Course Name: Software Engineering
Course Number: 14:332:452
Group: #10

Super Foods

Final Report
Submission date: April 25, 2021

URL: https://github.com/BriannaSolano/superfoods2.0

Group Members:
Robert Kulesa, Justin Chan, William Basanaga, Lindsay Wisner, Guilherme Kenji Silva Horikawa, Saurabh Bansal, Keith Lo, Brianna Solano Aguilar, Rawad Sayah, Jeremy Kim, Gian-Soren Morici

Table of Contents

Section 1: Customer Statement of Requirements

Problem Statement

Customer #1:
One of my favorite things to do on my nights is go out to eat at a restaurant, but lately with the Covid-19 Pandemic going on, it is very hard for me to get out of the house to do so; when I do get to leave the house, I feel extremely uncomfortable. What I want is to have a contactless, efficient delivery service. I hated waiting for my food when I was physically in the restaurant, and being home I don't want to wait any longer than I have to. Even if I did get takeout delivered to me, more than half the time, the food comes cold or is thrown around in the car and the food leaks out to the outside of the container. I don't want my order arriving messy as if it was thrown around.

What could Super Foods do? Super Foods needs to create a delivery system that is very convenient, efficient, effective and above all else safe. I think that this will benefit not only myself, but even those who are compromised and cannot leave their homes in times like this. I would like to have an app where I can make my order without feeling the pressure that I was wasting someone's time like when I am ordering by phone. An app would allow me to write down my order to the finest details and have all of the requests processed without anyone mishearing me over the phone. Even though an app would have the least communication and interaction with members of the restaurant, it would still allow me to convey complicated orders with a low chance of the order being messed up. I would never have to worry about the receiver mishearing my order or forgetting a special request I make for a dish. For this, I think that it would be a good idea to have an app that orders food and has drones to fulfill the deliveries. This would limit my face to face interaction with outside people and would make me feel safer than if I were getting my order directly from a delivery driver. The drones would even make the delivery process much quicker whether I am at home or I decided to opt-in for a curbside pick up. The drones will never need to stop at stop signs or red lights. They would just fly in a straight line and deliver my food while it's still hot. The estimated time of arrival would not change because of a traffic accident or road work. On top of that, I would not even have to worry about the food falling from the seat to the floor if the delivery driver makes a sudden break. Although I think food is very important to a restaurant, the delivery aspect of a restaurant is just as important. Just imagine if the food you ordered always arrived hot and in the condition that the restaurant packed it. That would be amazing. Sometimes I have to wait forever all to get a cold meal; I think if they put in place a system and used current technologies to create the most efficient route of delivery, I would definitely be a regular customer.

Customer #2:
It's annoying, after a long day, to go drive out to my favorite restaurant, and order my favorite meal. Just to find out after some time that it can't be made correctly because the restaurant ran out of stock of a key ingredient.

What could Super Foods do? It would be nice if I could immediately be warned that an ingredient is out of stock when I click on an item that contains it, that way I can choose to order a different item. Better yet, if I see the ingredients that are needed to make my dish start to go down, I can set aside what I am doing to place an order immediately before the ingredients run out. This level of transparency gives me the power to weigh the priorities of the tasks that I need to do. If someone told me that if I placed an order within the next 5 minutes or else I wouldn't be able to order it later, I would.

Customer #3:
I enjoy going out to restaurants often, but I don't like not knowing whether what I'm eating is healthy or not. I have two young children with severe food allergies making the selection of restaurants even slimmer. Fast food and TV dinners are not what I want to be feeding my kids. Often in restaurants we enjoy high quality meals, but dishes are often overloaded with calories, salt or sugar. It would be nice if I could see the health data, especially calorie values, of the foods being offered. That way, I can make the decision to order a healthier item if I'm feeling health conscious.

What could Super Foods do? On the menu, Super Foods can implement a program to display the nutritional information of an item when selected by the customer. Indicators such as red font can indicate if there are large amounts of calories. A button will allow the customer to toggle the ability to view nutritional values of items on the menu. Superfoods can also keep track of the total calorie count of the customer's entire order, assuming the button is toggled on.

Employee:
Delivering for a company has its ups and downs. Sometimes I get tips on top of each delivery which is nice and all, but sometimes I feel like it's just not worth leaving the restaurant just to deliver a small dinner four blocks away. I have to take out my car, get stuck in traffic, waste ten minutes of my life only to receive complaints that I'm delivering cold food.

What could Super Foods do? Super Foods could make me a way to deliver these orders without having me drive to those places. They could also make me something that would allow me to not get stuck in traffic. I am thinking that a drone style delivery system would be ideal. This way I could just dispatch a drone to those time wasteful jobs and save my energy for longer hauls, or catering orders.

Manager:
Since I have been with this company, I have noticed that it is extremely difficult to keep track of the inventory we have coming in and out of the restaurant. The way that we take inventory now is that every night, after we close, we must count all of our ingredients, and that is what is going to determine what we need to order more of. This becomes very time consuming to all of those involved, and I think the time we spend counting inventory can be used in many other areas of the restaurant, not in places that are unnecessary. Manually doing the inventory also becomes very stressful because you have to make sure that your counting is exact, and it is very hard to

keep track when we can have a lot of the same item. It also becomes an issue during inventory when we have to throw away a product due to it being spoiled or not selling. Sometimes without realizing, we order too much of the same thing and end up throwing it out.

What can Super Foods do? I think Super Foods could build an inventory system for our restaurant. I think it should have some features like automatic inventory count, manual override, automatic reordering of products, and overall, I think it just needs to be organized in a good manner. When I say automatic inventory count, I want the inventory to automatically update once a shipment comes in. I think manually updating the inventory every time a product comes in would be very tedious and it would allow a lot of room for error. At the same time, we should be able to manually override the inventory count if we feel necessary. Sometimes when the chefs are in the kitchen, there is food that is dropped and needs to be thrown away before it even reaches the customer. The inventory I also think should be processed in a way that after something is ordered, it is automatically deducted from the inventory count so it has a most up to date count that we can be made aware of. The system should also be able to notify me when a product needs to be ordered again, and it should even order it for the restaurant when it reaches a certain point. This will prevent me from forgetting to order important products vital to the restaurant's production. As far as organization goes, I just want it to be easy to read and organized in a way that places all of the products into categories whether it be freezer and refrigerated foods, perishables, or even basic supplies. This would be far easier to navigate through. In the database that Super Foods creates, I think it makes most sense for me to be able to manually enter the products we have once, and from there on out it would save it. This will let me know exactly what I'm looking for based off of what I decided to name the new items. In this same database, you can add specific details about the food that will be portrayed to customers, such as the calorie count. This could go hand in hand with whether a product is out of stock or not.

Owner:
Restaurants want to provide their customers with a unique experience while also generating revenue. It's typically the same routine for every restaurant, sit, order, wait, eat. I often see adults bored on a weekend continuously debating on something fun to do and ultimately deciding on nothing. As a restaurant owner, I want to provide my customers with a fun experience that can only be given at my restaurant. Our alcoholic beverages don't receive as much attention from our customers as they should, resulting in poor sales in that area.
I want my alcoholic drinks to receive more attention and be the focus of this new experience. Overall I want to give my customers an opportunity to have fun while drinking with friends through games. I want my restaurant to have a fun environment where customers continue to return.

How will Super Foods help? I want SuperFoods to have a drinking game that is both fun, easy, and built from scratch. I want customers to be required to sign in with an account to have access to the game. I believe this would incentivize my customers to create accounts and allow the game to remain unique to the restaurant. If the game were to be accessed only by account, my restaurant would gain a reputation for having an exclusive game. From the application, I want customers to be able to select the game option from the main menu. When selected, my customers would be redirected to a website where they can select to play the drinking game. I want the drinking game to provide customers with a range of “Would you rather” questions. Customers will have the option to choose between two choices till the timer in the game runs out. From there the game randomly picks one of the options given and displays it to the customer with the text “Take a shot”. At any time, customers can quit the game.

Owner:

I love the application Turbo-Yum but as a restaurant owner I feel that it does not reach its full potential. I always have a problem determining how customers feel about my restaurant. I want to know how my new waiter is performing or the new ratings of my new dish from the customers' point of view. Obtaining reviews from customers on specific things requires too much time and work that can be allocated somewhere else. Turbo-Yums doesn’t provide me with a solution for this review problem. I also feel as if Turbo-Yum does not help my restaurant provide a relationship with its guests, such that they don’t feel like guests but rather like regulars to my restaurant. I find it hard for my restaurant to reach out to its customers beyond just food. Not enough customers are creating accounts to the application Turbo-Yums. Rather than going through the trouble of signing in, they would rather check out as a guest. Turbo-Yums does not provide that incentive for customers to take the time and create an account with us. I find that customers come to the restaurant, order something they have not ordered before, enjoy it, but forget what it was when they return back to the restaurant since the app doesn’t store this information. Rather than searching for items, they opt for something they typically ordered. I find that the app has many major features but not everyone should have access to them. Typically, new employees who are not familiar with the app interface may mistakenly access the inventory and change stock. This problem has become frequently at the restaurant and not every employee should have the same employee status and app privileges as others.

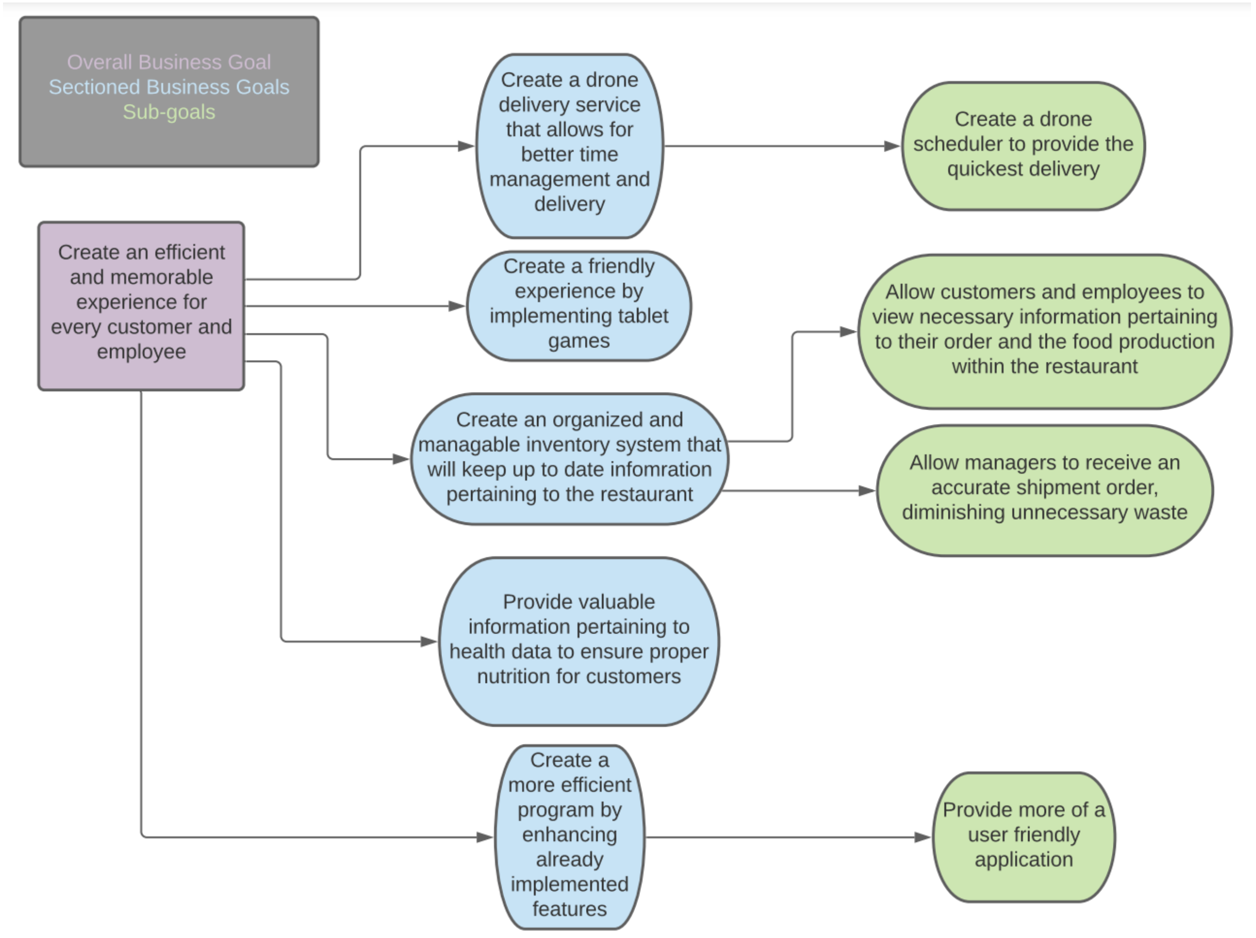
How will Super Foods help? I want Super Foods to enhance Turbo-Yums so that it reaches its full potential. I want the application to provide my customers with a guide on what specific subject my restaurant looks for in a review. I want the application to allow customers to leave reviews on particular foods, on their waiters, the restaurant itself, or any comments they have. I would also want the application to notify waiters if there is a special anniversary day for a guest in the restaurant. Making them feel that my restaurant cares for its customers. I also want Super Foods to provide incentives for my customers to create an account and continuously sign in rather than continuing as a guest. Super Foods should provide a reward system for customers who create an account and sign in. Earning coupons after spending a certain amount or eating at the restaurant a certain number of times. These rewards would be tied into the account. Through the account creation, account holders should have the ability to review past orders from the restaurant. This would incentivize customers even more to create an account, so they will not forget that dish they liked the previous visit. I want Super Foods to expand on the portal and give different status to my employees. I want to have a diverse selection of status to give my employees such as waiter, bartender, chief, and manager. I want these statuses to have privileges right in the app for my employees. Such as chiefs and managers can edit inventory while waiters can only view. This would give my employees a better environment to navigate. I want Super Foods to be an extension of the applicationTurboYums. My customers and employees are already used to the TurboYums interface and I don’t want to give them a new tool that they have to learn. I still want to retain some of TurboYums key features like app transactions and menu view but with new options in place.

Section 2: Glossary of Terms

- **Calories -** Health data specific to the the product
- **Customer -** A person or organization that buys goods and services from a store or business. In this report, the customer is the one who is utilizing the application.
- **Customer Account -** An account that a customer has. The information is given by the customer, and the database is updated with their information. This information is then called back for rewards and saved information for orders.
- **Delivery -** A process for the customer to obtain their food. Delivery is done with Drones or Human driving to the location.
- **Drop Down Menu -** Customers and employees can use this application feature which gives them options to relocate them to different sections within the application
- **Guest Account -** A temporary account that stores information about the customer and their order. The data is stored for precautionary purposes, but the customer can not receive any rewards nor save their information in the database.
- **Inventory -** A list of available items that are able to be used, such as goods and products that are currency in stock on restaurant property
- **Menu -** A location with all the information about the food items sold from the restaurant. Contains information like ingredients, calories, and price.
- **Manager -** A person or organization that is in charge of the restaurant. They spearhead ordering new ingredients for the restaurant and making sure the restaurant runs smoothly.
- **Order/Delivery Queue -** A list of orders that are placed in first in, first out order (FIFO) so that customers will receive their order in a timely manner.
- **Restaurant Automation -** Makes a restaurant flow more efficiently and more easily. Uses devices preloaded with software that manages several tasks which helps eliminate many of the required taste that are normally done via employees
- **Take-Out -** A process for the customer to obtain their food. Takeout is when the customer comes into the restaurant to pick up the food.
- **User Account -** A private location on a network server for an individual who utilizes the application to store information such as their username, password, etc.

