Restaurant Automation

Report 1 Full

February 12, 2017

Project Team

Mit Patel

Raj Patel

Nill Patel

Dylan Herman

Prabhjot Singh

Moulindra Muchumari

Project Website: https://github.com/Mitbits

# All team members contributed equally!

Table of Contents

Customer Statement of Requirements

Running a restaurant with pen and paper in the age of technology is getting very inefficient. In order to keep up with the competition and take advantage of the opportunities provided by new technologies. We would like to have a system that will integrate all the employees and allow for seamless tracking of each task. Automation decreases cost by hiring less people and helps to remove the human error that causes customer dissatisfaction.

Customers don’t have a way to express feedback regarding their experience

Sometimes our customers may have a bad experience at our restaurant that can lead to a loss of customers. If we are aware of such negative experiences, we can prevent them from occurring in the future, thus increasing customer satisfaction. Currently, submitting complaints at our restaurant comes at a cost of time to both the manager and the customer. Usually, our customers do not want to go out of their way to waste time submitting a complaint. Moreover, spending time attending to customer complaints comes at an inconvenience to the manager because he or she must temporarily stop overseeing restaurant operations.

Imagine a full time employee at an office decides to take their one-hour lunch break at our restaurant across the street. The customer arrives at our restaurant, waiting to be seated. A waiter unwelcomingly greets the customer. Usually, such an incident would not be worthy of complaining about to the manager. The customer is seated, and the waiter takes the customer's order. The food arrives, but the customer receives the wrong item. The waiter sighs and brings the food back to the kitchen. Fifteen minutes later, the customer’s food finally comes out. After finishing his/her food the customer pays the bill and leaves the restaurant in a hurry because they had to go back to work. The customer was unable to give the restaurant’s manager feedback about the poor service they received and most likely wouldn’t go back there to eat.

We would prefer a solution that would allow us to easily receive customer feedback without coming at a cost of time to us or the customer. Also, we would like to be able to view and save customer feedback for personal records to prevent any feedback from being missed.

There is a communication issue between the host and the customer to notify the customer when he or she can be seated

When customers arrive at our restaurant, they want to spend less time waiting, and more time dining with their friends and/or family. During busy hours, wait times can be as long as one to two hours long, meaning that the lobby will be crowded. In such circumstances, it is difficult for the host/hostess to notify a customer that their table is ready above all the noise and ruckus. Miscommunication errors can lead to inefficiency in the seating system, thus increasing overall wait time and customer unhappiness.

Assume a family of six people visit our restaurant at 6PM on a Friday night to enjoy a family dinner. The family enters the restaurant and the host informs them that the wait time will be one hour and 30 minutes to be seated. The family decides to take advantage of the long wait time by visiting the mall across the street. One hour later, an unexpected amount of tables were vacant, making room for the family of six. Unfortunately, the family was not present at the time they were at the top of the queue. The host spent 5 minutes looking for the family of six, unable to find them. As a result, the host assigned the table to the next customers in line. The family arrived 10 minutes later, only to learn they had to wait an extra hour.

We would prefer to have a system that will let us contact our customers when there is a wait time. The customers should be able to keep in contact with us to know if they would like to keep their reservation.

Managing customer checks both efficiently and in an organized manner

Using the old fashioned system to generate bills and keep records is cumbersome, inefficient, and unorganized. The current system requires our waiters to manually calculate subtotals for a customer's bill, which can be time consuming and prone to errors. Moreover, keeping records of all checks for future references requires extra work and responsibility for the manager of our restaurant.

Imagine a party of 15 walks into our restaurant, and they order a lot of food. When it comes time for the bill, the waiter has to start manually adding up each order to get the total for the bill. This can take quite some time because the waiter has to make sure the bill is accurate. After the bill is paid, the waiter adds the bill to the daily stack of bills which will then need to be logged.

One possible solution that could help us solve this is a system that will help us remove the manual calculation of bills and store the data on a computer. This will help save time for both the customer and the restaurant.

Inefficiency with table management & cleaning

A common problem we find in our restaurant is the inability to keep track of unoccupied or dirty tables, which leads to inefficiencies in seating our customers. This means we lose potential revenue because our customers experience longer wait times and are not satisfied.

When a customer comes into our restaurant they first approach the host to be seated. The host has to look at the whiteboard and see which tables are available and seat the customer(s) accordingly. At busy times in our restaurant, the customer may have to wait a significant amount of time for a table to become vacant. However, the wait time is longer than it should be because the whiteboard doesn't get updated instantly when the busboy cleans the table.

A solution we believe would solve this problem is to have a system that allows us to view the dining area in realtime. It would display when tables are vacant, occupied or dirty using different colors and patterns to distinguish between the three statuses. This would allow us to view instantaneous updates about the status of the table and decrease wait time and increase customer satisfaction, when compared to the traditional whiteboard method.

Splitting checks & generate bills

When parties finish their meal, there are instances where they forget to mention splitting checks in the beginning and inform the waiter of splitting checks when the total check has already been calculated. In such a case, the waiter would have to ensure the orders of each individual and split the checks accordingly. In such a situation the waiter would have to go back to the cashier and inform them to print individual checks for each customer. Situations like these usually become a hassle for both the waiter as well as the customers. This leads to customer dissatisfaction, adds to the wait time of the customer(s) which in turn makes that table unavailable for a longer period time for the next customer(s).

A possible solution to this should allow us to split the checks whenever necessary, so this way we don't end up wasting time and resources and can seat the incoming party as soon as possible.

Wasted time in relaying the order to the kitchen

In our restaurant, the waiter has to jot down the order on a notepad for the kitchen and take a carbon copy to the cashier. This is very inefficient in the sense that the waiter will have to make multiple trips between the table, kitchen and the cashier. Doing so for multiple tables, takes away time from serving our customer's needs. The kitchen notifies the waiter when the order is ready by ringing a bell, but the waiter is not going to know whether the bell was rung for his order or a different one. This wastes the time of our waiters and causes them to run around for no reason. Also, we had previously lost or damaged the paper copies. Sometimes we have issues reading the waiter's handwriting.

A possible solution is to make this entire process electronic. The waiter could take the order on a tablet or some device instead of jotting down on a notebook. Then, the order could be sent electronically to the kitchen and the cashier. Also, the same system could notify the waiter when his order is ready. We think this would really decrease the amount of trips to be taken by the waiter. Moreover, we think it would be easier for us to process the bills.

Prioritizing and managing the flow of orders in the kitchen

Our kitchen is typically a chaotic place. Most of chaos results from us having to keep track of all the orders and which of our waiters they can from. However, not only do orders have to be served in a first-in first-out fashion, but the time of delivery made needs to dynamically change based on multiple factors such as: fairness, is our restaurant having a slow day and they want to keep as many customers in the restaurant as possible to appear "busy", does a customer want their food to be held until they finish their appetizer?, and etc. Thus managing the kitchen is more complex than simply queueing up orders as they come. That's why adding automation can create order in our kitchen.

A solution we believe would solve this problem is to have a system that automatically keeps track of orders as they come in. However, in our restaurant the orders that come first are not always the orders that get dished out first. We would need to have multiple order queues to make sure all tables get the food in a fair manner (appetizers come first, etc...). We would also like the system to automate the delegation of tasks (which cooks cook what). These multiple queues would also be help for customers that push orders back; they should be put behind other orders that have not been dished out yet.

Managing and keeping track of all the raw materials in the kitchen

Our restaurant has problems with tracking raw materials in the kitchen. We usually order the raw materials for the next week so we receive them on Sunday before the week starts. Since we can't know exactly how much we will use, we order more than required and throw away extra materials at the end of the week. We lose a lot of money and materials because of this. We tried to keep track of the raw materials every day by noting them on paper. But, our cooks found it difficult to note down what they used while cooking. Sometimes we were wrong with our estimate and we ran out of materials during the week.

We would like a solution that electronically deducts materials whenever the cooks finish an order. Then, the system could notify us to if any materials are running low, so we could order them on the go.

Planning the menu and activities for the day

Many restaurants plan activities the night before to streamline cooking for the next day. This allows them to serve more customers and reduce the waiting time for customers to receive their food. A lot of restaurants do not plan these activities properly or allocate enough time to finish them the night before. Hence, this reduces productivity next day and increases stress and unhappiness in the kitchen. Moreover, this might directly affect the revenue of the restaurant and decrease customer satisfaction.

A solution is to combine the planning activities with the cooking activities, so that planning for the next day happens concurrently with serving customers. This solution might work because the planning activities can be completed when the restaurant might not be busy. Also, this simplifies managing the kitchen for the Chef by allowing one system to assign activities for the cooks.

On many days, we are unprepared for cooking items in the kitchen because we did not prepare the required materials properly the day before. We would like a solution that will help us properly prepare the required materials for the next day to improve the efficiency in our kitchen.

Cleaning dishes

Busboys in our restuarant perform two main tasks: clean tables and clean dishes. They have be in both the kitchen and the dining area and it causes a lot of running around. It is also hard for the chef to find a busboy to clean dishes. This decreases the efficiency of the chef because he needs to be in the kitchen to manage the cooks. Some restaurants mitigate this problem by having some busboys stay in the kitchen as dishwashers. However, this is an inefficient solution because the dishwashers are not needed when there are no dishes to be washed and the busboys don't need to be in the dining area when there are no dirty tables. Moreover, this requires us to hire more people for this solution.

We would like to have a system that will tell the busboys when the dishes or tables need to be cleaned. The chef's should be able to contact any of the busboys that are free. Overall we want less clutter in the dining area and kitchen.

Glossary of Terms

Manager - Responsible for managing employees, overseeing the daily operations of the restaurant, and ensuring profitability of the restaurant.

Kitchen - The area where the chef(s) cooks and prepares food. Also, the busboy(s) transports and cleans dishes in this area.

Chef - A professional cook that prepares food for the restaurant.

Customer - A person that purchases food or service at the restaurant.

Waiter/Waitress - Responsible for taking orders and completing requests for customers to ensure customer satisfaction.

Host/Hostess - Manages the restaurant lobby and makes seating arrangements for customers .

Busboy - Cleans dishes and clears tables for the restaurant.

Dining Area - The area where customers eat their meals.

Cashier - A waiter responsible for completing customer transactions.

Split Check - A check that is customized based on all the items purchased by one individual.

Lobby - The area where the customer checks into the restaurant and waits for a table.

Technical Terms

Graphical User Interface - A visual interface that allows workers at the restaurant to interact with the system

Order Queue - lists out food orders to the chef based on first come first serve

Database - A storage device that archives daily restaurant sales, inventory, and feedback.

Floor Plan - An interface that is used to show which tables are currently busy, free ,or dirty.

Notification - A way to alert employees so they can run the restaurant more efficiently.

Reservation - A system that allows customers to claim a table at a specific time before going to the restaurant

Stage - Represents the three categories of meals used in our order queue system i.e appetizer, entree dessert

Walk-in - When customers come into the restaurant without requesting reservation prior to coming in

User Stories

High (core) Priority
Medium Priority
Low Priority

Waiter

Identifier	User Story	Size
ST-W-1	As a waiter, I can remotely send customer orders directly to the chef	6
ST-W-2	As a waiter, I can choose to split the check multiple ways or generate one check.	5
ST-W-3	As a waiter, I can receive notifications when the food is ready for my table.	4
ST-W-4	As a waiter, I can receive notifications when a table requests my assistance.	3
ST-W-5	As a waiter, I can see how long each order will take approximately based on the data collected from previous orders.	5

Customer

Identifier	User Story	Size
ST-C-1	As a customer, I can remotely make a reservation for the restaurant.	4
ST-C-2	As a customer, I can receive remote notifications that indicate when my table is available.	2
ST-C-3	As a customer, I can alert the waiter to request assistance.	3

Host

Identifier	User Story	Size
ST-H-1	As a host I shall be able to see a floor plan of the dining room with which seats are available.	7
ST-H-2	As a host I shall be able to mark tables as reserved or available.	3
ST-H-3	As a host I should be able to see real time updates of the expected time a table should be done.	6

Busboy

Identifier	User Story	Size
ST-B-1	As a busboy I shall be immediately notified when a table needs to be cleaned	3
ST-B-2	As a busboy I shall be notified when I should go to the kitchen to clean dishes	3
ST-B-3	As a busboy I shall be able to mark tables as cleaned	2
ST-B-4	As a busboy I shall be notified when to bring more refreshers to tables.	3

Manager

Identifier	User Story	Size
ST-M-1	As a manager, I can see the status of every item in the inventory. Such as if an item is running low, it will give me an indication.	6
ST-M-2	As a manager, I can modify accounts and permission for each employee and add or remove new ones as necessary.	5
ST-M-3	As a manager, I can view all the transactions that took place in a given day.	3
ST-M-4	As a manager, I can view statistics on the sales in a given day.	4

Chef

Identifier	User Story	Size
ST-CF-1	As a chef, I can view orders that have been placed by the customers	4
ST-CF-2	As a chef, I can notify the waiter that the order is done	4
ST-CF-3	As a chef, I can manage the inventory, and notify the manager when items are running low	7
ST-CF-4	As a chef, I can prioritize and manage orders in the queue	6
ST-CF-5	As a chef, I can view the status of each table	3
ST-CF-6	As a chef, I can alert the busboy to clean the dishes	2
ST-CF-7	As a chef, I can make the use of raw materials more efficient using the data from the system	7
ST-CF-8	As a chef, since I pre-make dishes, I can get information on the most popular dishes of the past day, so I can decide which dishes to prep before tomorrow's opening.	6

Functional Requirements Specification

Stakeholders

Restaurant Owners/Managers - Restaurant owners/Managers would primarily have interest in our system because it provides a method of automation to optimize the efficiency of the restaurant. In turn, the owners would benefit from a rise in profit and a decrease in wasted resources (i.e. time, money, etc).

