputs. This is most notable on the cook queue when an item is completed or not.

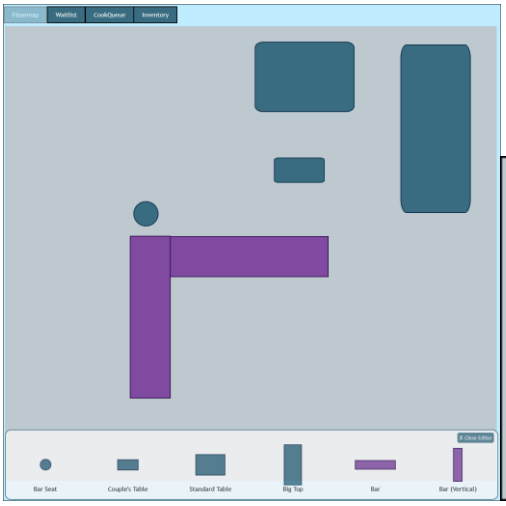
Pictures



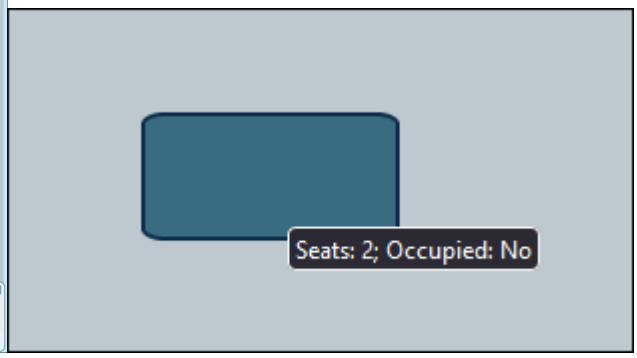
Blank Floor Map



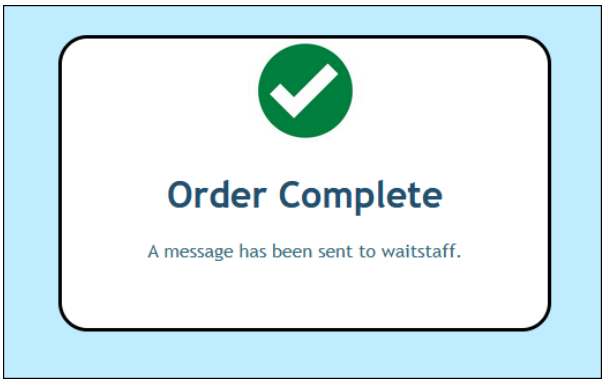
User Message when editor opens

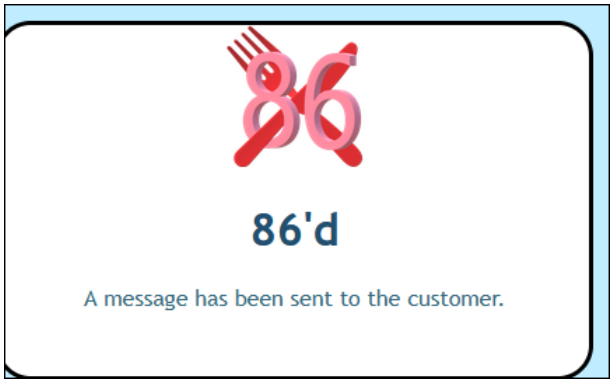


Map with user added items



Tooltip shows status and occupancy of table





Message shown when an order is completed Message shown when it cannot be

Tests

The main thing I am interested in testing is the ability for the system to update the database. I will input data into the UI and then see if it is written to the database, and if the UI can then read back the database entries. While I am testing, I will use the php error message functionality to see what needs to be fixed. The following will be conducted:

Can the UI update inventory quantities?

Can the UI read data to display available tables?

Can the system recognize when the restaurant is full?

What scenarios is the UI not responsive to user input?

Since this is a fairly simple system, complex testing is not really possible. I will do my best to conduct as many tests as possible to cover many scenarios.

References

I researched syntax on the following websites:

https://stackoverflow.com/

https://www.w3schools.com/

https://helpx.adobe.com/coldfusion/developing-applications/accessing-and-using-data/introduction-to-databases-and-sql/using-sql.html

https://developer.mozilla.org/en-US/docs/Web/HTML

https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX

https://jquery.com/

https://www.geeksforgeeks.org/javascript/

Wireframes created on

https://whimsical.com/

Development Environment

Visual Studio Code: https://code.visualstudio.com/

XAMPP: https://www.apachefriends.org/

MySQL: https://www.mysql.com/

phpMyAdmin: https://www.phpmyadmin.net/