Section 3: System Requirements

3.1: Business Goals


Overall Business Goal
Sectioned Business Goals
Sub-goals
Create an efficient and memorable experience for every customer and employee
Create a drone delivery service that allows for better time management and delivery
Create a drone scheduler to provide the quickest delivery
Create a friendly experience by implementing tablet games
Allow customers and employees to view necessary information pertaining to their order and the food production within the restaurant
Create an organized and managable inventory system that will keep up to date infomration pertaining to the restaurant
Allow managers to receive an accurate shipment order, diminishing unnecessary waste
Provide valuable information pertaining to health data to ensure proper nutrition for customers
Create a more efficient program by enhancing already implemented features
Provide more of a user friendly application

3.2: Enumerated Function Requirements

3.2.1: Drone/Driven Delivery Service

Identifier	**Priority** **(Higher number indicates higher priority)**	**Requirement**
REQ - D1	4	As a customer, I need to be able to input my delivery address so that my order can be delivered to the correct location.
REQ - D2	5	As a manager, I need the system to determine which

		delivery method is optimal to reduce delivery costs. This would mean calculating distance and deciding whether or not drone delivery should be used.
REQ - D3	4	As a manager, I only want the drone operator to view the specs relating to the drone delivery system.
REQ - D4	3	As a drone operator, I need to log battery level and battery swaps in the app when drones leave and return to the store. This ensures the drone successfully delivers food and returns without losing power.
REQ - D5	3	As a drone operator, I should be able to update the battery inventory in the app when a spare battery is fully charged.
REQ - D6	3	As a manager, I need to authorize the drone delivery before the drone goes out for delivery. This would allow me to make sure that all deliveries are being made.

3.2.3: Adult "Would you Rather" Game App

Identifier	**Priority** **(Higher number indicates higher priority)**	**Requirement**
REQ - A1	1	As a customer, I need the ability to view the rules of the game.
REQ - A2	1	As a customer, I need the ability to play the game after reviewing the rules.
REQ - A3	2	At the beginning of every question, the system should display a ready slide for a certain amount of time before the question is displayed.
REQ - A4	4	The system should display the "would you rather" questions for a total of 15 seconds before prompting to the randomly selected answer.
REQ - A5	3	As a customer, I need to request an alternate "would you rather" statement whenever I receive an old question or a question that does not fit my morals.
REQ - A6	5	The system should prompt for the answer of the "would you rather" question for a certain amount of time after displaying the question. This is displayed for a set time then prompted to the ready screen.
REQ - A7	5	The system should allow the user to quit during the

		question prompt and redirect users to the main menu.

3.2.4: Inventory System

Identifier	**Priority** **(Higher number indicates higher priority)**	**Requirement**
REQ - I1	5	As an employee, I need to view and modify inventory count so that the kitchen is aware when a product is under stocked.
REQ - I2	4	As a manager, I want my menu items to be hidden from view if an item is out of stock.
REQ - I3	4	As a manager, every time an item is ordered, I want the inventory to reflect those changes and deduct from the overall inventory count.
REQ - I4	2	As a manager, I need to receive notifications when inventory of an item is out so that I can restock for the following work night.
REQ - I5	4	As a manager, I need the Inventory to be adjusted automatically when a shipment arrives.
REQ - I6	4	As an Employee, I need to view incoming deliveries & their dates so that I can inform my customers accordingly when an item will be back on the menu.
REQ - I7	2	As a manager, I need to cancel or reschedule incoming deliveries in case our stock drastically changes.

3.2.5: Calories/Health Data

Identifier	**Priority** **(Higher number indicates higher priority)**	**Requirement**
REQ - C1	2	As a customer, I need to view a menu with/without caloric or nutritional value so that I can feel confident in what I'm ordering.
REQ - C2	5	As a customer, I want to decide whether or not I want to be aware of the calorie count of each menu item.
REQ - C3	5	As a customer, I need the system to calculate the total amount of calories of my order, allowing me to be aware of how much I’m eating.

REQ - C4	4.5	As a manager, changes to the menu should also be reflected on the caloric values.
REQ - C5	3	As a customer, I need to be able to see dietary restrictions and common food allergies so that I am not taking any health risks.

3.2.6: Menu/Order Systems and Customer Interface

Identifier	**Priority** **(Higher number indicates higher priority)**	**Requirement**
REQ - T1	5	As a user, I need to view and manage my order so that my order is accurate to what I want.
REQ - T2	5	As a manager, I need to edit menu items and their description so that our menu is up to date.
REQ -T3	4	As a user, I need to create an account with my username, email, birthday & password so that I can place an order and collect rewards.
REQ -T4	3	As a manager, I need to be notified when a user fails to sign in after 5 attempts to maintain security.
REQ -T5	5	As a manager, I need to lock placing orders when the restaurant is closed.
REQ - T6	3	As a customer, I need to track my order even when the manager has locked placing orders.
REQ - T7	3	As a manager, the system needs to calculate the current price of the customer’s order accurately.
REQ - T8	3	As a user, I need to schedule a delivery time within the managers’ allocated time frame in case I do not want the food immediately.
REQ - T9	3	As a manager, I need to manage the windows orders are being delivered.
REQ - T10	5	As a user, I need to input my card information to complete the order.
REQ - T11	1	As a user, I need to leave star ratings or comments on orders to share my opinions about the restaurant.
REQ - T12	2	As a manager, I want to reward loyal customers by giving them a reward point for any orders over $20.

REQ - T13	2	As a manager, the system should apply a 10% coupon when a loyal customer collects 10 points as an incentive to order more often.
REQ - T14	1	As a manager, I want to reward points on birthdays so that customers feel valued.
REQ - T15	1	As a user, I need to view my order history in order to leave a review or place a similar order.

3.3: Enumerated Non-Functional Requirements

3.3.1: Drone/Driven Delivery Service

Identifier	Priority (Higher number indicates higher priority)	Requirement
REQ - D7	5	The system should be able to get real time, local weather data that is accurate and taken from a credible source.
REQ - D8	5	The system should provide a seamless and accurate transfer from street addresses to coordinates.
REQ - D9	5	The system should provide accurate order weight data.
REQ - D10	5	The system should provide accurate data on battery status.
REQ - D11	5	The system should decide whether a drone or driver should be sent out based on data such as weather, distance, and battery status.

3.3.3: Adult "Would you Rather" Game App

Identifier	Priority (Higher number indicates higher priority)	Requirement
REQ - A8	1	The system should provide smooth transitions from "would you rather" statements to another.
REQ - A9	5	Users should be able to easily understand the rules of the game.
REQ - A10	3	Users should be able to access the game across various devices.

REQ - A11	5	Users should be able to easily understand information displayed from the game.

3.3.4: Inventory System

Identifier	**Priority** (Higher number indicates higher priority)	**Requirement**
REQ - I8	3	The system should be intuitive and easy to understand/use.
REQ - I9	3	The system should offer a different view depending on whether the user is an employee or customer.
REQ - I10	2	The system should be well organized in a manner that makes most sense to employees -- this would mean categorizing the inventory items.

3.3.5: Calories/Health Data

Identifier	**Priority** (Higher number indicates higher priority)	**Requirement**
REQ - C6	3	The system should not have a terribly noticeable delay in the update of the caloric values.
REQ - C7	5	The system should instantly process and display ingredients and nutritional values to the current menu when added.
REQ - C8	5	The system should provide instantaneous updates to the menu with their caloric value when a new item/ingredient is added by the kitchen staff.
REQ - C9	4	System is compatible with most recent versions of software (ie. iOS 11 or Windows 10).
REQ - C10	5	The caloric and nutritional values should be accurate and taken from a credible source

3.3.6 Menu/Order Systems and Customer Interface

Identifier	**Priority** (Higher number indicates higher priority)	**Requirement**

REQ - T16	4	Order pricing should be accurately calculated and displayed.
REQ - T17	3	System should maintain an accurate record of sales/inventory loss.
REQ - T18	1	System should maintain an accurate record of rewards points.
REQ - T19	2	Menu items should be accurately organized pertaining to their group i.e. appetizers, main plates
REQ - T20	3	Menu Items should hold accurate information pertaining to their ingredients.
REQ - T21	4	Adjustments made to the ingredients and toppings of each item ordered should be accurate.
REQ - T22	5	Each user with a different restaurant occupation should only be able to see information pertaining to their job.

3.4: User Interface Requirements

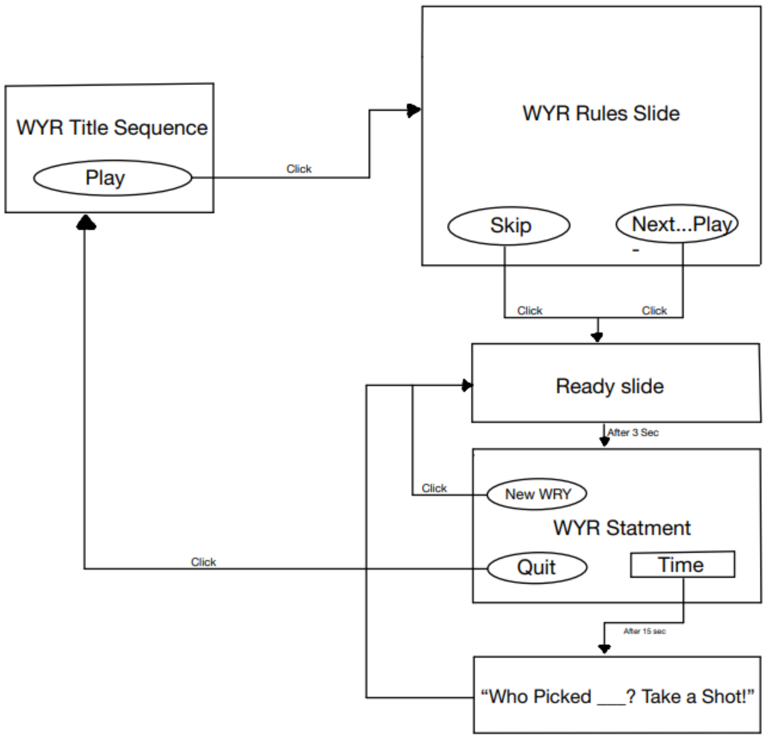
3.4.1: Drone/Driven Delivery Service

Identifier	**Priority** **(Higher number indicates higher priority)**	**Requirement**
REQ - D12	5	The app should display to the drone operator and the manager on duty the status of each drone.
REQ - D13	2	The app should display any issues impeding a drone delivery.

3.4.3: Adult “Would you Rather” Game App

Identifier	**Priority** **(Higher number indicates higher priority)**	**Requirement**
REQ - A12	5	The system should display a button to play in the game’s main menu.
REQ - A13	2	The system should display a button to skip the tutorial or play through the slides containing the rules.
REQ - A14	2	At the beginning of every question the system should

		display a ready slide.
REQ - A15	2	The system should display the would you rather questions for a total of 15 seconds.
REQ - A16	1	The application should display a timer at the bottom to allow users to see the remaining time to answer the question.
REQ - A17	4	The application should display a button during the “would you rather” question to display a new question.
REQ - A18	5	The application should display a randomly selected answer to the “would you rather question” in the middle of the user’s screen.
REQ - A19	5	The application should display a quit button during the question display.



3.4.4: Inventory System

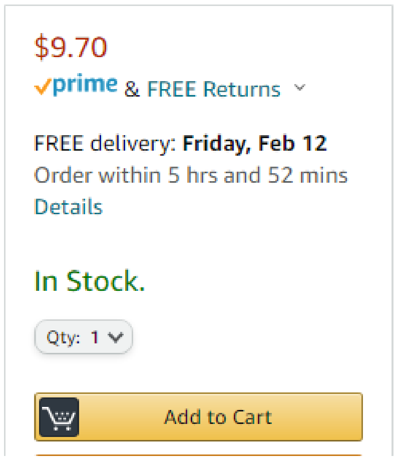
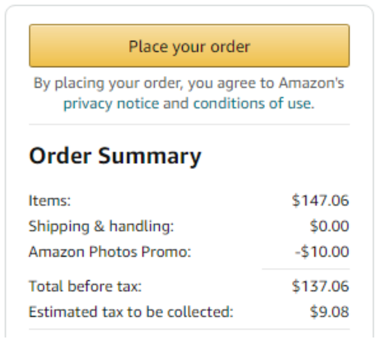
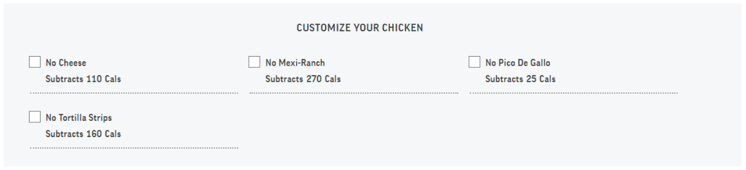
Identifier	Priority (Higher number indicates higher priority)	Requirement
REQ - I11	2	Display items that are out of stock and notify the customer on the menu.
REQ - I12	3	Inventory system should be split into categories of items (hot ingredients, cold ingredients, drinks, etc)
REQ - I13	2	Employees should be able to see the status of all items stock

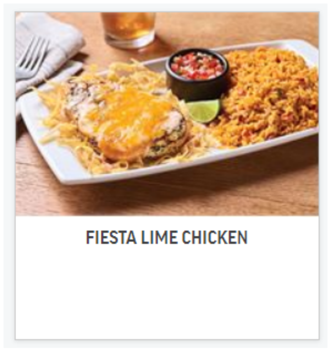
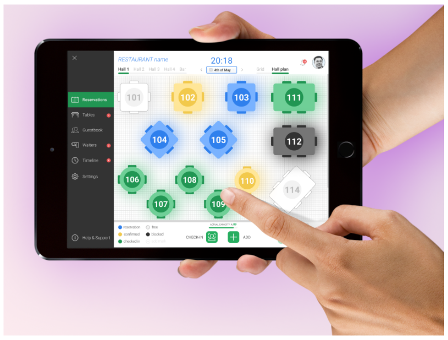
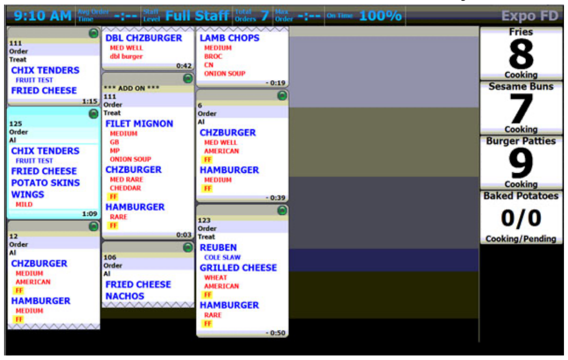
3.4.5: Calories/Health Data

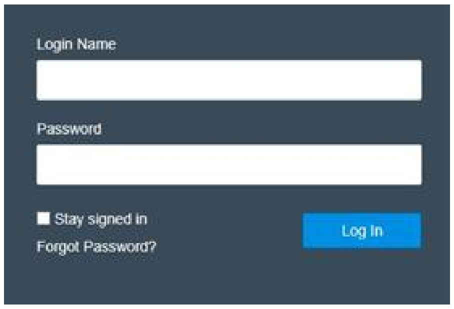
Identifier	Priority (Higher number indicates higher priority)	Requirement
REQ - C11	3	Display for calories should be easily readable.
REQ - C12	5	Ingredients and calorie values should be from a credible source and noted within the app.
REQ - C13	3	Ingredients/nutritional values should be displayed to their respective images.
REQ - C14	2	An option to hide detailed nutritional values should be available.