Customers - Customers benefit from the system in two ways without being the primary users of the system. The customers benefit from the improved service generated by the system itself (i.e. food tracking, instant service requests, etc). Also, the reservation system provides an easy way for customers to make reservations at a particular date and time without having to interact with a person.

Restaurant Employees - Restaurant employees benefit from this system because many of their daily tasks and responsibilities are automated by the system, thus increasing coordination, reducing human error, and reducing wasted resources.

Actors & Goals

Customer
Initiating and Participating
Role: Actor visiting the restaurant to order food and or drinks.
Goal: Customer makes reservations, places orders, and calls waiter.

Waiter
Initiating and Participating
Role: Employee of restaurant that is in charge of taking care of customers.
Goal: Waiter places the order, get notification when food is ready or when the customer requests assistance, and prepares the bill.

Host
Initiating and Participating
Role: Employee that greets customers in the lobby area.
Goal: Uses floor plan to assign tables to customers and manages reservations.

Chef
Initiating and Participating
Role: Employee that prepares all the food for customers in the kitchen
Goal: Gets orders from the waiter and prepares cooking. Notifies waiter when food is ready. Prepares food for the next day based on previous statistics. Notifies Busboy when dishes are full/dirty.

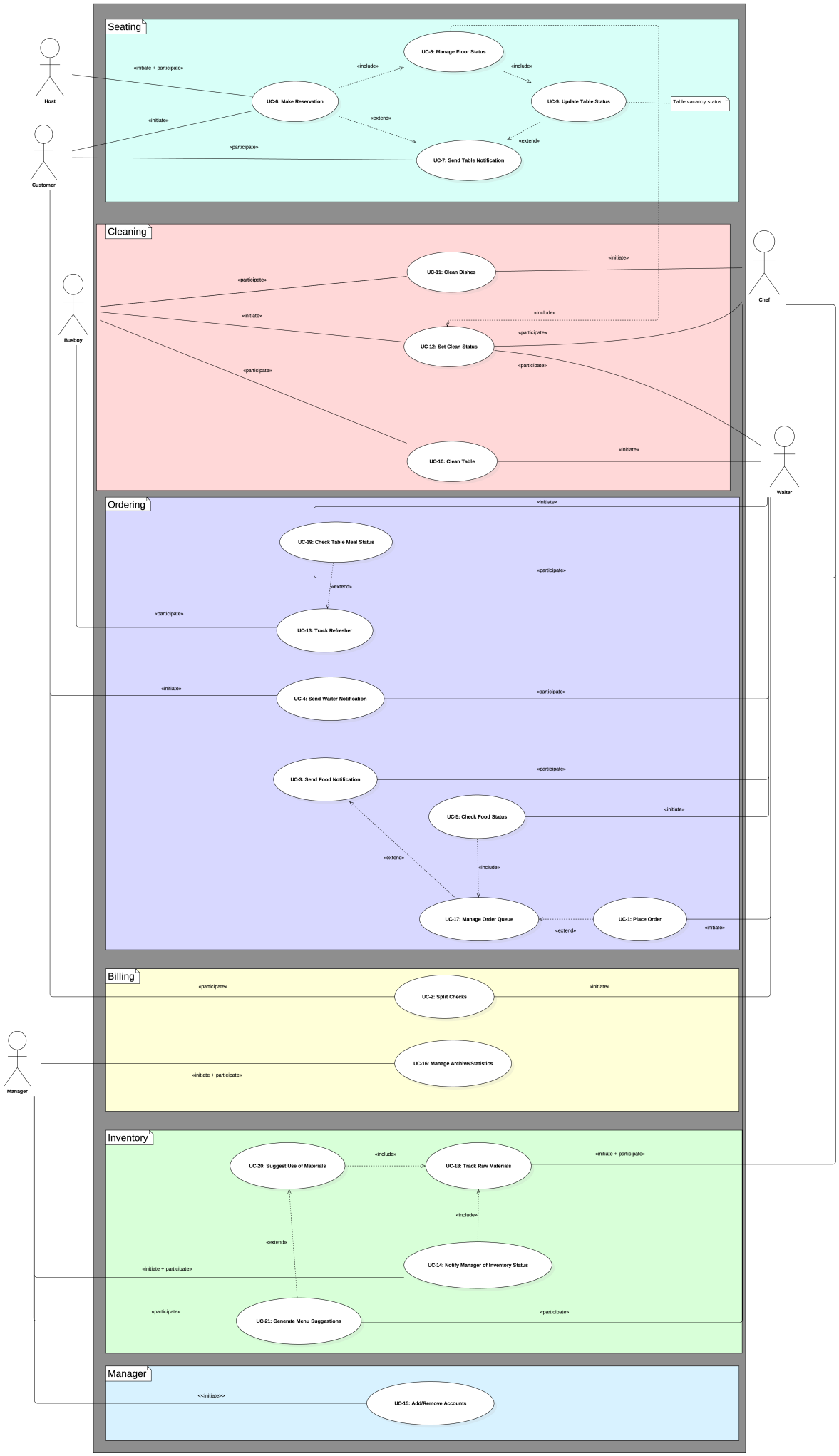
Busboy
Initiating and Participating
Role: Employee that is responsible for keeping the restaurant area clean
Goal: Busboy gets notified when he/she has to clean the tables and dishes.

Manager
Initiating and Participating
Role: Employee that oversees every other employee and manages the entire restaurant
Goal: Gather data from restaurant transactions and manage all employee accounts

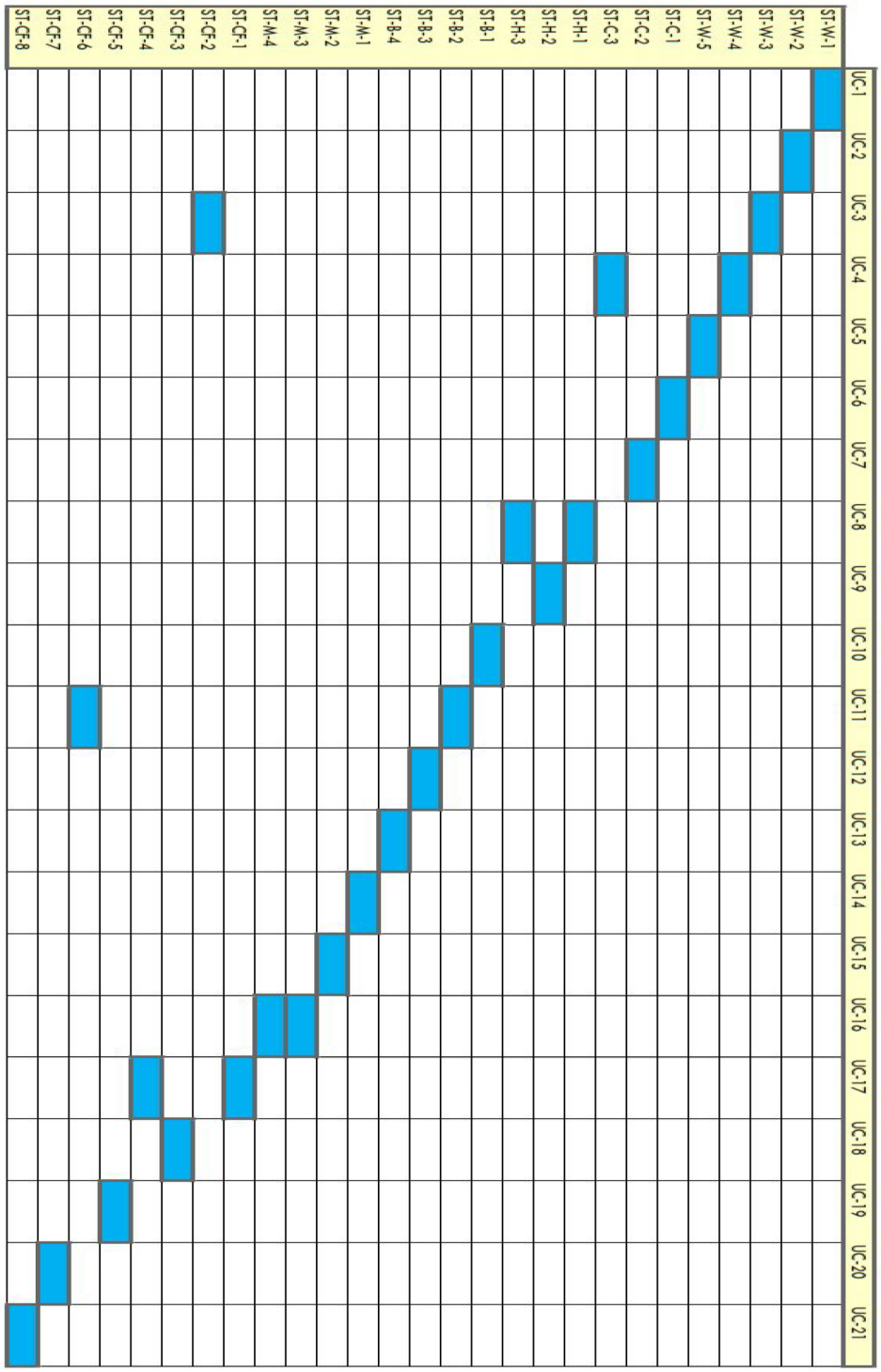
Use Cases

Casual Description

Our group's user stories accommodate the causal description requirement.