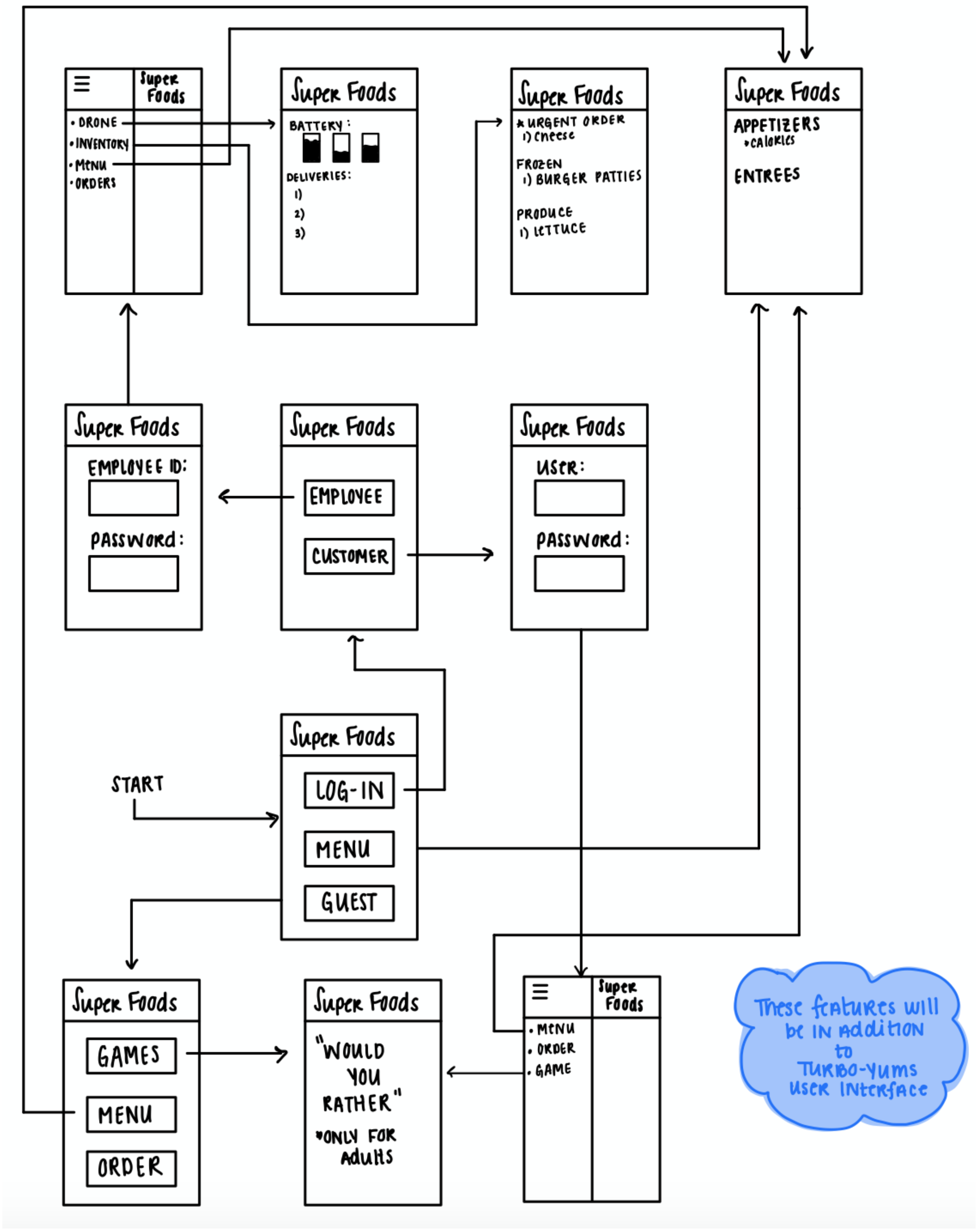
3.4.6 Menu/Order Systems and Customer Interface

Identifier	Priority (Higher number indicates higher priority)	Requirement
REQ - T23	5	System should display buttons to customers to add items to cart.

		 $9.70 prime & FREE Returns FREE delivery: **Friday, Feb 12** Order within 5 hrs and 52 mins Details In Stock. Qty: 1 Add to Cart
REQ - T24	5	System should display buttons to customers to place an order.  Place your order By placing your order, you agree to Amazon's privacy notice and conditions of use. **Order Summary** Items: $147.06 Shipping & handling: $0.00 Amazon Photos Promo: -$10.00 Total before tax: $137.06 Estimated tax to be collected: $9.08
REQ - T25	3	System should display options to modify ingredients.  CUSTOMIZE YOUR CHICKEN ☐ No Cheese Subtracts 110 Cals ☐ No Mexi-Ranch Subtracts 270 Cals ☐ No Pico De Gallo Subtracts 25 Cals ☐ No Tortilla Strips Subtracts 160 Cals
REQ - T26	1	Managers should have the ability to include pictures in their menu items.

		
REQ - T27	5	System should display tables to host, colored in red if a table is waiting for an order, green if the table is occupied, and neutral if a table is free, yellow if the table needs to be bussed. 
REQ - T28	5	System should display order tickets to chefs and bartenders in the order in which they came.  9:10 AM Full Staff 7 100% Expo FD Fries 8 Cooking Sesame Buns 7 Cooking Burger Patties 9 Cooking Baked Potatoes 0/0 Cooking/Pending CHIX TENDERS FRIED CHEESE DBL CHZBURGER LAMB CHOPS FILET MIGNON CHZBURGER HAMBURGER POTATO SKINS WINGS REUBEN GRILLED CHEESE NACHOS
REQ - T29	4	System should display buttons to chefs and bartenders to allow indication of a finished item.
REQ - T30	4	System should display one login screen for different occupations of a restaurant (Host, Chef, Bartender, Manager, Customer).

		Login Name Password Stay signed in Forgot Password? Log In


Super Foods
• DRONE
• INVENTORY
• Menu
• ORDERS
Super Foods
BATTERY:
DELIVERIES:
1)
2)
3)
Super Foods
★ URGENT ORDER
1) cheese
FROZEN
1) BURGER PATTIES
PRODUCE
1) lettuce
Super Foods
APPETIZERS
• calories
ENTREES
Super Foods
EMPLOYEE ID:
PASSWORD:
Super Foods
EMPLOYEE
CUSTOMER
Super Foods
User:
PASSWORD:
START
Super Foods
LOG-IN
MENU
GUEST
Super Foods
GAMES
MENU
ORDER
Super Foods
"WOULD YOU RATHER"
*ONLY FOR Adults
Super Foods
• MENU
• ORDER
• GAME
These features will be in addition to Turbo-Yums user interface

Section 4: Function Requirements Specification

4.1: Use Cases - Casual Descriptions

Drone/Driven Delivery Service

- **UC-1: Inputting Delivery Address-** Allows user to input delivery address
  Derivations: REQ-D1, REQ-T6
- **UC- 2: Drone Tracking -** Allows drone operators to track battery levels, track the location of the drones, and track issues that occur with drones.
  Derivations: REQ-D3, REQ-D4, REQ-D5, REQ-D6, REQ-D10, REQ-D12
- **UC-3: Order Tracking -** Allows the customer to track the location of their order
  Derivations: REQ-T6
- **UC-4: Decision Making -** Informs the drone operator if drones should be used or not
  Derivations: REQ-D2, REQ-D7, REQ-D8, REQ-D9, REQ-D10, REQ-D11, REQ-D13

Adult "Would You Rather" Game App

- **UC-5: Game Advancing -** Allows the customer to start the game from the game menu, guide themselves through a tutorial, and advance themselves through the questions.
  Derivations: REQ-A1, REQ-A2, REQ-A3, REQ-A4, REQ-A5, REQ-A6, REQ-A7, REQ-A8, REQ-A9, REQ-A10, REQ-A11, REQ-A12, REQ-A13, REQ-A14, REQ-A15, REQ-A16, REQ-17, REQ-A18, REQ-A19

Inventory

- **UC-6: Delivering Shipments -** Orders that have been placed and delivered to fulfill inventory needs
  Derivations: REQ-I5, REQ-I6, REQ-I7
- **UC-7: Displaying Inventory:** A screen type display system with options to click on inventory list depending on the category and displays: Green(Over half), Yellow(Halfway), Red(Low). Also provides an up to date version of the inventory count.
  Derivations: REQ-I2, REQ-I3 REQ-I8, REQ-I9, REQ-I10, REQ-I12, REQ-C7, REQ-I13
- **UC-8: Alerting for Low Stock:** Easy display alert system that notifies the manager of items in low stock and the manager has to acknowledge the alert to get rid of the notification
  Derivations: REQ-I4, REQ-I11
- **UC-9: Updating Menu Items -** Allows employees to manually add ingredients and their caloric values through the inventory. Also allows managers to update item descriptions and availability.
  Derivations: REQ-C4, REQ-C8, REQ-C9, REQ-C10, REQ-C12, REQ-T2, REQ-T4, REQ-T27, REQ-I1, REQ-I3, REQ-I4

<u>Calories/ Health Data</u>

- **UC-10: Displaying Menu Item:** Customers should be able to view calories for each menu item and the included descriptions with the calorie count. Also allows customers to view menu items containing or not containing certain ingredients according to dietary restrictions

  <u>Derivations:</u> REQ-C2, REQ-C3, REQ-C5, REQ-C6, REQ-C10, REQ-C11, REQ-C12, REQ-C13, REQ-T12, REQ-T17, REQ-T20, REQ-T21, REQ-T24, REQ-T26, REQ-T27
- **UC-11: Switch Menu:** Allows customers to switch between menus with and without calories and nutritional information

  <u>Derivations:</u> REQ-C1, REQ-C13, REQ-C14

<u>Menu / Order system and Customer Interface</u>

- **UC-12: Creating Account -** There are two types of accounts that can be made: Customer and Employee. Customer accounts lets the customer create an account containing their username, password, email, and birthday, where they may receive account holder benefits. The Employee account allows the manager to create employee accounts containing their username, password, email, SS and other proper credentials. The manager will also set the employee status which determines the features available to the employee.

  <u>Derivations:</u> REQ-T3, REQ-T23, REQ-T30
- **UC-13: Customer and Employee Login -** Allow users to login in with their accounts with different functions based on the role of their account.

  <u>Derivations:</u> REQ-T3, REQ-T4, REQ-T13, REQ-T16, REQ-T22, REQ-T29, REQ-T30, REQ-I9
- **UC-14: View and Manage Customer Account and Order -** Customers are able to view their reward status, view and manage past/current orders, leave a star rating for the order, and save card information for use in future orders.

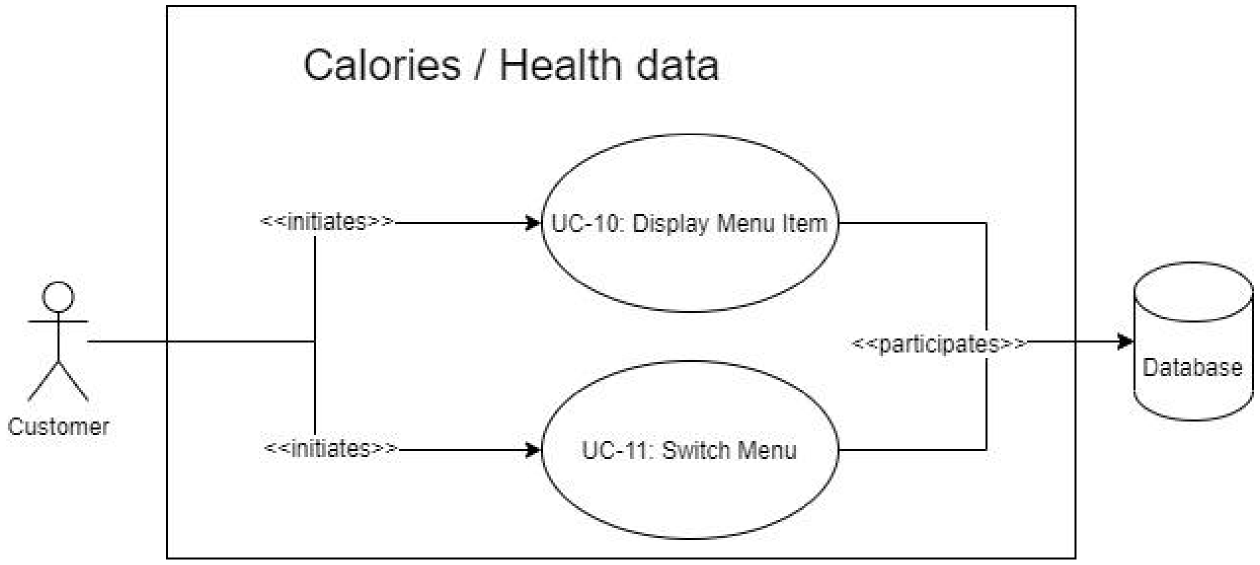
  <u>Derivations:</u> REQ-T1, REQ-T3 REQ-T5, REQ-T6, REQ-T7, REQ-T9, REQ-T11, REQ-T12, REQ-T13, REQ-T15, REQ-T16, REQ-T17, REQ-T18, REQ-T19, REQ-T21, REQ-T25
- **UC-15: Point Redemption System -** Allows the customer to redeem points towards a coupon for their next meal from their account.

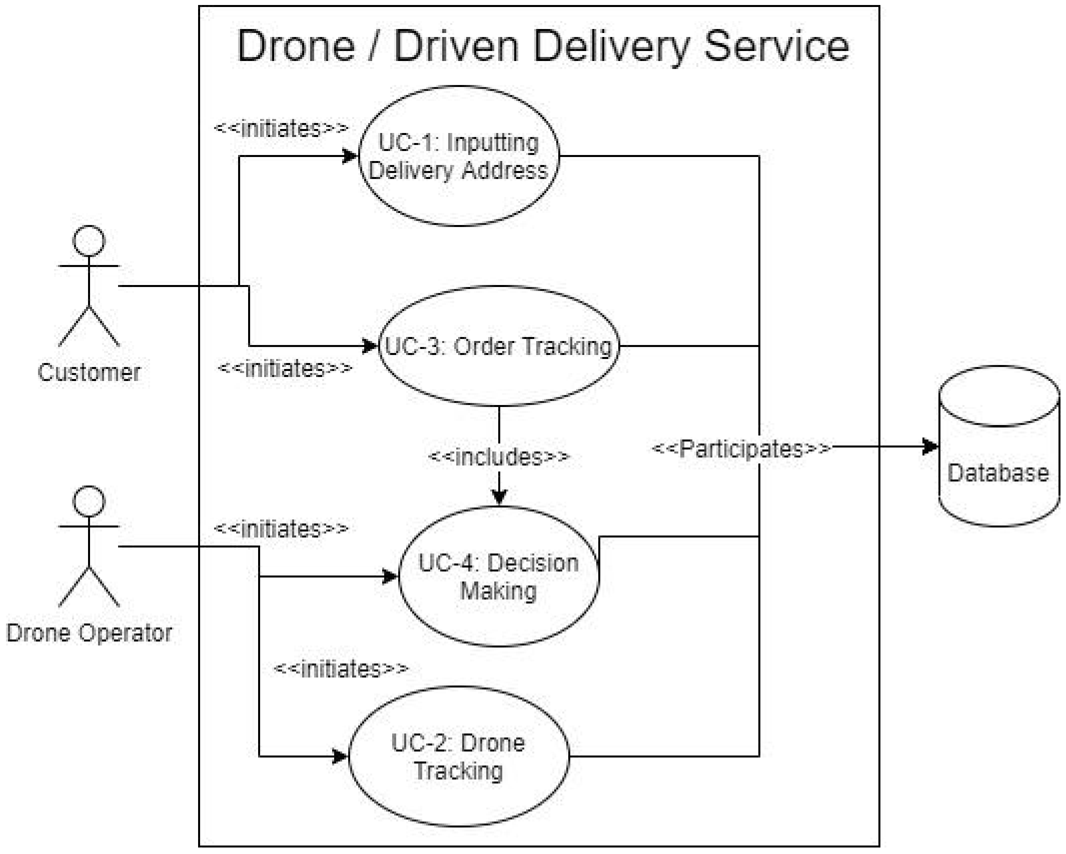
  <u>Derivations:</u> REQ-T3, REQ-T14, REQ-T18, REQ-T19
- **UC-16: Stop Delivery -** Allows the manager to select a time to stop customers from selecting delivery when ordering.