Seating
Host
Customer
«initiate + participate»
«initiate»
«participate»
UC-6: Make Reservation
«include»
UC-8: Manage Floor Status
«include»
UC-9: Update Table Status
Table vacancy status
«extend»
«extend»
UC-7: Send Table Notification
Cleaning
Busboy
Chef
Waiter
«participate»
«initiate»
UC-11: Clean Dishes
«initiate»
«include»
«participate»
UC-12: Set Clean Status
«participate»
«participate»
UC-10: Clean Table
«initiate»
Ordering
«initiate»
UC-19: Check Table Meal Status
«participate»
«extend»
«participate»
UC-13: Track Refresher
«initiate»
UC-4: Send Waiter Notification
«participate»
UC-3: Send Food Notification
«participate»
UC-5: Check Food Status
«initiate»
«extend»
«include»
UC-17: Manage Order Queue
«extend»
UC-1: Place Order
«initiate»
Billing
«participate»
UC-2: Split Checks
«initiate»
Manager
UC-16: Manage Archive/Statistics
«initiate + participate»
Inventory
UC-20: Suggest Use of Materials
«include»
UC-18: Track Raw Materials
«initiate + participate»
«include»
«extend»
UC-14: Notify Manager of Inventory Status
«initiate + participate»
«participate»
UC-21: Generate Menu Suggestions
«participate»
Manager
<<initiate>>
UC-15: Add/Remove Accounts

Traceability Matrix



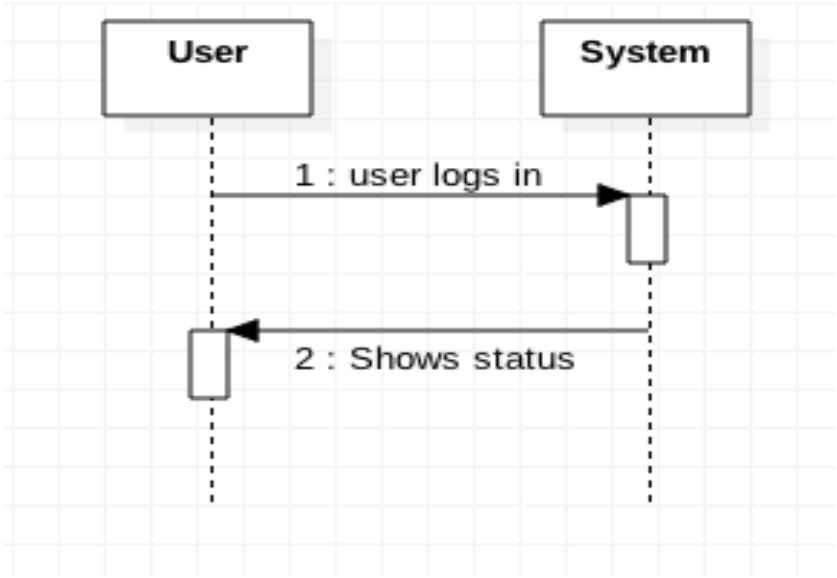
Seating	Ordering	Inventory	Cleaning	Billing	Manager
UC-6 Make Reservation	UC-1 Place Order	UC-14 Notify Manager of Inventory Status	UC-10 Clean Table	UC-2 Split Checks	UC-15 Add/Remove Accounts
UC-7 Send Table Notification	UC-3 Send Food Notification	UC-18 Track Raw Materials	UC-11 Clean Dishes	UC-16 Manage Archive/Statistics	
UC-8 Manage Floor Status	UC-4 Send Waiter Notification	UC-20 Suggest Use of Materials	UC-12 Set Clean Status		
UC-9 Update Table Status	UC-5 Check Food Status	UC-21 Generate Menu Suggestions			
	UC-13 Track Refresher				
	UC-17 Manage Order Queue				
	UC-19 Check Table Meal Status				

The traceability matrix above shows how the use case's map to the user stories previously described. It helps figure out which use cases play a crucial role in this system.

Fully-Dressed Description

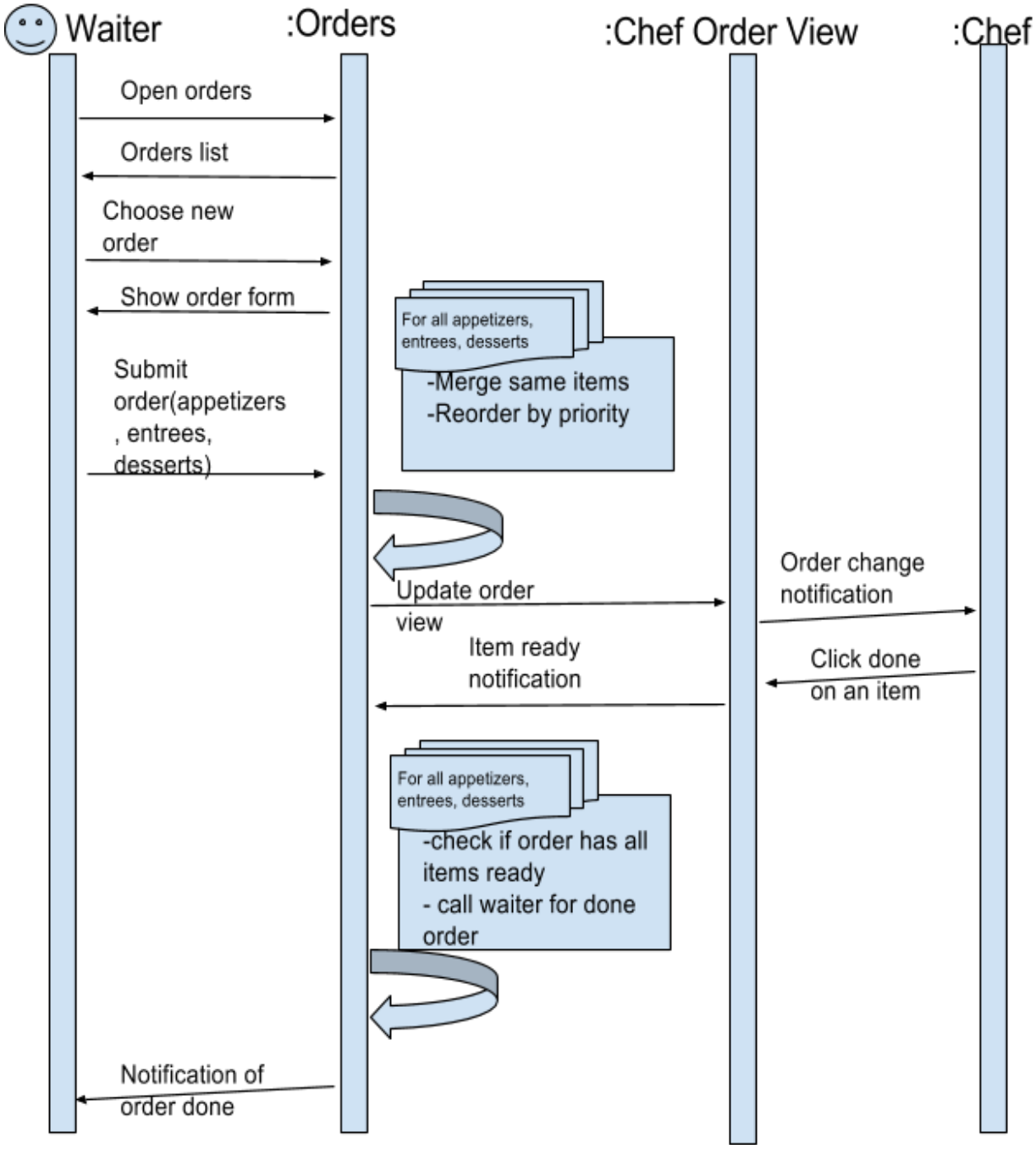
Use Case UC-8 Manage Floor Status
Related User Stories: ST-H-1, ST-H-3
Initiating Actor: Host
Actor's Goal: Uses floor plan to assign tables to customers and manages reservations.
Participating Actors: Busboy, Waiter
Preconditions: ● Database already includes Empty, Busy, Dirty and Reserved Status ● Shows the number of seats on each table
Postconditions: ● Each table shows correct status
Flow of Events for Main Success Scenario: -> 1) User logs into system <- 2) System indicates status using different color schemes and patterns

Sequence Diagram Use Case UC-8



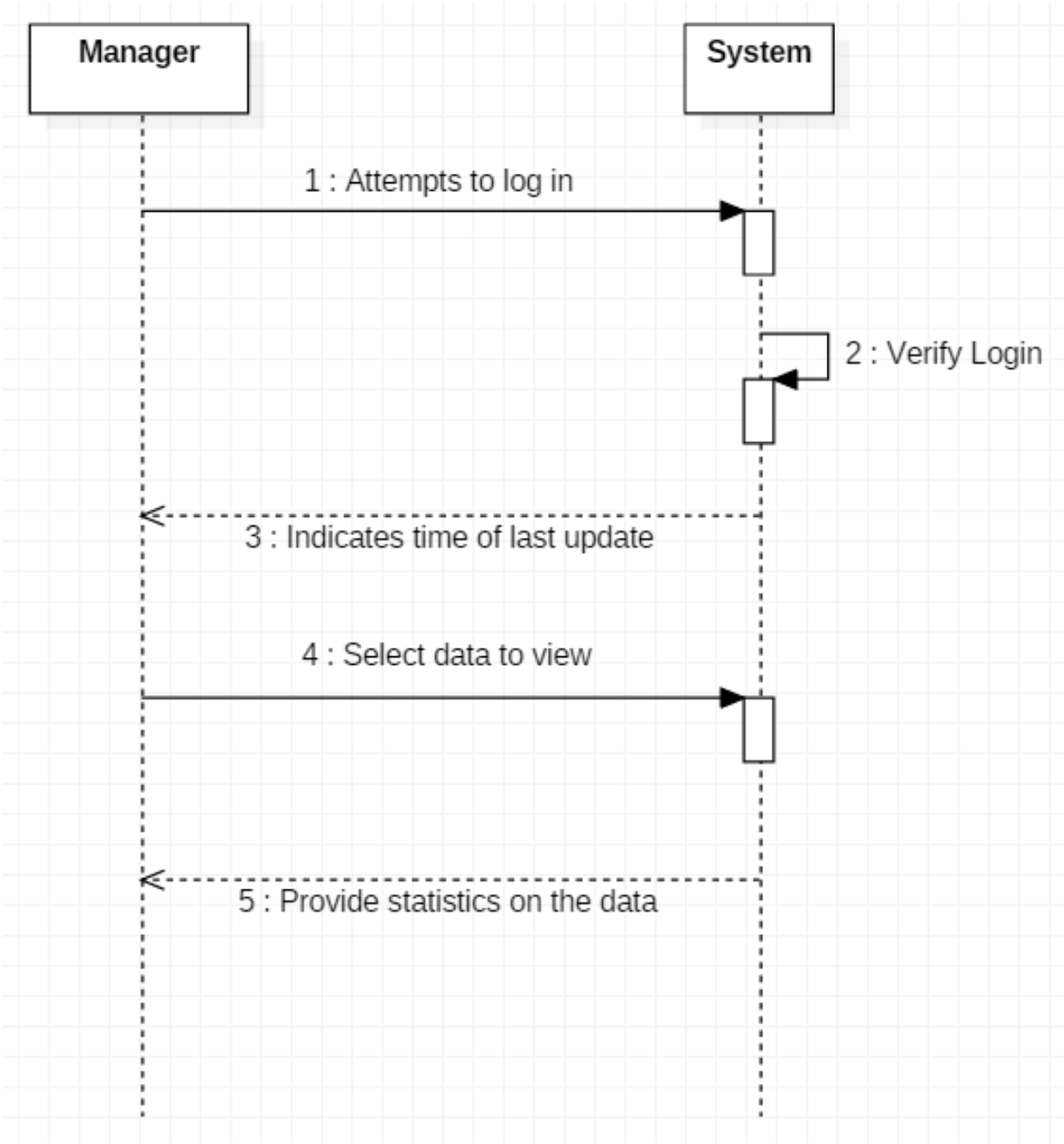
Use Case UC-17 Manage Order Queue
Related User Stories: ST-CH-1, ST-CH-4
Initiating Actor: Chef
Actor's Goal: Order placed by waiters are organized into a queue so that they can be distributed fairly to the customers. In addition, it must take into account changes in orders.
Participating Actors: Waiter, Customer
Preconditions: ● Customer placed an order with a waiter ● Other orders may or may not have been placed
Postconditions: ● The customer gets each of the portions of their order fairly (one table did not get its meals before another table). ● If the customer sends back an order the customer will get the corrected order in the proper time frame.
Flow of Events for Main Success Scenario: ->1) Waiter inputs order in the system. ->2) Order is organized into the queue system based on the stage of the meal (i.e. appetizers before main meal). <- 3) Waiter is eventually notified of the order.
Flow of Events for Alternate Success Scenario: ->1) Waiter inputs order in the system. -> 2) Order is organized into the queue system based on the stage of the meal (i.e. appetizers before main meal). -> 3) Order was possibly sent back for some reason. -> 4) Order is added back to the queue with the highest priority. <- 5) Waiter is eventually notified of the order

Sequence Diagram Use case UC-17



Use Case UC-16 Manage Archive/Statistics
Related User Stories: ST-M-3, ST-M-4
Initiating Actor: Manager
Actor's Goal: Be in charge of daily sales and transaction of the restaurant.
Participating Actors: None
Preconditions: ● There was at least one customer who visited and paid for the bill ● The database system was updated after the restaurant was closed
Postconditions: ● Each statistics gives accurate sales, averages, and sums
Flow of Events for Main Success Scenario: -> 1) Manager logs into system <- 2) System verifies login <- 3) System indicates time of last update -> 4) Manager selects data he/she wants to view <- 5) System provides statistics on select option