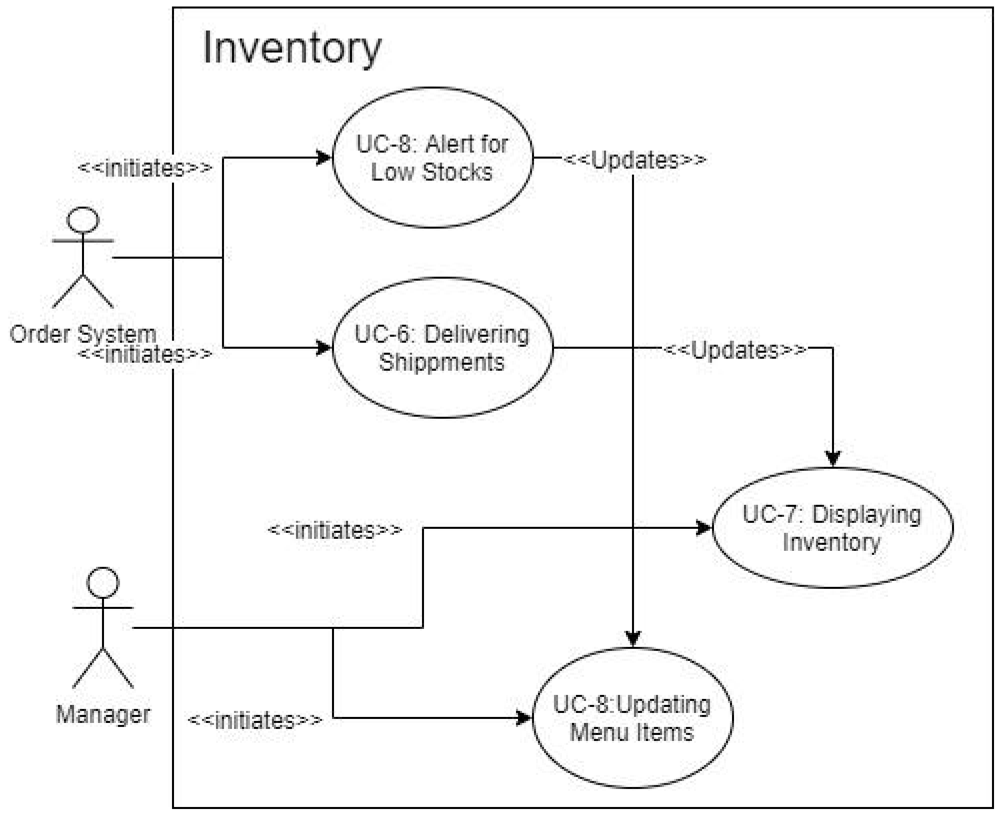
  <u>Derivations:</u> REQ-T8, REQ-T10
- **UC-17: Tables Display -** Allows host to view the table layout based on a color coded scheme where each color is on table status.

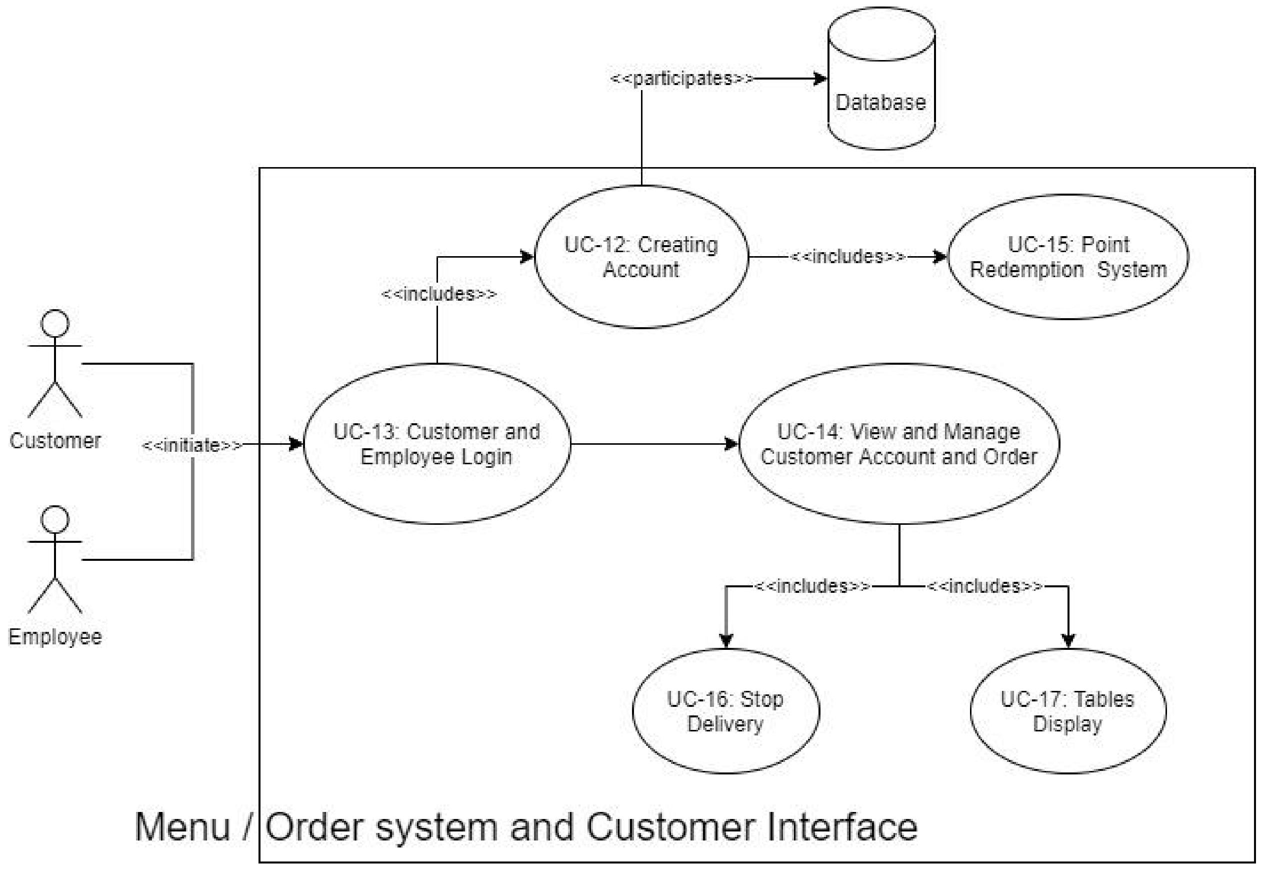
  <u>Derivations:</u> REQ-T28

4.2: Use Cases - Diagrams






Inventory
<<initiates>>
UC-8: Alert for Low Stocks
<<Updates>>
Order System
<<initiates>>
UC-6: Delivering Shippments
<<Updates>>
UC-7: Displaying Inventory
<<initiates>>
Manager
<<initiates>>
UC-8:Updating Menu Items


<<participates>>
Database
UC-12: Creating Account
<<includes>>
UC-15: Point Redemption System
<<includes>>
Customer
<<initiate>>
UC-13: Customer and Employee Login
UC-14: View and Manage Customer Account and Order
Employee
<<includes>>
<<includes>>
UC-16: Stop Delivery
UC-17: Tables Display
Menu / Order system and Customer Interface

4.3: Traceability Matrix

	Use Case																
Requirement	**PW**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
D1	4	X																
D2	5				X													
D3	4		X															
D4	3		X															
D5	3		X															
D6	3		X															
D7	5				X													
D8	5				X													
D9	5				X													
D10	5		X		X													
D11	5				X													
D12	5		X															
D13	2				X													
A1	1					X												
A2	1					X												
A3	2					X												
A4	4					X												
A5	3					X												
A6	5					X												
A7	5					X												
A8	1					X												
A9	5					X												
A10	3					X												
A11	5					X												
A12	5					X												
A13	2					X												
A14	2					X												
A15	2					X												
A16	1					X												
A17	4					X												
A18	5					X												

A19	5					X												
I1	5									X								
I2	4							X										
I3	4							X										
I4	2								X	X								
I5	4						X											
I6	4						X											
I7	2						X											
I8	3							X										
I9	3							X						X				
I10	2							X										
I11	2								X									
I12	3							X										
I13	2							X										
C1	2											X						
C2	5										X							
C3	5										X							
C4	4.5									X								
C5	3										X							
C6	3										X							
C7	5							X										
C8	5									X								
C9	4									X								
C10	5									X	X							
C11	3										X							
C12	5									X	X							
C13	3										X	X						
C14	2											X						
T1	5														X			
T2	5									X								
T3	4												X	X	X	X		
T4	3									X				X				
T5	5														X			
T6	3	X		X											X			
T7	3														X			

T8	3																X	
T9	3														X			
T10	5																X	
T11	1														X			
T12	2										X				X			
T13	2													X	X			
T14	1															X		
T15	1														X			
T16	4													X	X			
T17	3										X				X			
T18	1														X	X		
T19	2														X	X		
T20	3										X							
T21	4										X				X			
T22	5													X				
T23	5												X					
T24	5										X							
T25	3														X			
T26	1										X							
T27	5									X	X							
T28	5																	X
T29	4													X				
T30	4												X	X				

4.4: Use Cases - Fully Dressed Descriptions

UC-4: Decision Making
Related Requirements: REQ-D2, REQ-D7, REQ-D8, REQ -D9, REQ-D10, REQ-D11, REQ-D13
Initiating Actor: Customer
Actor's Goal: Provide the system with the proper credentials such as address, for the system to perform the proper decision making for delivery method.
Participating Actors: Database
Preconditions: Customer must place an order Customer must enter address being delivered to
Postconditions: Customer will have order confirmation after processing payment
Flow of Events for Main Success Scenario: 1. → Customer enters delivery information including street address 2. ← System converts street address into coordinates 3. ← System decides drone delivery is the proper form of transportation 4. ← System notifies customer time of delivery 5. ← System assigns drone to delivery, taking into account battery parameters 6. ← System notifies employee any issue pertaining to a drone delivery
Flow of Events for Alternate Success Scenario: 1. → Customer enters delivery information including street address 2. ← System converts street address into coordinates 3. ← System decides that car delivery is the proper use of transportation 4. ← System notifies customer time of delivery

UC-9: Updating Menu Items
Related Requirements: REQ-C4, REQ-C8, REQ-C9, REQ-C10, REQ-C12, REQ-T2, REQ-T4, REQ-T27, REQ-I1, REQ-I3, REQ-I4
Initiating Actor: Employee, Manager
Actor's Goal:

Be able to update menu items in the app and add ingredients, calorie indication, item description, and availability.
Participating Actors: Database
Preconditions: Employees must log into the system using their credentials.
Postconditions: The item should be displayed on the Menu with all the necessary information.
Flow of Events for Main Success Scenario: 1. → Employee opens Inventory 2. ← System displays the user Inventory count 3. → Employee opens Menu 4. ← System displays menu with option to change the menu 5. → Employee clicks option and adds the new menu item 6. ← System checks to make sure all requirements are fulfilled 7. ← System updates Menu
Flow of Events for Alternate Success Scenario: 1. → Employee opens Inventory 2. ← System displays the user Inventory count 3. ← System displays option to change the inventory count 4. → Employee Updates inventory count 5. ← System verifies new inventory count 6. ← System updates Menu

UC-13 Customer and Employee Login
Related Requirements: REQ-T34, REQ-I9, REQ-T16, REQ-T19, REQ - T4, REQ -T3
Initiating Actor: All users (customers, employees, managers, etc.)
Actor's Goal: To login to the application to receive account benefits
Participating Actors: Database
Preconditions: The user has the system loaded. The user has their login credentials available.
Postconditions: The user is prompted to the main menu with a selection of variable features based on account

role (employee status, customer)
Flow of Events for Main Success Scenario: 1. → The user clicks what role they are from the main menu (employee or customer) 2. → The system displays the login screen with the login in credentials specified 3. ← The user enters their login credentials such as email and password 4. → The database verifies login and account role (employee or customer) 5. → The system prompts the user to the main menu
Flow of Events for Alternate Success Scenario: 1. → The customer clicks "customer" from the main menu 2. → The system displays the login screen with the login in credentials specified 3. ← The customer clicks on create a new account 4. → The system prompts the user to the account creation screen 5. ← The system display the credentials birthday, email, and password 6. → The customer inputs the credentials 7. ← The system stores the information to the database 8. ← The system prompts the users to the main menu

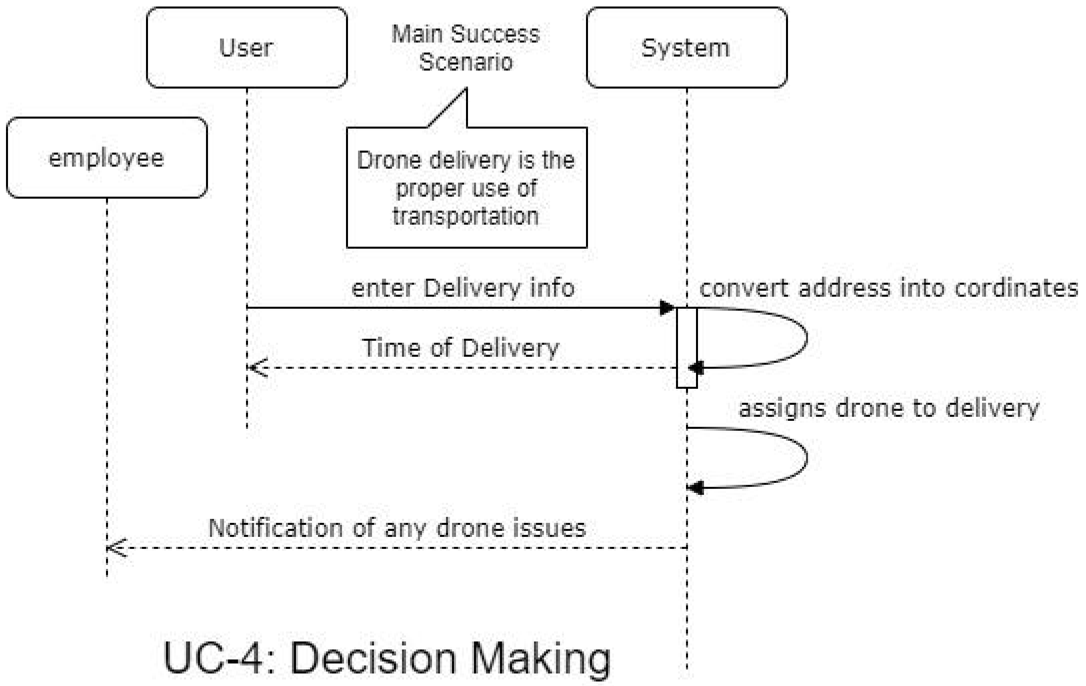
UC-14: View and Manage Customer Account and Order
Related Requirements: REQ-T1, REQ-T3 REQ-T5, REQ-T6, REQ-T7, REQ-T9, REQ-T11, REQ-T12, REQ-T13, REQ-T15, REQ-T16, REQ-T17, REQ-T18, REQ-T19, REQ-T21, REQ-T25
Initiating Actor: Customer
Actor's Goal: To view and manage their order and account.
Participating Actors: Database
Preconditions: Customer logs into application Customer must click order from the drop down menu
Postcondition: Customer is able to view and manage their orders
Flow of Events for Main Success Scenario: 1. → Customer selects "Order" where they are prompted to the menu 2. ← System displays menu with button next to each item offering "Add to order" 3. → Customer selects items to add to their order 4. → Customer clicks on cart 5. ← System shows items in cart, an order total, and options for delivery

6. → Customer chooses option for delivery via press button
7. ← System prompts customer to enter delivery method
8. → Customer enters all necessary information including name and address
9. ← System prompts customer to payment
10. → Customer enters card information or can select “Cash”
11. ← System prompts user to verify information
12. → Customer verifies information
13. ← System confirms order

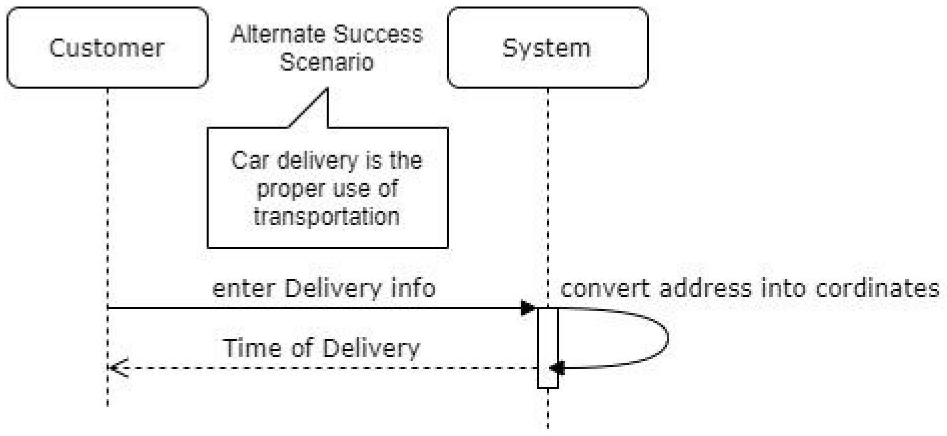
Flow of Events for Alternate Success Scenario:

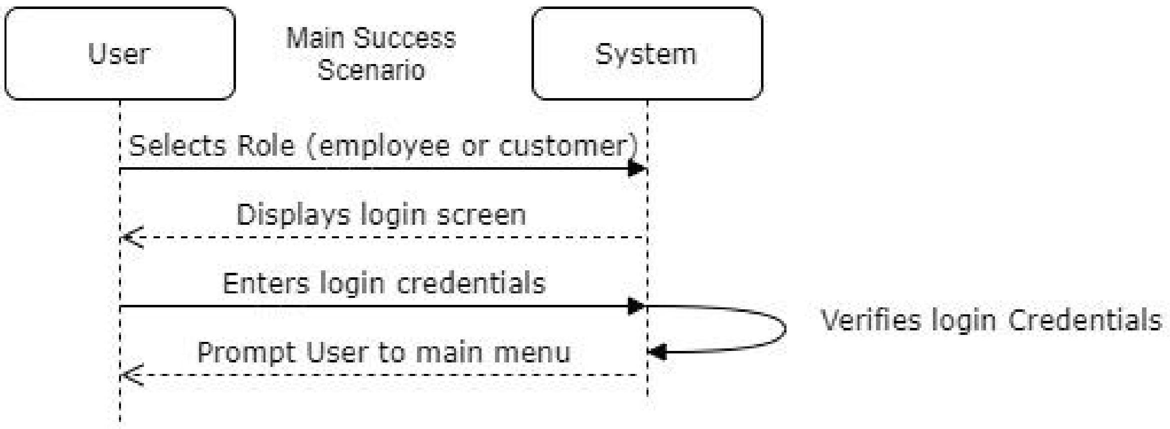
1. → Customer selects “Account” where they are prompted to their account information
2. ← System displays reward options, past orders, reviews, and card information
3. → Customer clicks on card information
4. → Customer updates card information by entering card holder name, card number, cvc, and expiration date.
5. ← System prompts user back to account information
6. → Customer views their reward status and current points
7. → Customer clicks on past orders
8. ← System displays all past orders the account holder has placed, along with the reviews that were given for each order if any.

4.5: System Sequence Diagrams

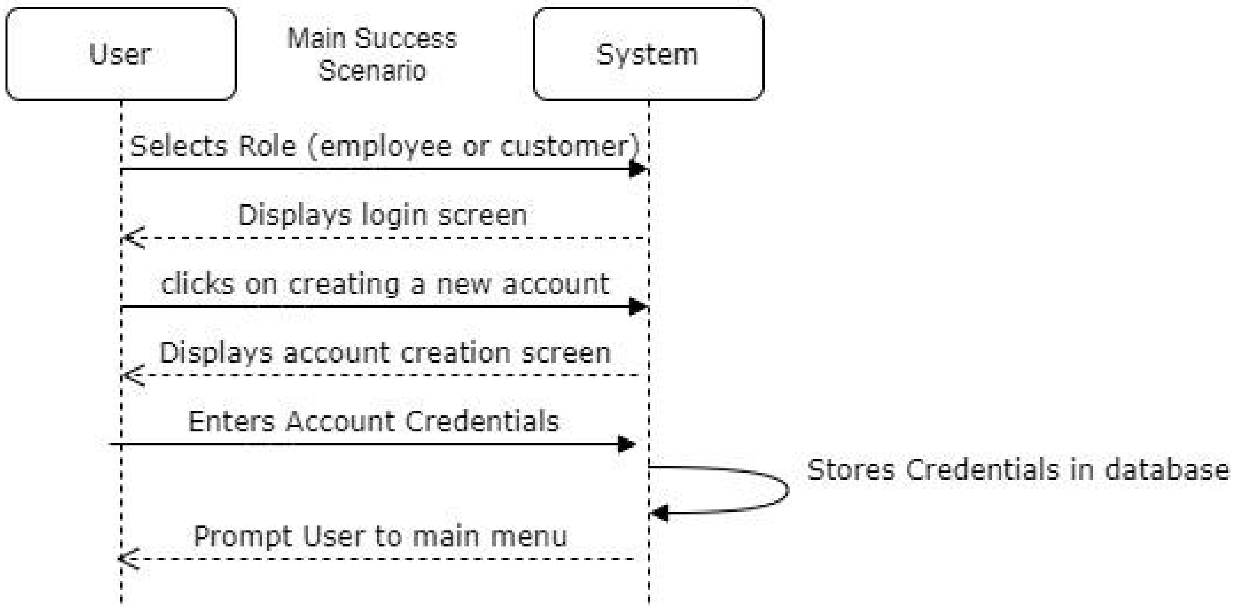

User
Main Success Scenario
System
employee
Drone delivery is the proper use of transportation
enter Delivery info
convert address into cordinates
Time of Delivery
assigns drone to delivery
Notification of any drone issues

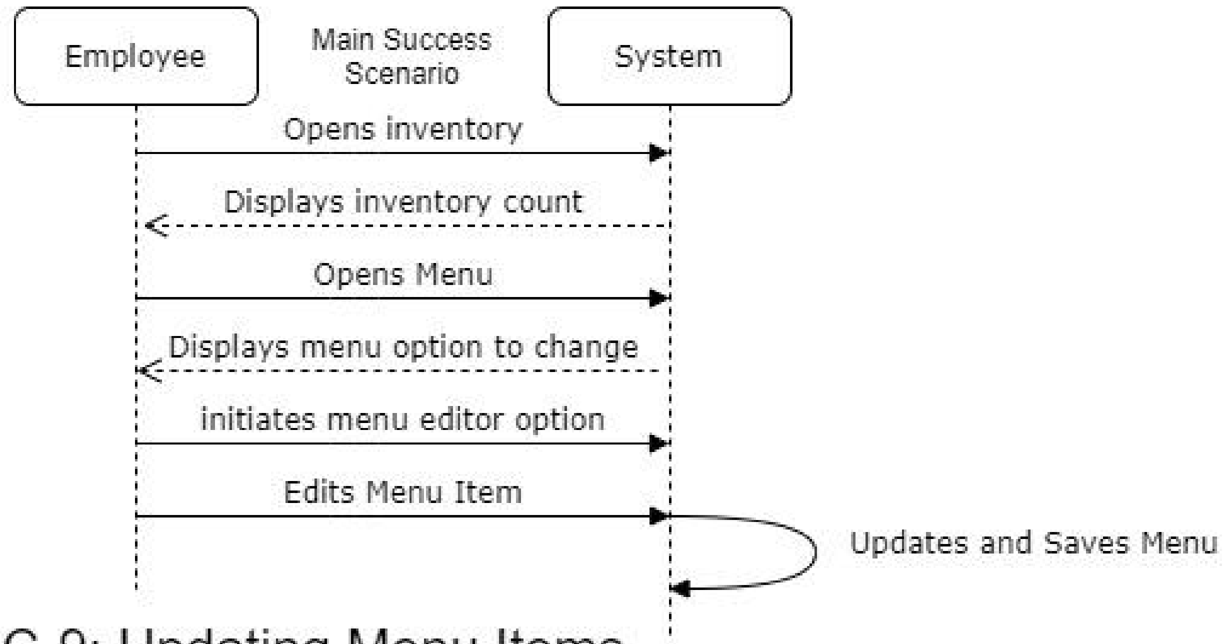
UC-4: Decision Making


Customer
Alternate Success Scenario
System
Car delivery is the proper use of transportation
enter Delivery info
convert address into cordinates
Time of Delivery

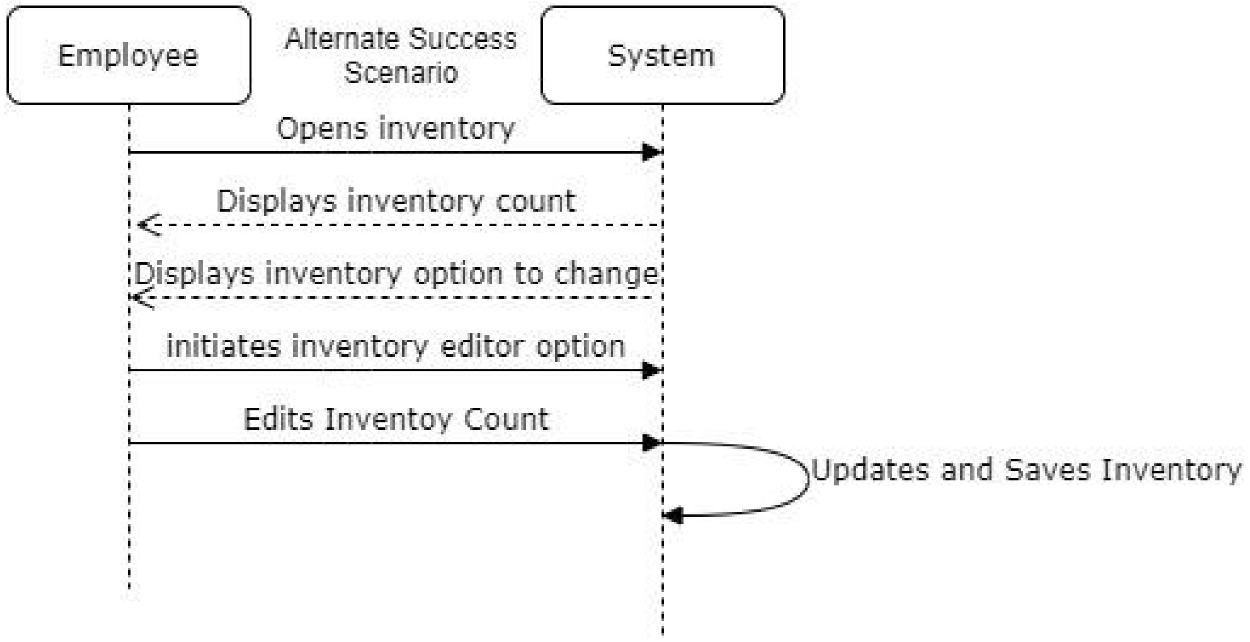


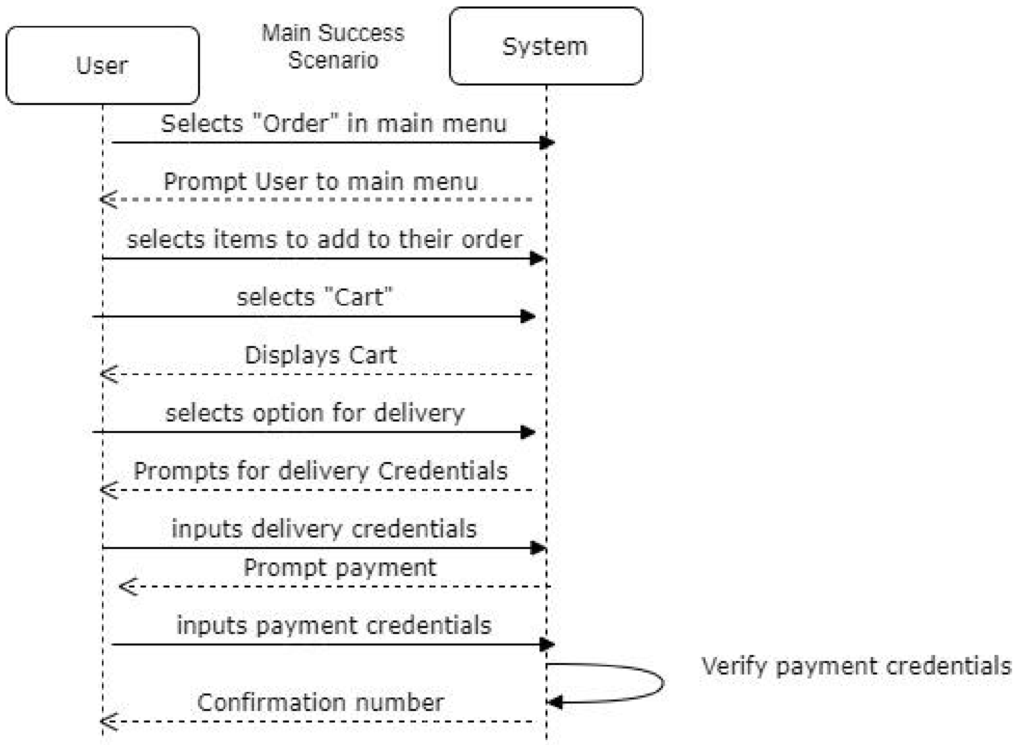
UC-13: Customer and Empoyee Login


User
Main Success Scenario
System
Selects Role (employee or customer)
Displays login screen
clicks on creating a new account
Displays account creation screen
Enters Account Credentials
Stores Credentials in database
Prompt User to main menu