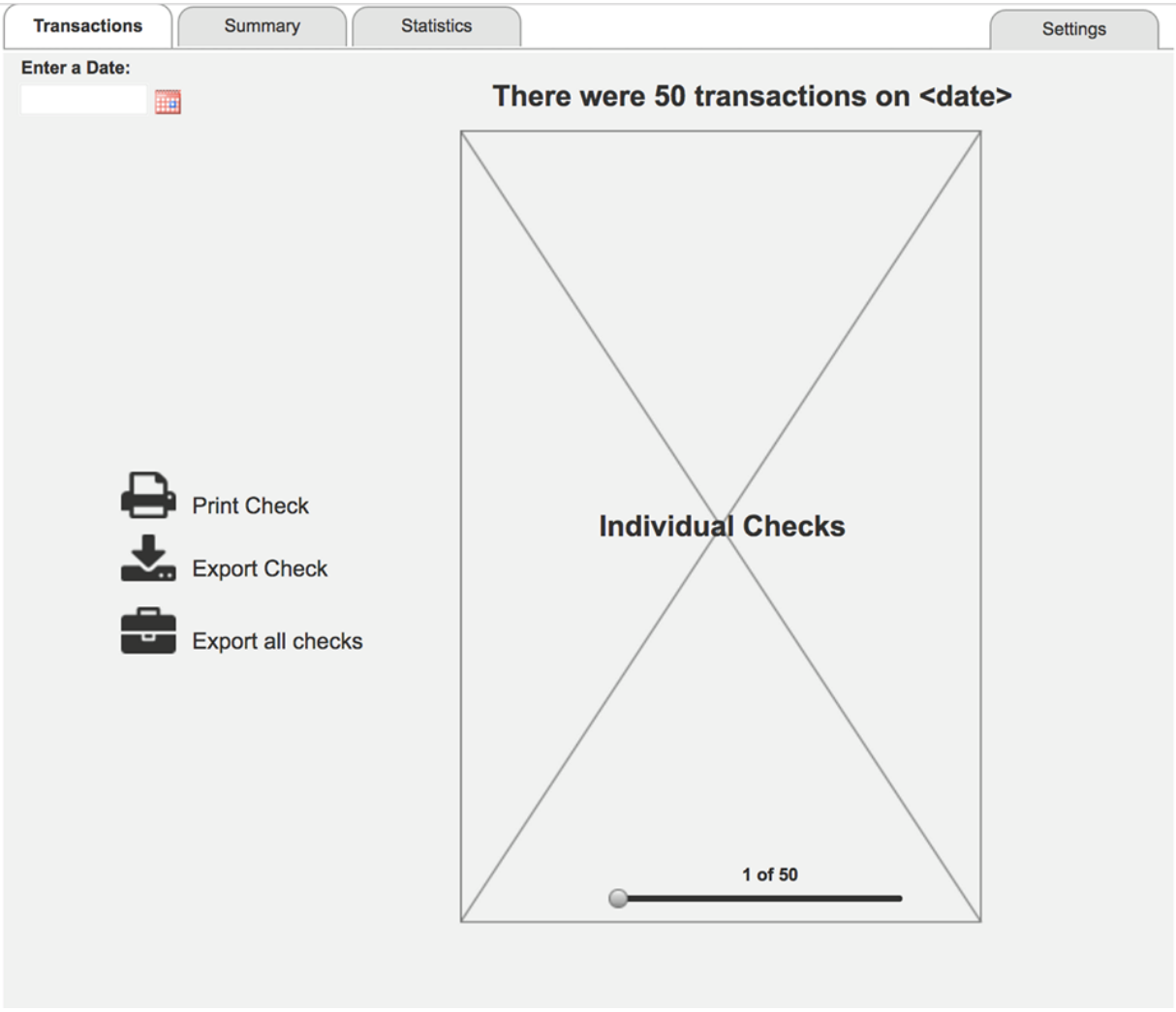
Sequence Diagram: Use Case UC-16

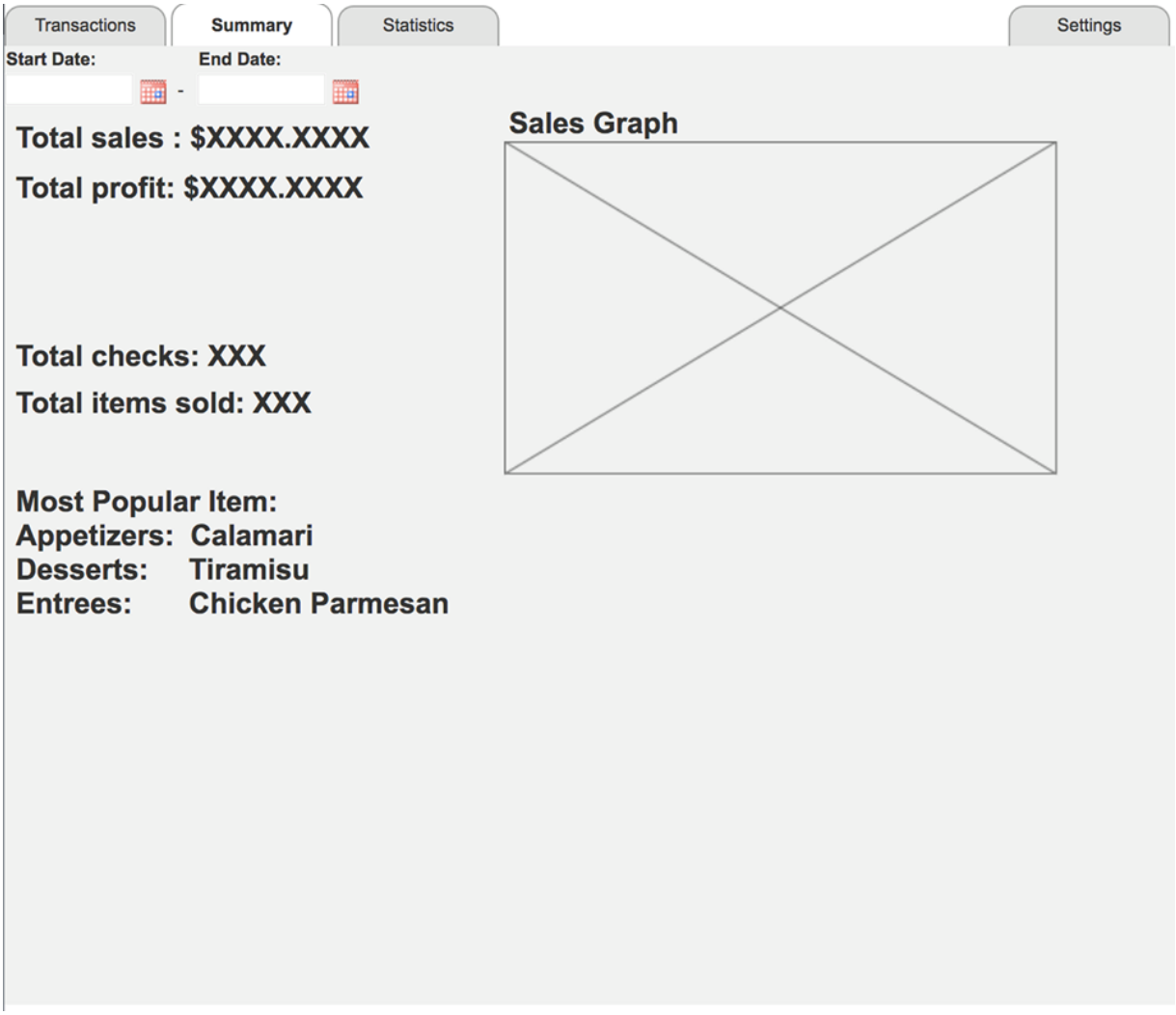


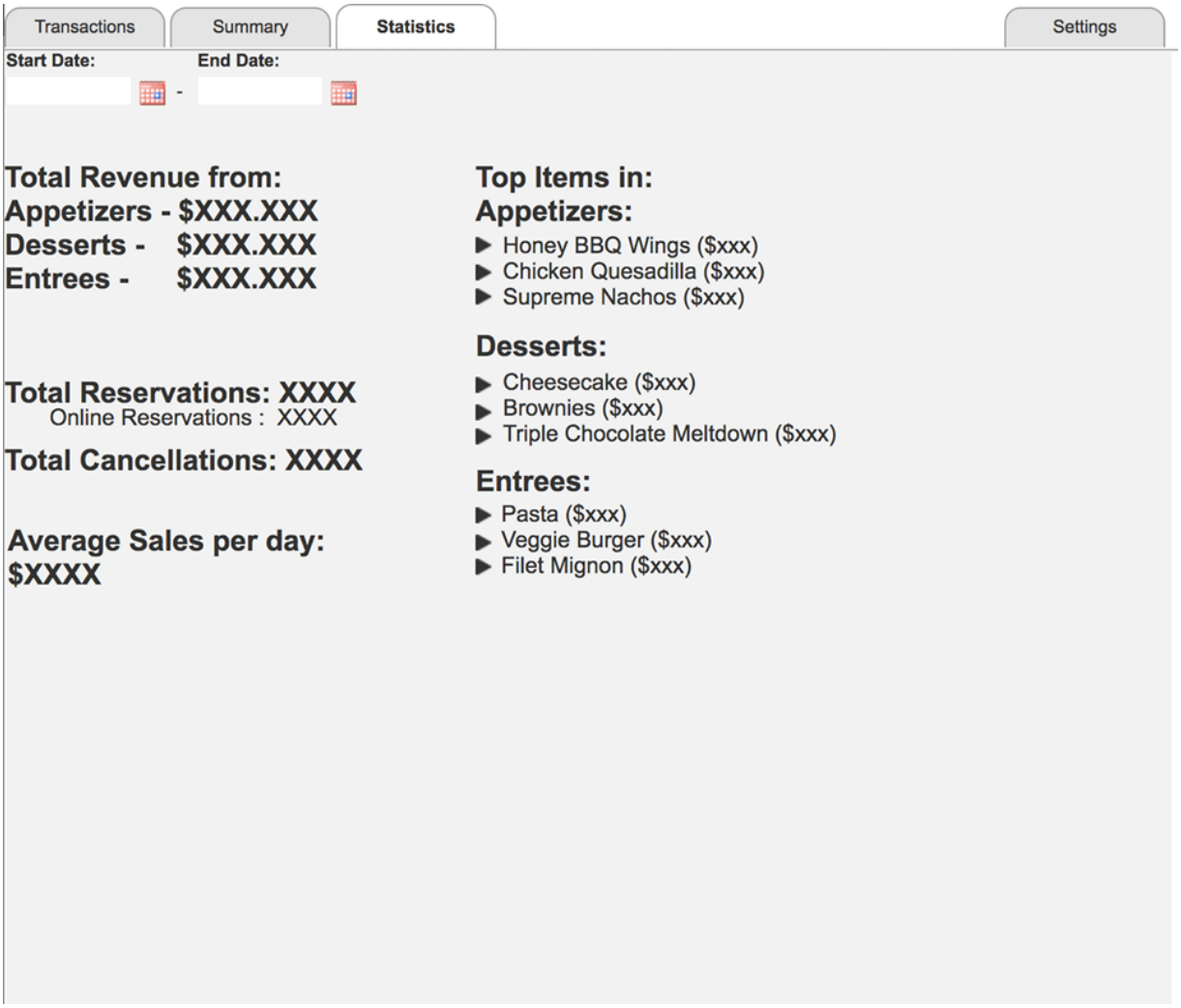
User Interface Specification

Preliminary Design

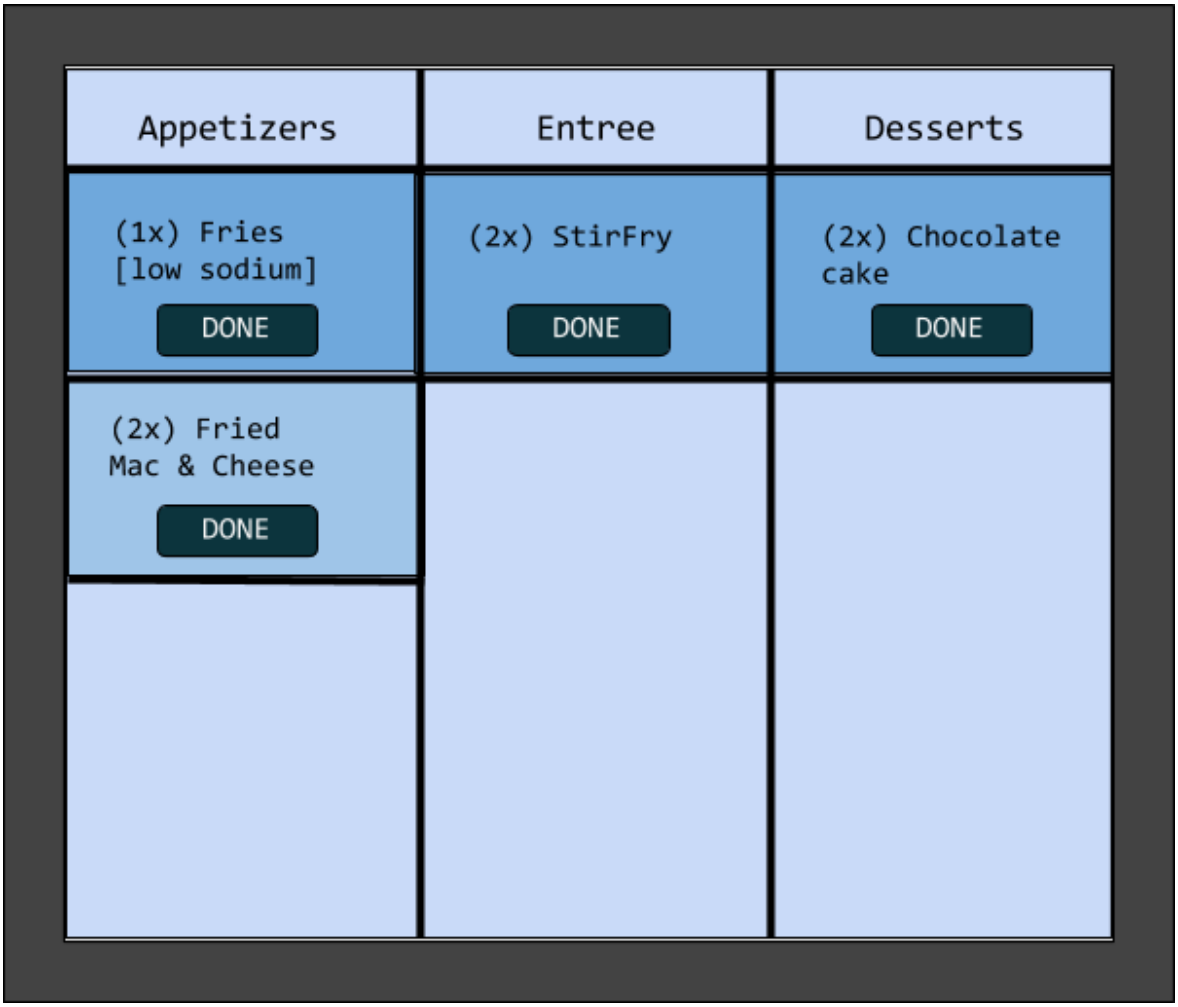
The manager will be able to see all the checks for the given date, and be able to export them if he needs to. From this page the manager can navigate to the summary or statistics tab to view more information.



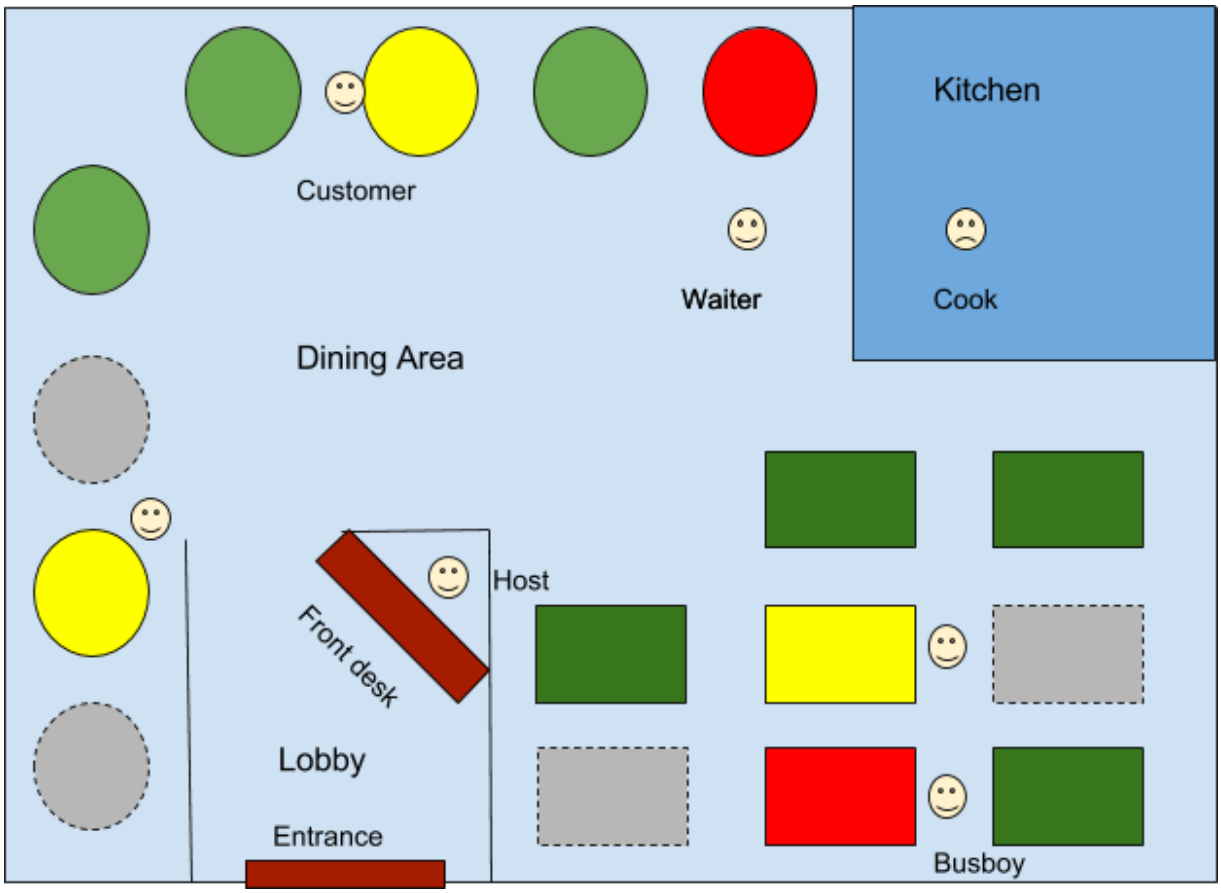

Transactions
Summary
Statistics
Settings
Start Date:
End Date:
-
Total sales : $XXXX.XXXX
Total profit: $XXXX.XXXX
Sales Graph
Total checks: XXX
Total items sold: XXX
Most Popular Item:
Appetizers: Calamari
Desserts: Tiramisu
Entrees: Chicken Parmesan


Transactions
Summary
Statistics
Settings
Start Date:
End Date:
-
Total Revenue from:
Appetizers - $XXX.XXX
Desserts - $XXX.XXX
Entrees - $XXX.XXX
Total Reservations: XXXX
Online Reservations : XXXX
Total Cancellations: XXXX
Average Sales per day:
$XXXX
Top Items in:
Appetizers:
Honey BBQ Wings ($xxx)
Chicken Quesadilla ($xxx)
Supreme Nachos ($xxx)
Desserts:
Cheesecake ($xxx)
Brownies ($xxx)
Triple Chocolate Meltdown ($xxx)
Entrees:
Pasta ($xxx)
Veggie Burger ($xxx)
Filet Mignon ($xxx)

The Chef will be able to see which items on the list he/she should get done first. The chef could also modify the queue if needed.



Floor Plan



Legend:

Available Table

Occupied Table

Dirty Table

Reserved Table

User Effort Estimation

Scenario 1: Manage Floor Status

1. User logs in and can view the whole floor plan
    a. If user wants more information on a specific table, they can tap on it
    b. Tap on back after user is done with looking at information

Estimated Total Clicks: 2-4

Scenario 2: Manage Order Queue

1. User logs into system
2. Waiter navigates around categories to place customer orders
3. Chef navigates order around based on priority
4. Chef hits done when food is ready
5. Waiter confirms notification through the device

Estimated total clicks 5-15

Scenario 3: Manage Archive/Statistics

1. User logs into system
2. User navigates to archive
    a. If user wants to view transaction tab they just click once on it
    b. If user wants to view summary tab they just click once on it
    c. If user wants to view statistics tab they just click once on it
3. User goes back/logs out of system

Estimated total clicks: 3-6

Domain Analysis

Domain Model

Concept Definitions:

UC- 8: Manage Floor Status, UC-9: Update Table Status, UC-7: Send Table Notification, UC-6: Make Reservation

Responsibility Description	Type	Concept Name
Authenticates and sets permissions for the user logging in.	D	Authenticator
Container for user's authentication data, such as pass-code, timestamp, permissions, etc.	K	Key
Collects data about reservations and utilizes it to suggest reservation intervals for table sizes	D	ReservationTableManager
Actively collects table request data for a given table size cluster.	K	TableLoadStatus
Groups tables by clusters and manages the reservation interval and number of reservation to walkin tables for each table cluster	D	TableClusterManager
Pair up tables to make them a bigger size based on availability	D	TableBuddyManager
Manages status of table (clean, reserved, in-use, free) and takes state changes	D	TableStatusManager
Notifies relevant personnel of changes in table status	D	TableStatusNotifier
Displays total floor plan	D	FloorPlanPage
Displays reservation options	D	ReservationPage

request the busboy what table to clean	K	CleanRequest
host requests a table to be marked for walkin	K	WalkinRequest
waiter sends when table is done and needs to be cleaned	K	SetCleanedRequest
send to busboy to notify which table to clean	K	CleanNotify
send to waiter to notify table is ready	K	ReadyNotify

UC-16: Manage Archive/Statistics, UC-2: Split Checks

Responsibility Description	Type	Concept Name
Authenticates and sets permissions for the user logging in.	D	Authenticator
Logs all daily sales of the restaurant	D	Logger
Extracts and calculates statistics from several data sets such as meals, sales, and profit for the requested time period	K	SystemStats
Container for past inventory count, transactions, and statistics	K	Archive
Updates and displays system time	D	UpdateTime
Divides bill among customers	D	CheckSplitter

UC-17: Manage Order Queue, UC-1: Place Order, UC-3: Send Food Notification, UC-5: Check Food Status, UC-19: Check Table Meal Status

Responsibility Description	Type	Concept Name
Customer order is put into the food queue	D	PlaceOrder
Container for customer orders. Acts as controller and marks food as out for cooking and keeps track of food until it is fully cooked.	K	OrderQueue
Notification for when food is ready	D	FoodReady
Divides meals into stages and moves meals between stages based on time since order.	D	StageManager

Manages queue priority based on the time the order came in, the meal type, and the average time to make the meal in the order.	D	PriorityCalculator
Queries the order queue system to check if the food is ready or not.	D	FoodChecker
Relevant participants can track a table's meal stage.	K	MealStageTracker
a request for food to be marked as done	K	FoodDone
polls on behalf of the waiter to ask when parts of his order are ready (one per waiter order)	D	FoodPollRequest
sent to waiter when a "meal stage" of his order is ready	K	OrderReady

UC-14: Notify Manager of Inventory Status, UC-18: Track Raw Materials, UC-20: Suggest Use of Materials, UC-21: Generate Menu Suggestions

Responsibility Description	Type	Concept Name
Contains the state of the restaurant's current inventory. Maintains queues in order of when they should be used (by expiration date)	K	InventoryContainer
gets sent of list of ingredients and subtracts them from the inventory and forms an items list for the order which contains which "batches" of items to use	D	InventoryTracker
Contains the mapping of all meals to their respective ingredients. Processes meal requests from the chef.	D	IngredientTable
Keeps track of ingredient popularity.	K	IngredientPopularity
Keeps track of meal popularity.	K	MealPopularity
Based on meal popularity, ingredient popularity, meal prices and ingredient prices, it suggests a meal plan for the next week.	D	MealSuggester

Sent by the chef containing the meal to be made by him	K	MealRequest
Contains the list of items to be used to make the meal and which "batch" of them to use.	K	ItemsList
Sent to manager when the low threshold of an inventory item has been reached.	K	ItemLowNotifcation

When considering all the use cases we came up with we avoided the domain modeling concepts for some of them because they weren't core features of the project.