UC-9: Updating Menu Items


Employee
Alternate Success Scenario
System
Opens inventory
Displays inventory count
Displays inventory option to change
initiates inventory editor option
Edits Inventoy Count
Updates and Saves Inventory


User
Main Success Scenario
System
Selects "Order" in main menu
Prompt User to main menu
selects items to add to their order
selects "Cart"
Displays Cart
selects option for delivery
Prompts for delivery Credentials
inputs delivery credentials
Prompt payment
inputs payment credentials
Verify payment credentials
Confirmation number

UC-14: View and Manage Customer Account and Order

Section 5: Effort Estimation using Use Case Points

CUSTOMER

Best Case Scenario

1. New customer chooses to log-in as a guest. (1 Click)
2. Chooses Menu. (1 click)
3. selects food category. (1 Click)
4. selects food item. (1 Click)
5. adds item to cart. (1 Click)
6. clicks on cart. (1 Click)
7. chooses to in store (1 Click)
8. chooses payment option as cash (1 Click)
9. selects the “No Receipt” option. (1 Click)
10. chooses not to rate their experience. (1 Click)

Total = 10 Clicks

Worst Case Scenario **(note: capital letters represent constants)**

1. Enters Username of ‘U’ characters (U Clicks)
2. Enters Password of ‘P’ characters (P Clicks)
3. chooses the login option (1 Click)
4. enters wrong information (repeats steps 1-3 ‘n’ times)
5. opens menu (1 click)
6. chooses appetizer category (1 click)
7. chooses appetizer (1 click)
8. skims through ‘A’ appetizers (repeats steps 7 ‘A’ times)
9. makes ‘x’ modifications to appetizer (x clicks)
10. adds app to cart. (1 Click)
11. orders for ‘m’ people (repeats steps 7,9,10 ‘m’ times)
12. chooses drink category (1 click)
13. chooses drink (1 click)
14. skims through ‘D’ drinks (repeats steps 13 D times)
15. adds drink to cart with ‘s’ specifications. (1+s Clicks)
16. chooses entree category (1 click)
17. chooses entree (1 click)
18. skims through ‘E’ entrees (repeats step 17 E times)
19. makes ‘y’ modifications to entree (y clicks)

20. adds entree to cart. (1 Click)
21. orders for ‘m’ people (repeat steps 12-13,15-17,19-20 ‘m’ times)
22. Goes back to main screen. (1 Click)
23. chooses to play game (1 click)
24. chooses tutorial and continues to game after (2 clicks)
25. chooses to play ‘r’ rounds (r Clicks)
26. exits game to main menu (1 click)
27. clicks on cart. (1 Click)
28. chooses to in store (1 Click)
29. chooses payment option as card (1 Click)
30. chooses card number box and enters card number (1+12 Clicks)
31. chooses cvv box and enters cvv (1+3 Clicks)
32. chooses expiration box date and enters expiration date (1+4 Clicks)
33. selects the “Print Receipt” option. (1 Click)
34. chooses star rating. (1 Click)
35. chooses to leave ‘c’ character review. (‘c’ Clicks)

Total = n*(U+P+1) + 1 + 1 + A + m*(2+x) + 1 + D + m*(2+s) + 1 + E + m*(2+y) + 1 + 4 + r + 27 + c

= n*(U+P+1) + m*(6 + x + s + y) + A + D + E + 36 clicks

EMPLOYEES

Best Case Scenario

1. Enters Username of ‘U’ characters (U Clicks)
2. Enters Password of ‘P’ characters (P Clicks)
3. chooses the login option (1 Click)
4. enters wrong information (repeats steps 1-3 ‘n’ times)
5. no tables
6. Signs out (1 click)
7. Confirms sign out (1 click)

Total = n*(U+P+1) + 2 clicks

Worst Case Scenarios

CHEF/BARTENDER

1. Enters Username of ‘U’ characters (U Clicks)

2. Enters Password of ‘P’ characters (P Clicks)
3. chooses the login option (1 Click)
4. enters wrong information (repeats steps 1-3 ‘n’ times)
5. confirms completion of ‘i’ items (i clicks)
6. Signs out (1 click)
7. Confirms sign out (1 click)

Total = n*(U+P+1) + i + 2 clicks

HOST

1. Enters Username of ‘U’ characters (U Clicks)
2. Enters Password of ‘P’ characters (P Clicks)
3. chooses the login option (1 Click)
4. enters wrong information (repeats steps 1-3 ‘n’ times)
5. chooses table to occupy (1 click)
6. confirms occupation (1 click)
7. seats ‘t’ tables (repeats 5-6 ‘t’ times)
8. Signs out (1 click)
9. Confirms sign out (1 click)

Total = n*(U+P+1) + 2*t + 2 clicks

MANAGER

1. Enters Username of ‘U’ characters (U Clicks)
2. Enters Password of ‘P’ characters (P Clicks)
3. chooses the login option (1 Click)
4. enters wrong information (repeats steps 1-3 ‘n’ times)
5. selects inventory (1 click)
6. selects ingredient (1 click)
7. enters quantity (3 clicks - Assuming can reach hundreds range)
8. updates quantity (1 click)
9. repeats for ‘i’ ingredients (repeats steps 5-8 ‘i’ times)
10. Signs out (1 click)
11. Confirms sign out (1 click)

Total = n*(U+P+1) + 6*i + 2 clicks

Section 6: Domain Analysis

6.1: Conceptual Modeling

6.1.1: Concept Definition

Sub-Project	Responsibility	Type	Concept
Drone Delivery	**R1:** Display drone delivery queue, and drone information (Battery level, drone tracking, emergency request)	D	Controller
	R2: Display status and location of order (viewable by both: Employees and Customer)	D	Order Display
Game App	**R3:** Prompts customers to play the “Would You Rather” game(includes Tutorial, play option, skip option, and end option)	D	Game Play
Inventory	**R4:** Display proper inventory count	D	Inventory Interface
	R5: Manages shipment information and provides information for low-stock	D	Inventory Interface
	R6: Display alerts (Low-stock, shipment, etc)	D	Alert Display
Calories / Health Data	**R7:** Display filtered menu using user input for nutritional information	D	Controller
	R8: Display total menu health data	D	Calorie Display
Menu/Order Systems and Customer Interface	**R9:** Display the options for the customer, waiter, chef, and manager respectively	D	Interface
	R10: Store employee login information	K	Employee Profile
	R11: Handle payment processing	D	Payment System
	R12: Store customer rewards points based upon previous orders	K	Customer Profile
	R13: Store customer login information	K	Customer Profile

	R14: Manage interactions with the database	K	DB Connection
	R15: Display change of table status when a customer selects a table, leaves, and when a busboy cleans a table	D	Table Status
	R16: Store the customer order in the database	K	Customer Profile
	R17: Allows Employees and Customers to input information and stores the data	D	Interface
	R18: Allows customers to leave a review(words+stars)	D	Interface

6.1.2: Association Definitions

Concept Pair	**Association Description**	**Association Name**
Customer Profile ↔ DB Connection	Fetch customer's data from the database	QueryDB
Customer Profile ↔ Interface	Display customer's option	Display
Interface ↔ Controller	Allow the user to interact with the app	User Action
Communicator ↔ DB Connection	Modify or insert data into the database	UpdateDB
Communicator ↔ Order Queue	Send order and queue to the database	QueryDB
Controller ↔ Table Status	Allow user to view table status	View TableStatus
Controller ↔ Payment System	Allow user to complete the payment	Make Payment
Payment System ↔ DB Connection	Store payment record in the database	Record Payment
Interface ↔ DB Connection	Get the data from the database for the user	QueryDB
Customer Profile ↔ DB Connection	Stores earned rewards/points in the database	Reward System
Employee Profile ↔ Interface	Displays all employee information	Display
Table Status ↔ Interface	Displays the current table layout with the status of the tables	Display
Inventory Count ↔ Inventory Interface	Displays proper inventory count for the products	QueryDB

Calorie Display ↔ Interface	Displays all caloric and health data for the customer and employee	Display
Controller ↔ Calorie Display	Allow user to rotate between filtered menus	Display
Communicator ↔ Order Display	Displays orders in a delivery queue for the drones	Delivery System
Inventory Interface ↔ Alert Display	Alerts the manager of any low stock inventory	Display

6.1.3: Attribute Definitions

Concept	**Attribute**	**Description**
Customer Profile	accountEmail	Associated email of the customer. A guest account is assigned if no account
	accountPassword	Password of the user account
	accountBirthday	Associated Birthday of the customer. The guest account contains a null birthday.
Order Display	confirmOrder	Allows the user to confirm the order after choosing food items
	mangeOrder	Allows users to delete items from their cart and track the status of their order.
	cartDisplay	Displays all the items the user has selected to add to their cart
	cartAdd	Allows the user to add a food item to their cart
Game Play	Play	Allows the customer to play the “Would You Rather” game.
	Skip	Allows the customer to skip a “Would You Rather” question.
	Quit	Allows the customer to quit the game at any time.
Inventory Interface	inventoryDisplay	Displays inventory list and status based on color (Green: Over half, Yellow: Halfway, Red: Low)
	inventoryUpdate	Allows employees to update the inventory count of items
Alert Display	lowStock	Notifies manager of items in low stock
	wrongPassword	Notifies the user if their password was incorrectly

		inputted
	flyDrive	Notifies drone operator whether delivery should be made with drones or delivery driver
Calorie Display	foodFilter	Displays food items based on dietary restrictions
	menuSwitch	Displays 2 different menus based on customer's need (With or Without calories and nutritional information)
	calorieCount	Displays total calorie count of order
Interface	tableStatus	Provides the user with the up-to-date status of each table in the restaurant. Tables can exist in 3 states: available, occupied and dirty
	rateMeal	Allows the user to rate a meal, and then have that rating stored and displayed
Employee Profile	receipt	Allows the user to choose which way they would like to receive the receipt (email, text, paper)
	hoursWorked	Displays hours a particular employee has worked in a particular payment cycle
	tablesAssigned	Shows what tables the employee has been assigned recently
	role	Role of the employee: food server, chef/cook, drone operator, etc.
Payment System	paymentMade	Updated system depending on whether the payment has been made
Controller	paymentMade	Once the customer pays, the table is then confirmed and the order is initiated in the kitchen
	tableConfirm	The table is greyed out once it is picked by the customer and stays grey until cleaned
	tableList	Shows the customer the current tables available
Reward System	currUserPoints	Displays present user reward points balance
	rewardsRedeemable	Allows the user to redeem available rewards
DB Connection	dbAuthenticationCreds	Stores the DB login for authorized personnel
Table Status	tableNumber	Display the table number
	seatCount	Mx number of people that can be seated at a

		particular table
	currTableStatus	Displays table color based on if it’s empty, occupied, or unclean.

6.1.4: Traceability Matrix

		Domain Concepts									
Use Case	PW	Controller	Order Display	Inventory Interface	Alert Display	Calorie Display	Employee Profile	Customer Profile	Payment System	Table Status	Interface
1	3.5		X					X			
2	3.83	X			X						
3	3	X	X								X
4	4.57				X						
5	3.21	X									X
6	3.33			X							
7	3.25			X							
8	2			X	X						
9	4.05	X		X							
10	3.82			X		X					
11	2.33	X				X					
12	4.33						X	X			X

13	3.625						X	X			X
14	2.875		X					X	X		X
15	2							X	X		
16	4	X	X								
17	5	X								X	X

Drone/Driven Delivery Service

- **UC-1: Inputting Delivery Address -** When the customer is in their profile and completing an order, they have the option to input their address into the provided boxes and also save the address within their profile.
- **UC-2: Drone Tracking -** The controller must keep track of the drones, battery levels, and any occurring issues with said drone. If there is an issue, there will be a displayed alert notifying the drone operator.
- **UC-3: Order Tracking -** On the user interface, the controller keeps track of the order status of the order, and displays all necessary information to the customer.
- **UC-4: Decision Making -** When decisions need to be made on whether or not a drone should be flying, the alert display will notify the drone operator with any issues.

Adult "Would You Rather" Game App

- **UC-5: Game Advancing -** The controller allows the customer to navigate their way through the game, going through the tutorial as well as different questions, and they will all be displayed on the user interface.

Inventory:

- **UC-6: Delivering Shipments -** The inventory interface will display items that are needed to be ordered for the next shipment delivery.
- **UC-7: Displaying Inventory -** The inventory interface will show all items in inventory as well as their price and stock number.
- **UC-8: Alerting for Low Stock -** The inventory interface will monitor the stock of the items, and once an item reaches low stock, there will be an alert displayed.
- **UC-9: Updating Menu Items -** The controller allows the user to update the items including their inventory count and descriptions, and it will then be displayed on the inventory interface.

Calories/Health Data

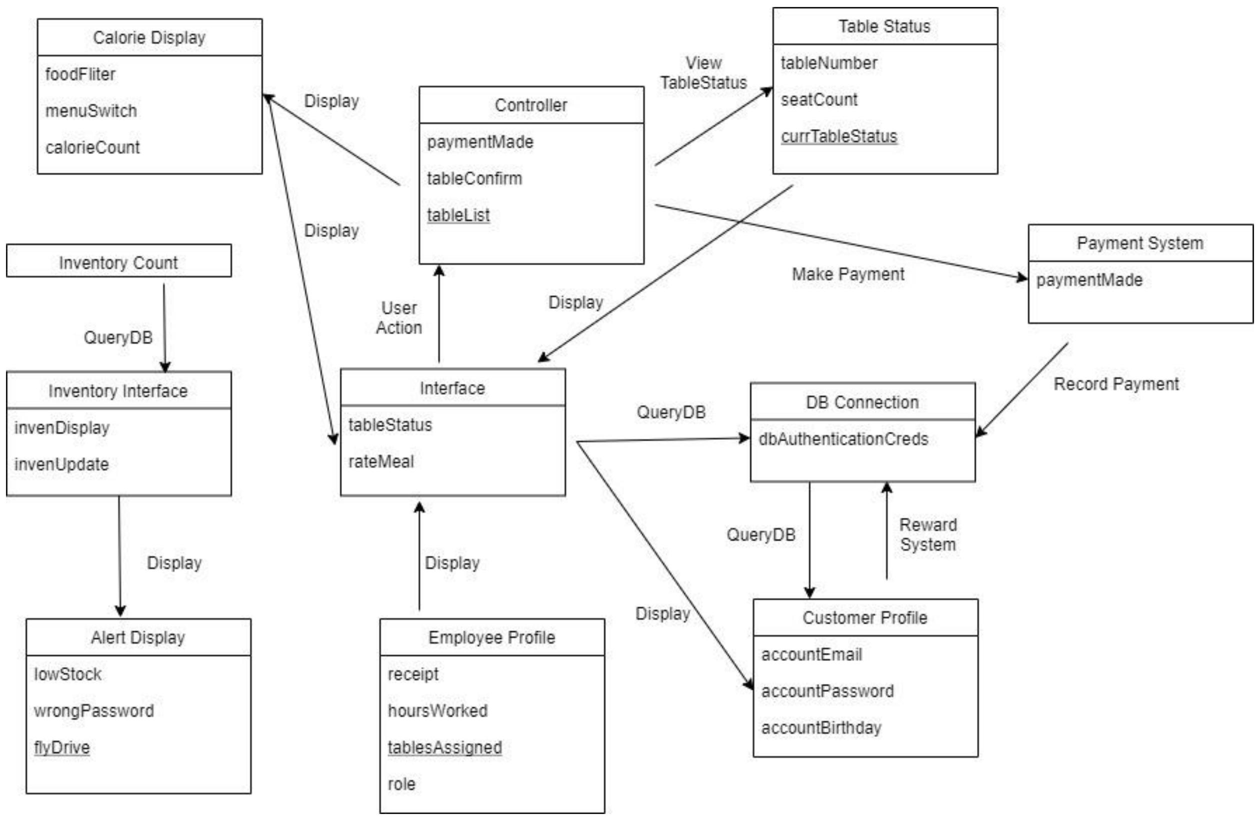
- **UC-10: Displaying Menu Item -** A filter function will allow the user to control what items are displayed on the menu based on restricted diet or allergies.
- **UC-11: Switch Menu -** The controller will allow the user to switch between whether or not they want the nutritional facts to be displayed on the menu for each menu item.

Menu/Order Systems and Customer Interface

- **UC-12: Creating Account -** An account is able to be created for either the customer profile or the employee profile. This will all be managed through the interface.
- **UC-13: Customer and Employee Login -** Either the employee or the customer will be able to login to their profile through the user interface.
- **UC-14: View and Manage Customer Account and Order -** The customer is able to manage their customer profile through the interface which allows them to use order to display to manage past/present orders and update their payment preferences in their profile.
- **UC-15: Point Redemption System -** The customer profile keeps track of all the rewards the customer has acquired or spent, and you are able to use them for future purchases through the payment system.
- **UC-16: Stop Delivery -** The controller will allow the manager to make changes to stop the deliveries that are coming through the ordering system.

- **UC-17: Tables Display -** The controller will allow the employees to keep track and manage the table status of those in the restaurant -- this will be displayed on the user interface.

6.1.5: Domain Model Diagram



6.2: System Operation Contracts

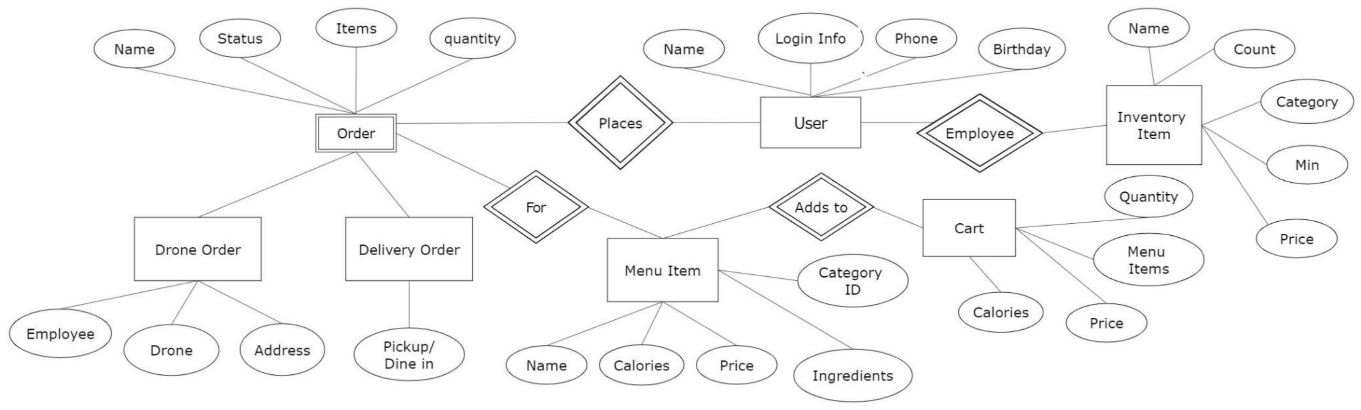
Operation	**Decision Making**
Use Case	UC-4
Preconditions	Customer must place an order Customer must enter address being delivered to
Postconditions	Customer will have order confirmation after processing the payment

Operation	Updating Menu Items
Use Case	UC-9
Preconditions	Employees must log into the system using their credentials .
Postconditions	The item should be displayed on the Menu with all the necessary information.

Operation	Customer and Employee Login
Use Case	UC-13
Preconditions	The user has the system loaded. The user has their login credentials available.
Postconditions	The user is prompted to the main menu with a selection of variable features based on account role (employee status, customer)

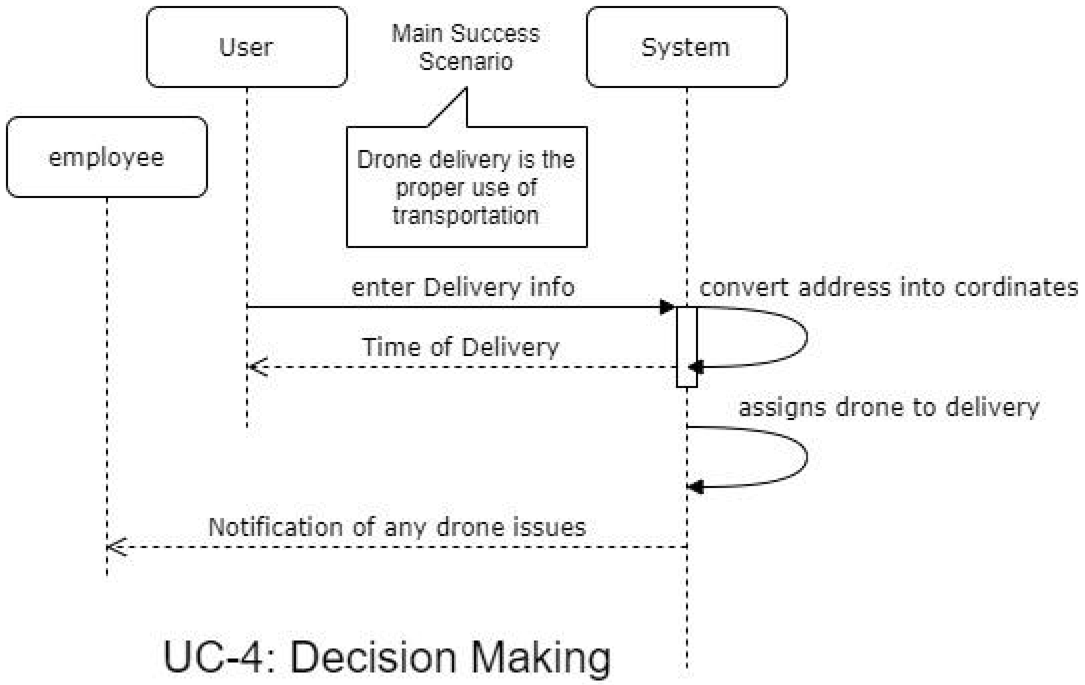
Operation	View and Manage Customer Account Order
Use Case	UC-14
Preconditions	Customer logs into application Customer must click order from the drop down menu
Postconditions	Customer is able to view and manage their orders

6.3: Data Model

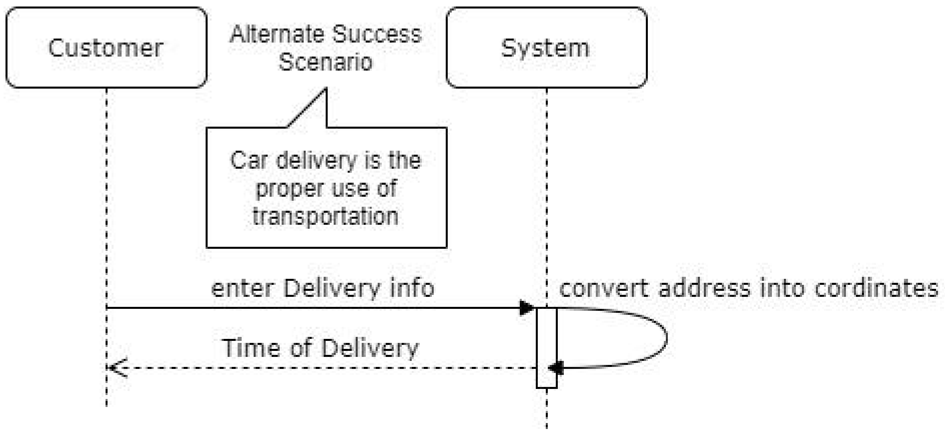


Section 7: Interaction Diagrams

UC-4: Decision Making

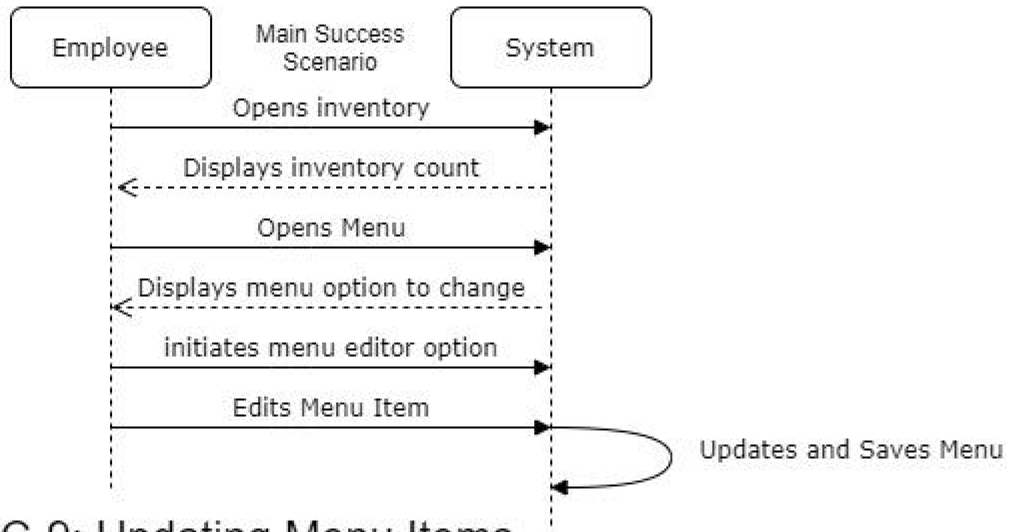

User
Main Success Scenario
System
employee
Drone delivery is the proper use of transportation
enter Delivery info
convert address into cordinates
Time of Delivery
assigns drone to delivery
Notification of any drone issues

UC-4: Decision Making

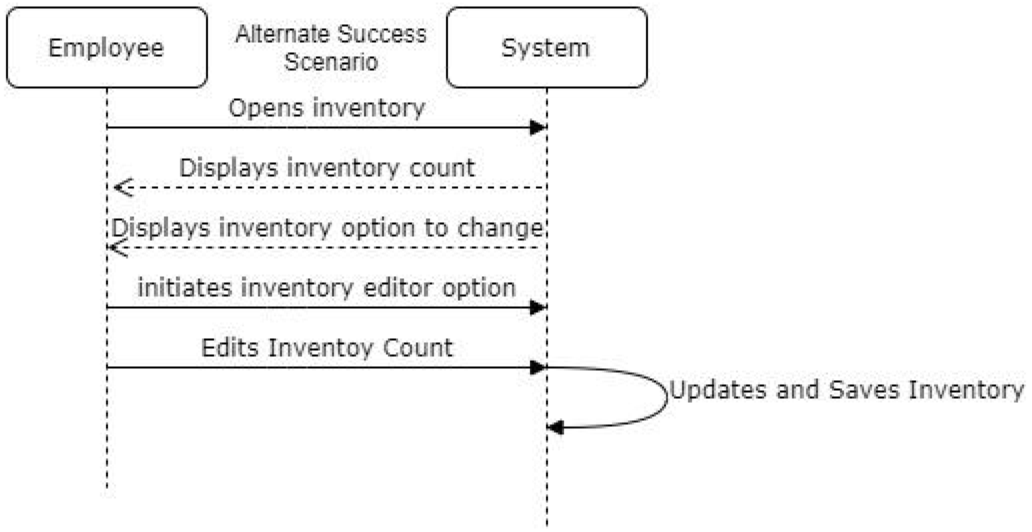

Customer
Alternate Success Scenario
System
Car delivery is the proper use of transportation
enter Delivery info
convert address into cordinates
Time of Delivery

The User works with the System in order to retrieve addresses for the Decision Matrix. The system uses this to obtain time of delivery, which factors in the Decision matrix output. In addition if any drone issues occur (such as broken components), the system will alert the employee, specifically the Drone Manager.

UC-9: Updating Menu Items

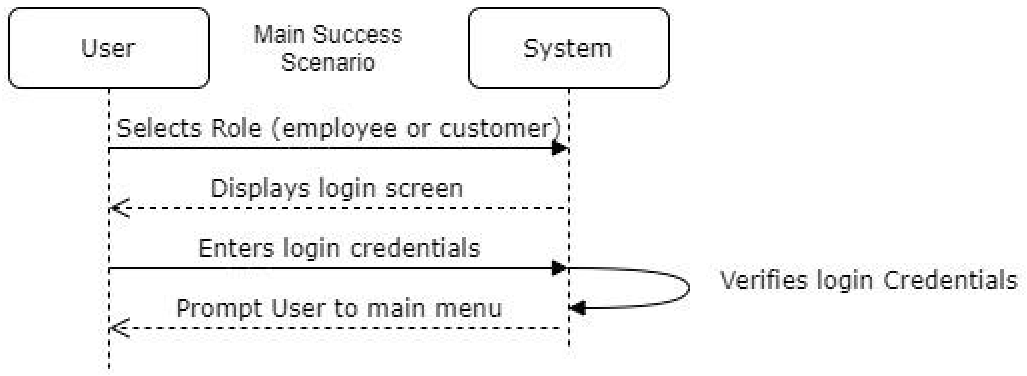


UC-9: Updating Menu Items

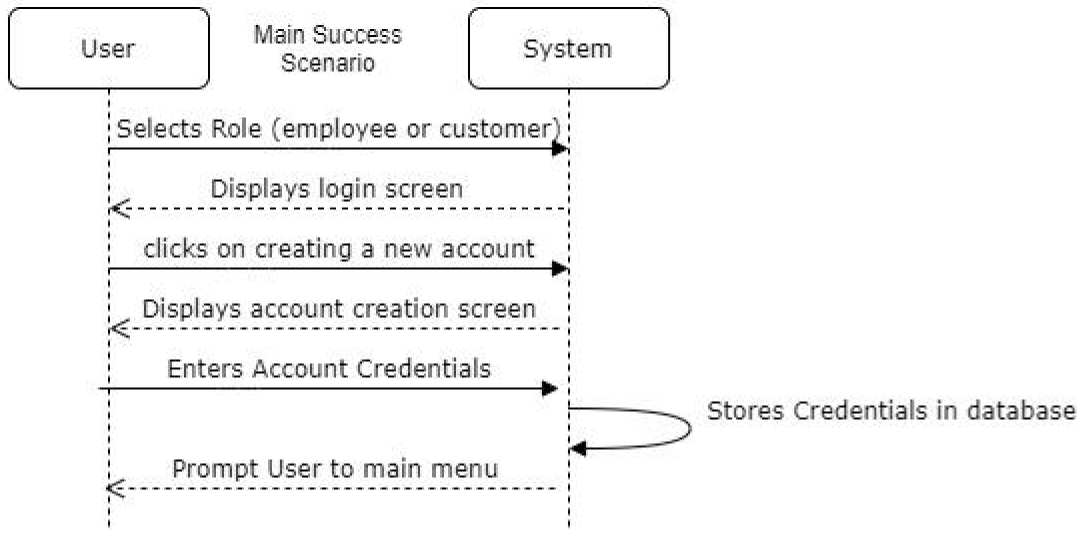

Employee
Alternate Success Scenario
System
Opens inventory
Displays inventory count
Displays inventory option to change
initiates inventory editor option
Edits Inventoy Count
Updates and Saves Inventory

The employee and the system work hand in hand in order to update and add existing menu items. Once the inventory system is opened, you will be able to view the current inventory count, and within there you will be able to select the option to open the menu editor. The menu editor will allow the employee to update the description of the menu item, and even the picture if necessary. This will be saved by clicking on a save button. As an alternate success scenario, after the inventory system has been opened, you can update the inventory count rather than the description of the menu item.