Association Definitions:

Floor Plan associations:

Concept pair	Association description	Association name
FloorPlanPage↔ TableStatusManager	TableStatusManager passes the status of all tables so that FloorPlanPage can display a visual to the host	send floor plan info
ReservationPage↔ TableStatusManager	TableStatusManager passes reservation table info for ReservationPage to display to customer	send reservation info
TableStatusManager↔ TableClusterManager	TableClusterManager passes the types and size of the tables and the interval for reserve table to TableStatusManager	send cluster layout
TableStatusManager↔	TableStatusManager requests help from	request merge tables

TableBuddyManger	TableBuddyManager to see if it's possible to merge tables to satisfy a request that TableStatusManager cannot currently satisfy	
TableClusterManager↔ TableBuddyManager	TableBuddyManager sends a request to TableClusterManager to merge tables to form a new one and move around clusters	merge tables
TableStatusManager↔ ReservationTableManager	TableStatusManager passes data on the number of reserved tables and stats so that ReservationTableManager can determine the correct interval tables can be reserved in	send reservation data
TableStatusManager↔ TableLoadStatus	TableStatusManager passes data about the request "load" for tables so that TableLoadStatus can determine the future number of required reserved vs walkin tables.	send table load data
TableLoadStatus↔ TableClusterManager	TableLoadStatus sends data to TableClusterManager so it can calculate the number of reserved vs walkin tables in each cluster	send walk/reservation data
ReservationTableManager ↔ TableClusterManager	ReservationTableManager sends the suggested reservation request interval to TableClusterManager	send reservation interval
TableStatusManager↔ StatusNotifier	TableStatusManager sends the required notification to StatusNotify so it can be passed to the relevant personnel	send request to notify
TableStatusNotifier↔ CleanNotify	TableStatusNotify forms a CleanNotify to be sent out to the busboy	send clean notify
TableStatusNotifier↔ ReadyNotify	TableStatusNotify forms a ReadyNotify to be send to the waiter to alert his/her table is ready	send ready notify
CleanRequest↔ TableStatusManager	CleanRequest tells TableStatusManager to mark table as "needs cleaning"	send clean request
WalkinRequest↔	WalkInRequest tells TableStatusManager	send walkin request

TableManager	to find available walkin table and mark it as in use	
SetCleanedRequest↔ TableStatusManager	SetCleanedRequest sent by busboy to mark a table as available after it has been cleaned	send setcleaned request

Order Management Associations:

Concept pair	Association description	Association name
PlaceOrder ↔ OrderQueue	PlaceOrder retrieves the orders from the waiter which get pushed into the order queue.	Order request
StageManager ↔ OrderQueue	StageManager provides the data about the “meal stage” of each table so that OrderQueue can perform fairly distribute orders so that no table gets all of eats meal "stages" before another.	Table Staging
PriorityCalculator ↔ OrderQueue	PriorityCalculator utilizes data to calculate the meals priority in the queue which utilized by QrderQueue to prioritize orders.	Priority Managing
StageManager ↔ MealStageTracker	StageManager provides the meal stage status for each table to MealStageTracker which provides an interface for viewers to know what meal stage a table is at.	Stage viewing
FoodReady↔ OrderQueue	Food that has been pushed out of the order queue is marked cooking and waits to be marked as done. FoodDone notifies the queue to mark the food as done cooking.	send food done
FoodPollRequest↔ Orderqueue	The FoodPollRequest (per waiter) knows all the stages of a waiter's order and continuously polls to ask if food is ready for food related to a given stage from the OrderQueue.	poll food done
FoodPollRequest↔ OrderReady	FoodPollRequest sends an OrderReady notification to the waiter when all the	send order ready

	food related to one of his order's stages is ready.	
PlaceOrder↔ FoodPollRequest	FoodPollRequest gets the PlaceOrder request so they can poll the order queue for the food in the proper order.	add order

Inventory Associations:

Concept pair	Association description	Association name
IngredientTable↔InventoryTracker	IngredientTable sends a list of ingredients for InventoryTracker to subtract from the inventory	inventory request
InventoryTracker↔InventoryContainer	InventoryTracker tells InventoryContainer what items to subtract out. InventoryTracker pulls the first items from each item's inventory queue and forms a list.	subtract and fetch items
InventoryTracker↔Items List	InventoryTracker forms a list from the items it pulled from the InventoryContainer and sends the list with which batch of items the chef should use	send items list
MealRequest↔Ingredient Table	The chef sends a MealRequest for the meal he is making to the IngredientTable who maps the meal item to its ingredients	send meal request
InventoryTracker↔ItemLowNotifcation	When InventoryTracker sees an item is below its "low" threshold it sends a ItemLowNotification to the manager saying which items to buy more of.	notify low items
IngredientTable↔MealPopularity	MealPopularity keeps a count of which items are requested to IngredientTable to keep track of most popular items.	track meal popularity
InventoryTracker↔IngredientPopularity	IngredientPopularity tracks which items are requested to InventoryTracker and keeps track of most popular ingredients.	track ingredient popularity
MealSuggester↔MealPopularity	MealSuggester checks MealPopularity to see which meals are popular to perform	Reads meal data

	stats and figure out an optimal meal plan for the next week.	
MealSuggester↔IngredientPopularit	MealSuggester checks MealPopularity to see which ingredient are popular to form an optimal meal plan for the next week.	Read ingredients data

Archiving/Logging Associations:

Concept pair	Association description	Association name
CheckSplitter↔Archive	As checks are split and paid the checks are then archived by Archive database.	archiving
CheckSplitter↔Logger	The splitting of checks and the payments made are logged to create a daily log of payments	logging
Archive↔SystemStats	Utilizes data from Archive to form stats for a request period	statistics forming
UpdateTime↔Logger	Logger uses UpdateTime to get consistent time stamps	consistent logging
UpdateTime↔Archive	Archive uses UpdateTime to get consistent time stamps for archiving	consistent archiving

Attribute Definitions:

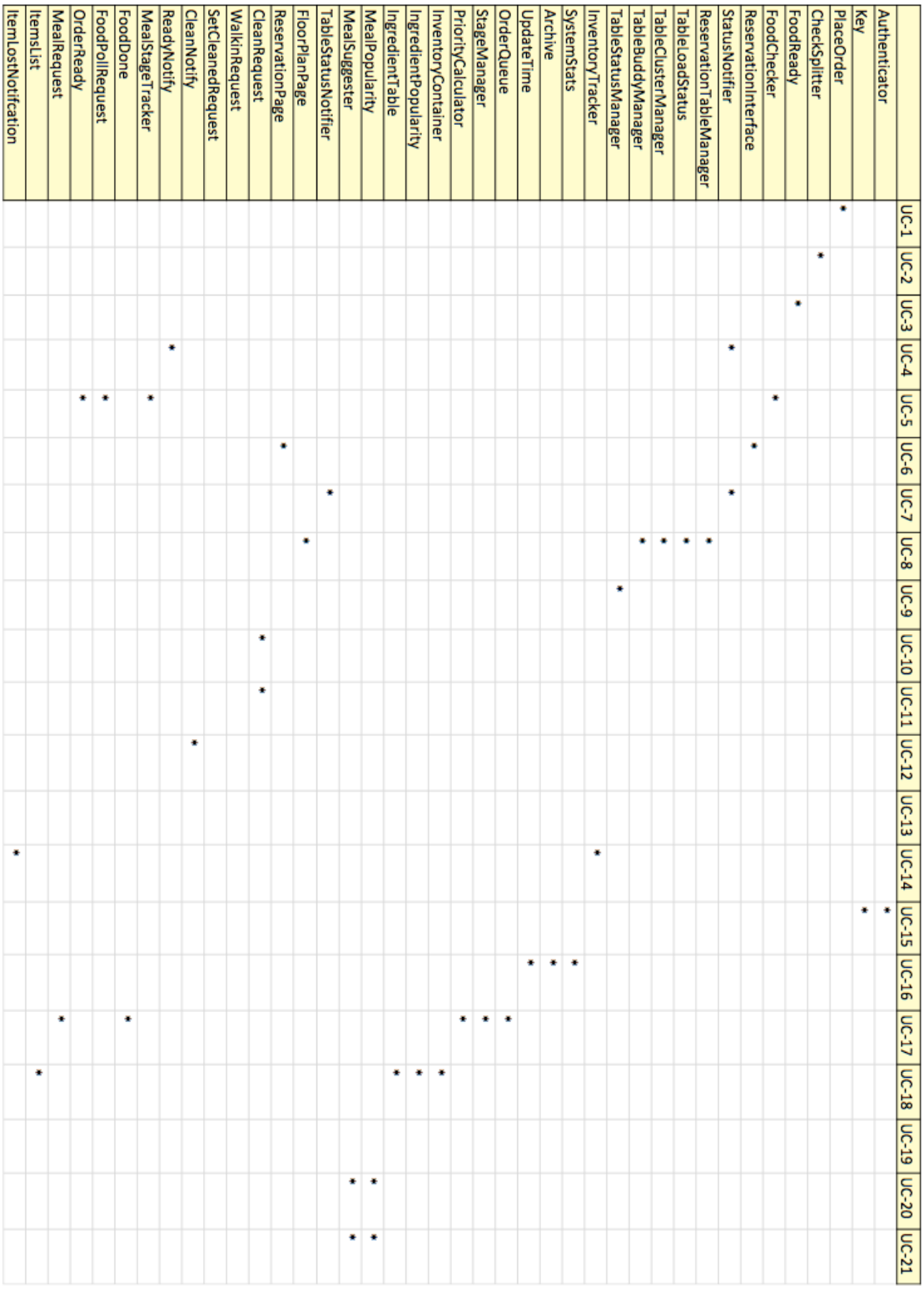
Concept	Attributes	Attribute Descriptions
Authenticator	-Role	Manager, Waiter, Chef, Host, etc.
ReservationTableManager	-ReservationInterval	Time between each reservation
TableLoadStatus	-TableSizeRequests	Number of requests for the respective table size
TableClusterManager	-ReservationNum -WalkInNum	-Number of reservations vs walk ins for a specific table

TableBuddyManager	-NewTableSize	-Adds tables of smaller size together to create a new table size
TableStatusManager	-TableStatus	-Holds the status of the table, such as clean, reserve, in-use, etc.
WalkinRequest	-TableID -CustomerID	-Holds the ID of the table that will be marked as walk in and the customer occupyng it
CleanNotify	-TableDirty	-Is true if the respective table is dirty, and if true sends a notification to the busboy
CleanRequest	-TableID	-Holds the ID of the table that the busboy needs to clean.
SetCleanedRequest	-TableID	-Holds the ID of the table that the busboy has cleaned
ReadyNotify	-TableID	-Holds the ID of the table that the waiter needs to serve
Logger	-ReceiptID	-Unique identification number for each receipt generated
SystemStats	-MealsSold -Sales -Profit	-Total numbers of meals sold in a given time period -Total number of sales in a given time period -Total profit in a given time period
Archive	-InventoryCount	-Keeps all the counts in the inventory per day
UpdateTime	-SystemUpdateTime	-Time the system was last updated
CheckSplitter	-NumberOfCustomers	-Number of customers to split the check across
PlaceOrder	-CustomerOrder	-List of items customer ordered

FoodPollRequest	-OrderID -OrderStatus	-Uses orderID to check if the OrderStatus
OrderQueue	-OrderQueue	-Queue of all the orders coming in to the kitchen
FoodReady	-IsFoodReady	-Holds the value received from FoodChecker about the status of the food
MealRequest	-MealName	-Container for the recipe of the meal requested
OrderReady	-IsOrderReady	-Checks if food is ready when it is requested by FoodPollRequest
StageManager	-TimeSinceOrder -NumOfMealsPerStage	-How long it has been since the order was placed -Number of meals in each stage
ItemLowNotification	-NotifyLow	-Alerts when certain items are low
PriorityCalculator	-MealType -AverageTime	-Type of meal (appetizers, entrees, desserts) -Average time to make meal based on previous data
ItemsList	-ShowList	-Shows what items will be needed for the meal
FoodChecker	-FoodStatus -OrderID	-Status of the order based on the order identification
MealStageTracker	-MealStage -TableID	-The stage of the meal(appetizers,entrees, dessert) the respective table is
InventoryContainer	-ListIngredients -Prices -ExpirationDates	-Keeps a tabular format of the list of ingredients -List of prices associated with each ingredient -The expiration date of the

		ingredients
InventoryTracker	-LowThreshold	-The low thresholds of each ingredient
IngredientPopularity	-TopIngredients	-List of each ingredient used and how often it was used and sorted from high to low
IngredientTable	-RecipeBook	-List of ingredients used for each recipe
MealPopularity	-TopMeals	-List of how many times a meal was ordered and sorted from high to low
MealSuggester	-SuggestedMeals	-List of meals being suggested based on collected data

Traceability Matrix

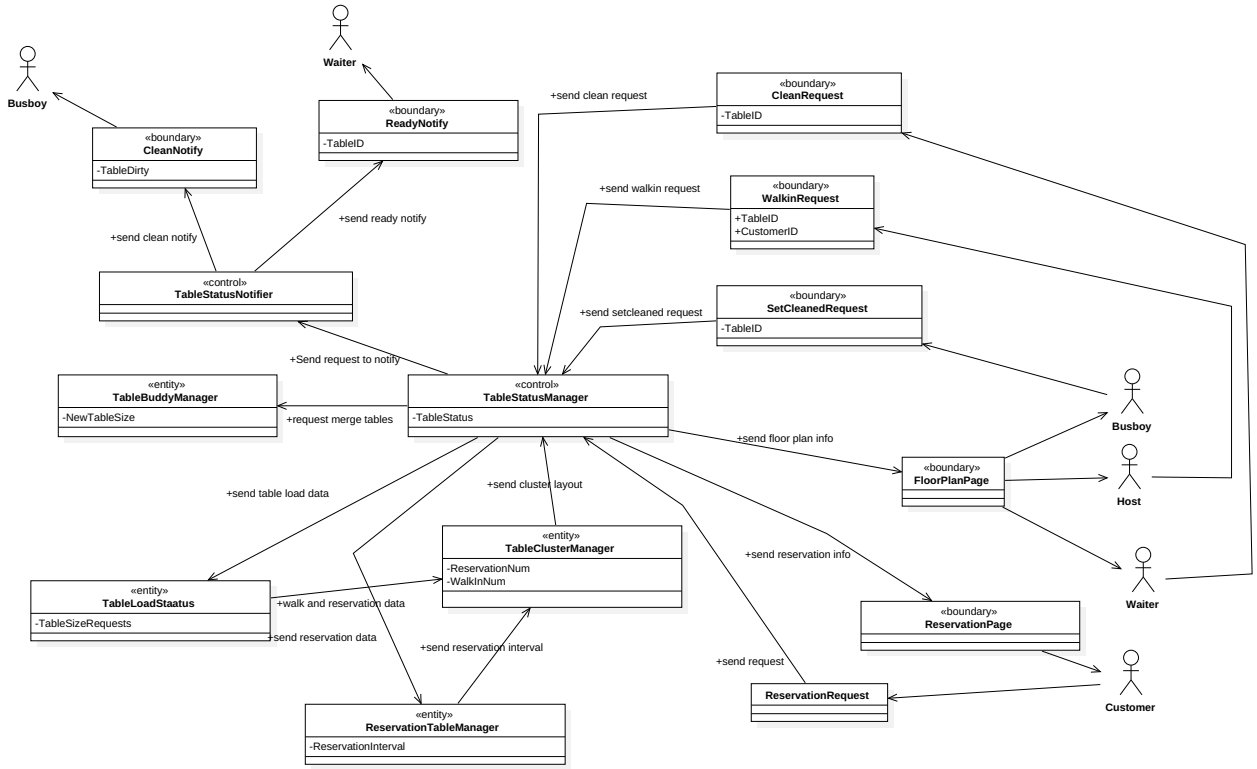


The traceability matrix above shows how each domain concept is connected to a use case. The domain concept might partly implement a use case or a domain concept might implement multiple use cases completely.

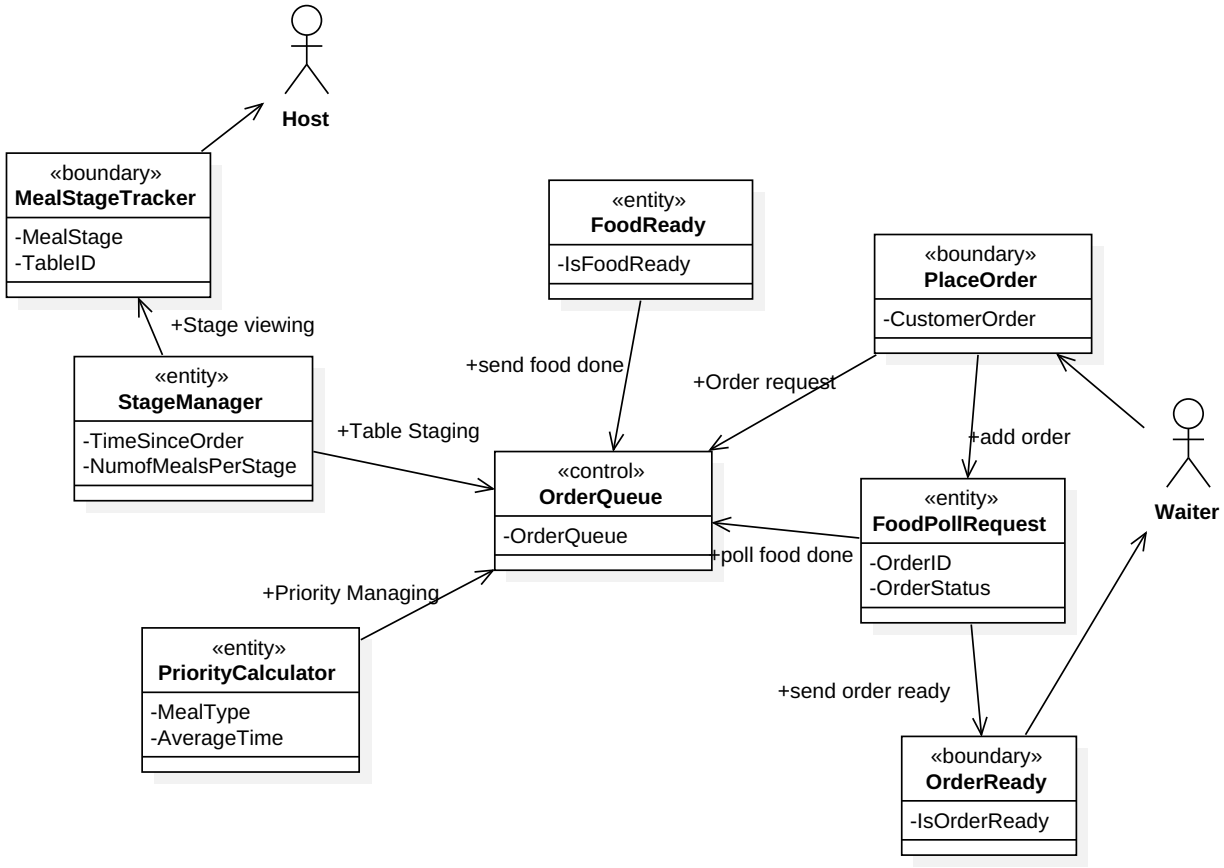
Domain Model Diagram

Below is the Domain models for the chosen use cases. It displays the interaction each respective subsystem

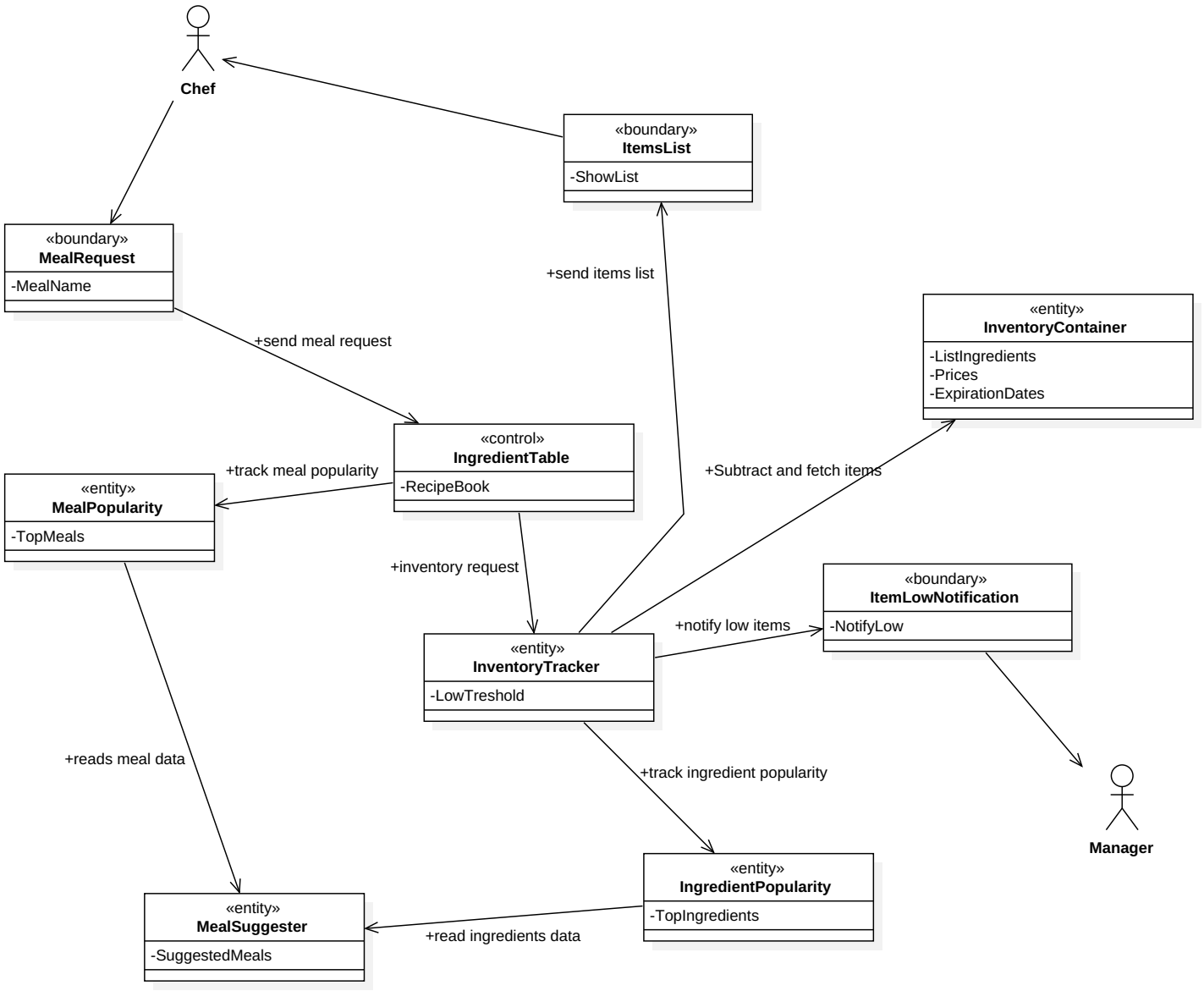
::Domain Model for UC- 8: Manage Floor Status, UC-9: Update Table Status, UC-7: Send Table Notification, UC-6: Make Reservation



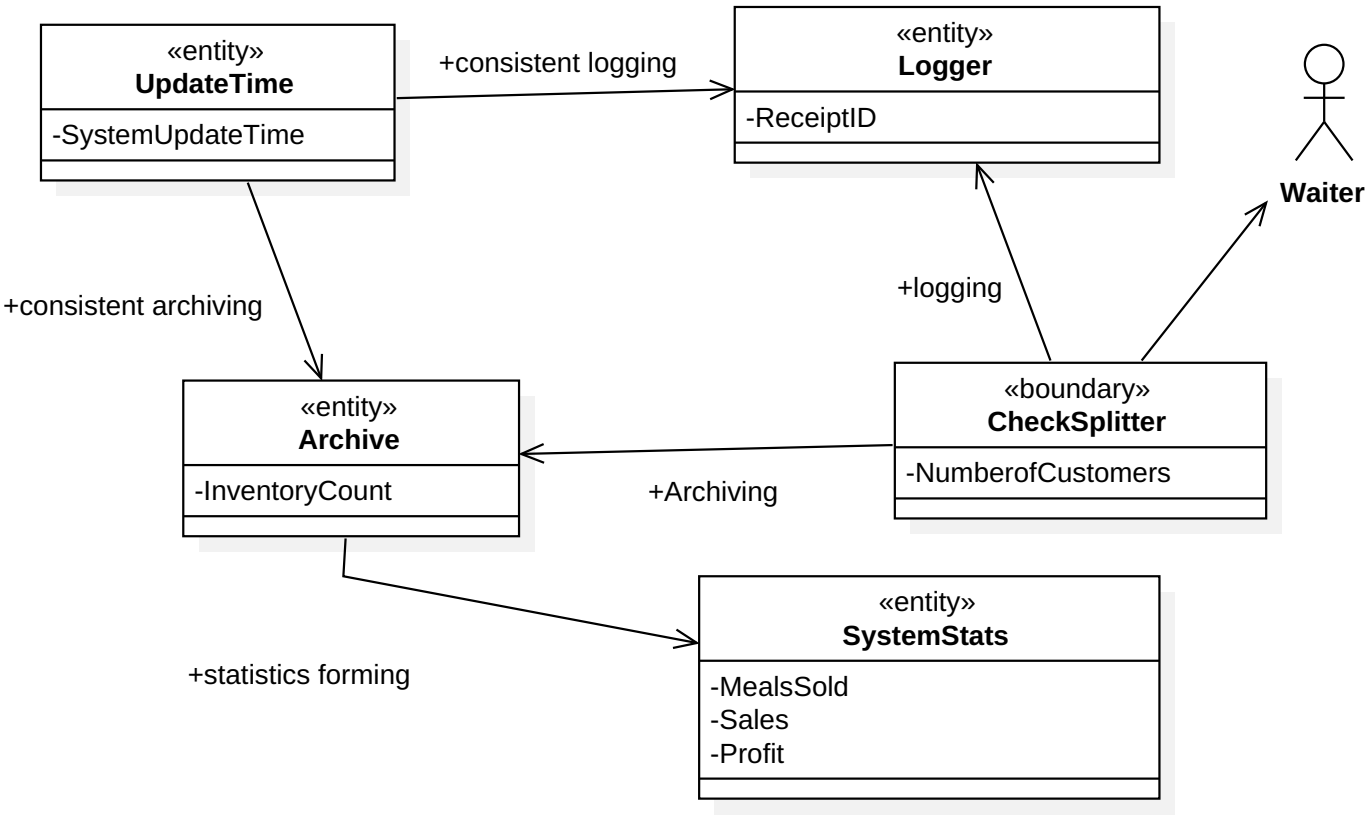
::Domain Model for UC-17: Manage Order Queue, UC-1: Place Order, UC-3: Send Food Notification, UC-5: Check Food Status, UC-19: Check Table Meal Status



::Domain Model for UC-14: Notify Manager of Inventory Status, UC-18: Track Raw Materials, UC-20: Suggest Use of Materials, UC-21: Generate Menu Suggestions



::Domain model for UC-16: Manage Archive/Statistics, UC-2: Split Checks



System Operation Contracts

Operation	**Manage Floor Status**
Preconditions	- User is authenticated by the authenticator using his key
Postconditions	- FloorPlanVisualInterface shows the floor plan for all tables and shows the status of each table

Operation	**Manage Order Queue**
Preconditions	- Customer's valid order is placed through PlaceOrder - OrderQueue and PriorityCalculator are initialized (MealType and AverageTime are either calculated from previous collected data or initial input).
Postconditions	- Waiter is notified that a "meal stage" of an order is ready through OrderReady - Waiter is able to track order status through FoodPollRequest - OrderQueue manages priority and cooking time for each order based on calculations by PriorityCalculator

Operation	**Manage Archive/Statistics**
Preconditions	- Manager is authenticated by the authenticator using his key - Logger has recorded the daily sales of the restaurant(at least one sale) - Archive contains past data of the restaurant
Postconditions	- SystemStats analyzes the data and displays statistics.

Mathematical Model

Suggesting Meal Plan

The Meal Suggester will determine the popularity of an item based on the popularity of the item as well as the the total profit gained from the sale of that particular item. This is done by assigning levels of popularity to each item based on how long the item has been popular. Within each level the total cost and revenue of the meal item is calculated to determine the profit margin. The cost aspect of this is determined by keeping track of the total ingredients used to make a particular meal. In the end, the system will suggest items within each level of popularity based on their respective profits.

Meal Suggester Attributes

- Popularity Level
    - Level: Low, Medium, High, Exclusive
- Meal Popularity
    - Average Meals/Week - Meals/Day of last 7 days divided by DayCount
    - Average Meals/Month
    - Average Meals/Quarter

Popular Durations	**Popularity Level**
Week, Month, Quarter	**Exclusive**
Week, Month	**Medium**
Month, Quarter	**High**
Week, Quarter	**Low**
Week	**Low**
Nothing	**Low**

```
FOR EACH : POPULARITY LEVEL
{
	FOR EACH : MEAL
	{
		COST = (NUM_SOLD) * (NUM_INGREDIENTS * INGREDIENT_PRICES)
		REVENUE = (NUM_SOLD) * (MEAL_PRICE)
		PROFIT = REVENUE - COST
	}
	SORT(MEAL by PROFIT, DESCENDING)
}
```

Manage Order Queue

Node Priority:

The Order Queue is a priority based system that contains is a queue of nodes. A node groups a set of orders based on the time range they were placed. For instance, orders placed between 9PM and 10PM will belong to a specific node. The time range for each node is not static. Nodes that are created earlier have a higher priority than nodes that are created later.

Stage Priority (Within a node):

Each node consists of three stages; appetizers, entrees, and desserts. When orders are placed, all of the items are divided among the three stages. Each stage has a base priority level with appetizers having the highest priority, entrees with the second highest priority, and desserts with the least priority. If an order only belongs to less than three stages, it will have a higher priority than the base priority value of that stage.

Item Priority (Within a stage):

Within each stage, items are grouped by each table order. The priority of an item will be determined based on the average time it takes to cook the items of a stage for a table order. Individual items within a stage are prioritized in a way such that the average time for each table order remains relatively consistent. Therefore, each table order is given a fair prioritization.

```
NODE_TIME_LENGTH = some value
FOR EACH : ORDER
{
	IF : CURRENT TIME - NODE_TIME_LENGTH >= PREV_NODE_START_TIME
	{
		CURRENT_NODE = NEW NODE(PREV_START_TIME + NODE_TIME_LENGTH)
	}
	NODE = PREV_NODE
	INSERT(ORDER, NODE)

	FOR EACH: ITEM {SET BASE STAGE PRIORITY }
```

```
       IF : NUMBER OF STAGES FOR ORDER < 3
       {
              FOR EACH: ITEM { INCREASE PRIORITY OF ALL STAGES }
        }

UPDATE_PRIORITIES:
       FOR EACH : STAGE
       {
              FOR EACH : ORDER
              {
                     DETERMINE TOTAL TIME FOR ORDER
                     DETERMINE AVERAGE TIME/ITEM FOR ORDER

                     ITEM PRIORITY += RATIO OF TOTAL TIME TO AVG TIME/ITEM
              }
       }

OnItemCompleted:
       UPDATE_PRIORITIES
```

Plan of Work

Tasks			Feb 20	Feb 26	Mar 2	Mar 5	Mar 8	Mar 10	Mar 12	Mar 24	Mar 25	Apr 15	Apr 26	May 1	May 3	May 5
Report # 2	Part 1	Interaction Diagrams														
	Part 2	Class Diagrams														
		System Architecture														
	Part 3	Algorithms														
		User Interface Design														
		Design of Tests														
		Project Management														
First Demo	Product Brochure															
	Demo															
Report #3	Part 1	Revise Report #1														
		Revise Report #2														
		Add summaries/finalize														
	Part 2	Full Report														
	Reflective Essay															
Second Demo																
Project Archive																

Note: The different colors indicate that there is a deadline for submission is between them.

Product Ownership

To make sure that our team is productive and efficient we divided up into 3 groups of two. Since we have completed an initial design of our system together, our next few weeks would be to build prototype of the system. Since we have not completely created detailed algorithms, we will try to build the UI and use basic/naive algorithms according to the concepts in the domain model. We created the tasks by grouping similar use cases in a subsystem and assigning each subsystem to a subteam.

Teams:

Dylan Herman and **Moulindra Muchumari** are responsible for Cleaning, Billing, and Manager subsystems.

Completed Tasks:

- Created preliminary UI design of the Manager's data and statistics view
- Planned out the use cases and domain models for the management subsystem

Current Tasks:

- Design what data points are to be collected from the restaurant and possible statistical measurements that can be analyzed from the data.
- Keep attending weekly meetings

Future Tasks:

- Implement the UI design for the manager's console and make changes accordingly
- Create/list out possible settings that the manager should manage
- Create the split check interface
- Implement the archiving and statistics system

Mit Patel and **Prabhjot Singh** are responsible for the Ordering subsystem.

Completed Tasks:

- Created the UI design for the chef's Order Queue
- Planned out the use cases and domain models for the Order Queue subsystem
- Created the mathematical models for managing order queue

Current Tasks:

- Implement the UI design for the chef's Order Queue
- Keep attending weekly meetings

Future Tasks:

- Create UI design for the waiter's tablet to place orders and receive notifications
- Create a sample menu/ menu categories
- Implement the Order Queue subsystem using the mathematical model and domain analysis

Nill Patel and **Raj Patel** are responsible for Seating and Inventory Management subsystems.

Completed Tasks:

- Created an interactive UI design for the floor plan using different colors for the status
- Keep attending weekly meetings
- Planned out the use cases and domain models for the Floor Plan subsystem
- Created the mathematical models for suggesting meal plans

Current Tasks:

- Implement the floor plan UI (only the non-interactive part)
- Keep attending weekly meetings

Future Tasks:

- Create the reservation interface frontend
- Design the UI for raw materials (ingredients) interface that displays the list of materials and their amounts
- Implement the food suggestion subsystem using the mathematical model and domain analysis

Improvements from Past Projects

In comparison to previous projects, the main intellectual contribution from our team would be the aspect of collecting a lot of data and using that data to determine trends that would help the overall efficiency and profitability of the restaurant. An instance of this can be seen in our inventory system. This system takes into account not only the ingredient usage but also keeps track of which ingredients are popular alongside the meals that are popular and uses them to determine suggestions that maximize profit as well as customer satisfaction. Simply keeping track of meal popularity is not enough since multiple meals may be popular just because they have a popular ingredient. It is important to take this into account so restaurants can make more of the popular meal that maximizes their profit and not have to make all popular meals.

Another instance where our project differs from others is the Order Queue Management system. Ordering meals to be cooked and pushed out to the customer is more complex than simply first-in, first-out ordering. Things that must be taken into account are: how large is a given order and how much time on average does it take to make a meal in that order? What type of meal is it, appetizer, entree, or dessert? What “stage” of the meal is one table on vs another? Simple first-in, first-out might put the table who orders all their meals at once over a table who just order one meal a little later than that table. Both tables should be able to get at least one meal before they get their second. We have taken into account these complexities and have devised mechanisms for ordering the meals in a fair fashion something other projects have not done.

All team members managed the project equally!

References

- "What's is the difference between include and extend in use case diagram?" N.p., n.d. Web. 5 Feb. 2017. <http://stackoverflow.com/questions/1696927/whats-is-the-difference-between-include-and-extend-in-use-case-diagram>.

- Marsic, Ivan. "Software Engineering Project Report." Software Engineering Project Report - Requirements. N.p., n.d. Web. 19 Feb. 2017. <http://www.ece.rutgers.edu/~marsic/Teaching/SE/report1.html>.

- StarUML
  -Program used to create Diagrams

- Axure
  -Program used to create wireframes for the UI