UC-13: Customer & Employee Login

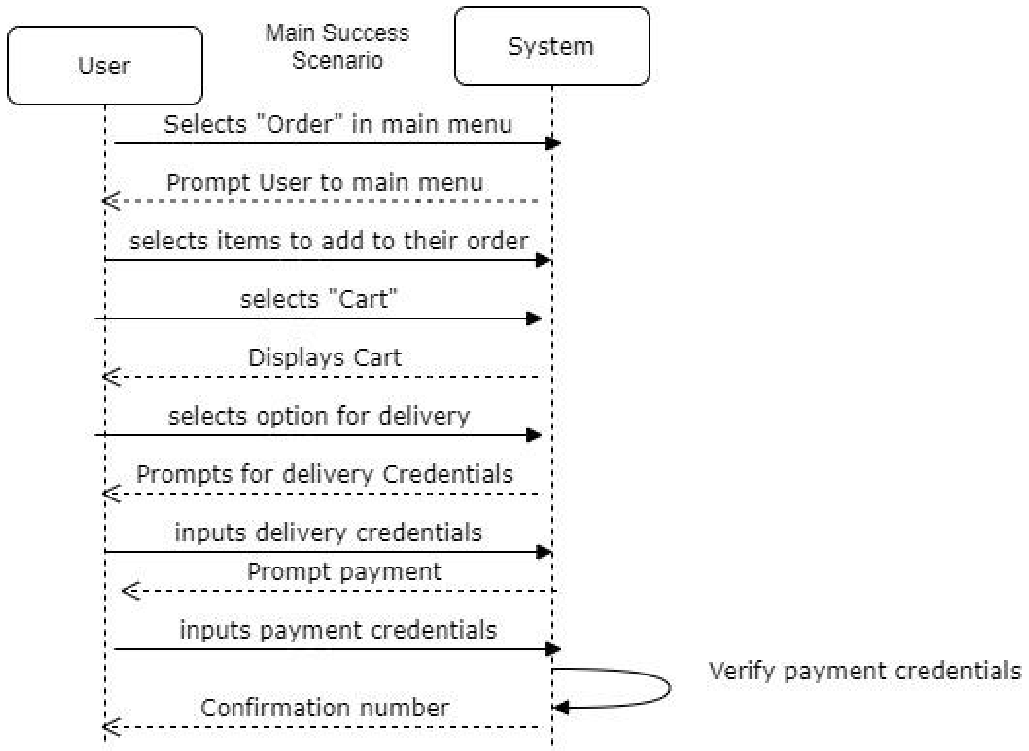


UC-13: Customer and Empoyee Login


User
Main Success Scenario
System
Selects Role (employee or customer)
Displays login screen
clicks on creating a new account
Displays account creation screen
Enters Account Credentials
Stores Credentials in database
Prompt User to main menu

Our system implements two different roles: customer and employee. The user selects their role and enters their login credentials on the login screen. The credentials are then verified by the firebase which contains a list of authenticated users. A second success scenario exists in the form of creating an account. The user selects their role which prompts a display screen. The user chooses to create an account instead of logging into an existing one. The account creation screen is displayed and the user enters account credentials. The credentials are stored in the database for that specific user and the user is then redirected to the main menu. The user can then follow the first main success scenario.

UC-14: View and Manage Customer Account and Order.

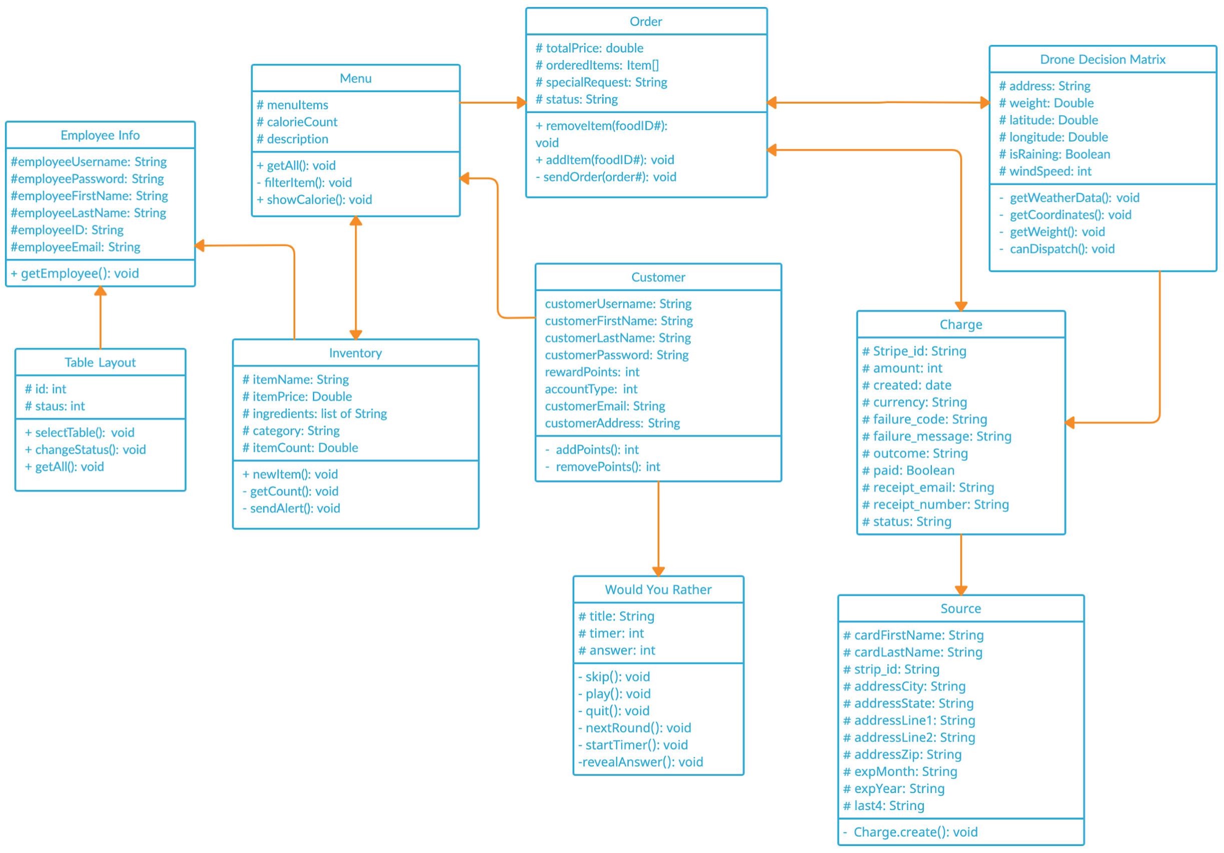

User
Main Success Scenario
System
Selects "Order" in main menu
Prompt User to main menu
selects items to add to their order
selects "Cart"
Displays Cart
selects option for delivery
Prompts for delivery Credentials
inputs delivery credentials
Prompt payment
inputs payment credentials
Verify payment credentials
Confirmation number

UC-14: View and Manage Customer Account and Order

To view and manage the customer's account and order, you must first login to the system. This would bring you to the customer portal that is customized to the specific customer. On the main menu, they can then choose to "order" which would allow the customer to make selections on foods they would like to purchase, adding it to their virtual cart. Once the order is done, you will be able to manage the order at the end. This would include entering delivery information if necessary (depending on if they are doing delivery or take out, or if their information is already not on file). After this is done, they will then be prompted to also enter their card information if it is not on file or they would like to add a new card. Once entering all necessary information, they are able to finish their order where they will receive a confirmation.

Section 8: Class Diagrams and Interface Specification

8.1: Class Diagram



8.2: Data Types and Operation Signatures

Class: Menu

Attributes:

- menuItems: List of food items that are available to the customer.
- calorieCount: Provides the calorie count for each individual item on the menu.
- description: String - Each menu item will have a description describing what the menu item is.

Methods:

- getAll() - The database retrieves all of the menu items.
- filterItem() - The database filters out menu items, according to dietary preferences.
- showCalorie() - The interface shows all of the caloric information pertaining to the menu item.

Class: Order

Attributes:

- totalPrice: double - The total price of all the items in the order.
- orderedItems: Item[] - The list of the food items in the car, picked out by the customer.
- specialRequest: String - A place for customers to comment on their order, making special requests that are not available in the user interface.
- status: String - Maintains the status of the customer's order.

Methods:

- removeItem(foodID#) - Removes food from the existing order; will take the food ID and quantity.
- addItem(foodID#) - Adds food items indicated by the customer via the menu interface, and will add it to the customer's cart.
- sendOrder(order#) - Used to send the order to the kitchen to be made by the employees.

Class: Customer

Attributes:

- customerUsername: String - The username associated with the customer's account
- customerFirstName: String - The first name of the customer.
- customerLastName: String - The last name of the customer.
- customerPassword: String - The password associated with the customer's account.
- rewardPoints: int - Tracks how many reward points the customer has accumulated.
- accountType: int - The type of account the user is.
- customerEmail: sting - The email associated with the customer and their account.
- customerAddress: string - The address associated with the customer and their account.

Method:

- addPoints() - Add an integer of reward points to the customer's balance.
- removePoints() - Removes an integer from the customer's balance.
- getPoints() - Gets the customer's reward balance.

Class: Employee Info - This class holds all of the basic information pertaining to the employee that is necessary for keeping track of their data.

Attributes:
- employeeUsername: String - This is the individual employee's self-made username to be able to have an account.
- employeePassword: String - This is the individual employee's self-made password to keep their profile protected.
- employeeFirstName: String - First name of the employee.
- employeeLastName: String - Last name of the employee.
- employeeID: String - This is an employee specific ID so that the system and manager are able to identify the employee's role and have the ability to edit and track necessary hours.
- employeeEmail: String - Email of the employee.

Methods:
- getEmployee() - This will display all the employees's information.

Class: Inventory

Attributes:
- itemName: String - The name of the item.
- itemPrice: Double - The price of each item.
- ingredients: list of String - A list of the different ingredients pertaining to one menu item.
- category: String - A string to keep track of what category a particular menu item belongs to.
- itemCount: Double - Keeps a tab on the number of items currently located in inventory.

Methods:
- newItem() - Allows the manager to add a new item to the inventory.
- getCount() - Shows the current inventory of a specific item.
- sendAlert() - Sends alert when inventory stock gets low.

Class: Table Layout

Attributes:
- id: int - The ID of a table.
- status: int - The current state/status of a table - 'green' is available, 'red' is occupied, and 'coral' is dirty

Methods:
- selectTable() - The customer selects a table.
- changeStatus() - The status of a table will change status upon selection.
- getAll() - The database retrieves all of the tables in the restaurant.

Class: Drone Decision Matrix

Attributes:
- address: String - address as imputed by the customer
- weight: double - weight of the order
- latitude: double - latitude of the customer
- longitude: double - longitude of the customer
- isRaining: boolean

- windSpeed: int

Methods:

- getWeatherData(): String - Gets the weather data from an external service.
- getCoordinates(String address): Double[2] - Finds the coordinates of a place using a street address and an online geocoding service. Used to calculate distance between the customer and the restaurant
- getWeight(): Double -
- canDispatch(): Boolean - This methods calculates if a drone can be used to deliver an order

Class: Charge

Attributes:

- Stripe_id: String - The unique ID of the charge
- amount: int - The amount being charged to the credit/debit card.
- created: date - The time and date that the transaction was enacted.
- currency: String - The type of currency being used.
- failure_code: String - The code associated with the reason for failure.
- failure_message: String - The message displayed when there is a failure.
- outcome: String - The outcome of the charge.
- paid: Boolean - Used to tell if the order has been paid for or not.
- receipt_email: String - The email address the receipt will be sent to.
- receipt_number: String - The unique receipt number.
- status: String - Associated with the status of the charge.

Class: Would You Rather

Attributes:

- title: String - The title of the game.
- timer: int - Numeric countdown
- answer: int - The answer to the would you rather question, Answer 0 or Answer 1.

Methods:

- Skip() - Will skip a would you rather question and initiate a new question
- Play() - Will initiate the would you rather game and provide a question
- Quit() - The customer is able to quit the game
- nextRound() - Will allow the customer to move on to another round
- startTimer() - Will start the timer for the question
- revealAnswer() - Will reveal randomly selected answer

Class: Source - This class will keep track of all of the details about the method of payment, if it was done by cash or card, and if the transaction is completed by card then the card information will be stored.

Attributes:

- cardFirstName: String - The first name of the person being charge

8.3: Traceability Matrix

	Software Classes									
Domain Concepts	Menu	Order	Customer	Employee Info	Inventory	Table Layout	Drone Decision Matrix	Charge	Would You Rather	Source
Customer Profile		X	X					X	X	X
Order Display		X	X							
Game Play			X						X	
Inventory Interface					X					
Alert Display					X		X			
Calorie Display	X	X	X							
Interface	X	X	X			X	X		X	

Employee Profile				X			X			
Payment System		X						X		X
Controller	X					X				
Reward System			X							
DB Connection		X		X	X		X			
Table Status				X		X				

<u>Customer Profile</u>

- Customer: Allows the customer to view and edit account-specific data.
- Charge: Allows customers to pay for their order.
- Would You Rather: Allows customers to play the dedicated drinking game.
- Source: Allows customers to save their payment method.

<u>Order Display</u>

- Order: Displays the customer's order with details on the items.
- Customer: Allows the customer to view and edit their order.

<u>Game Play</u>

- Customer: Allows only customers that have an account to play the game.
- Would You Rather: Customer interaction with the game is done through the Game Play.

<u>Inventory Interface</u>

- Inventory: Viewing and changing the inventory are done via the inventory interface.

<u>Alert Display</u>

- Inventory: Initiated by the inventory once an item stock is low.
- Drone Matrix: Initiated by the Drone Matrix if weather is deemed dangerous for drones to operate.

<u>Calorie Display</u>

- Menu: Displays the calorie count for items on the menu through the Calorie Display.
- Order: Displays total calorie count of the overall order.
- Customer: Allows customers to view the menu based on their dietary and calorie needs through the Calorie Display.

<u>Interface</u>

- Menu: The Menu is accessible through the interface.
- Order: Orders are placed through the interface.
- Customer: The customer portal is accessible through the interface.
- Table Layout: Table selection is done through the interface.
- Drone Decision Matrix: Data pertaining to the drone's function is displayed through the interface.
- Would You Rather: The game is accessible through the interface.

<u>Employee Profile</u>

- Employee Info: Contains Employee's personal information.
- Drone Decision Matrix: Allows only employees to access the drone decision matrix.

<u>Payment System</u>

- Order: The payment system is initiated once the order is confirmed.
- Charge: The system allows the user to pay for his order.
- Source: The system allows the customer to have a saved method of payment.

<u>Controller</u>

- Menu: Controller requires that the item be selected from the menu.
- Table Layout: Controller requires that a valid table is selected.

Reward System

-	Customer:	Reward points are saved on the customer’s profile.

DB System

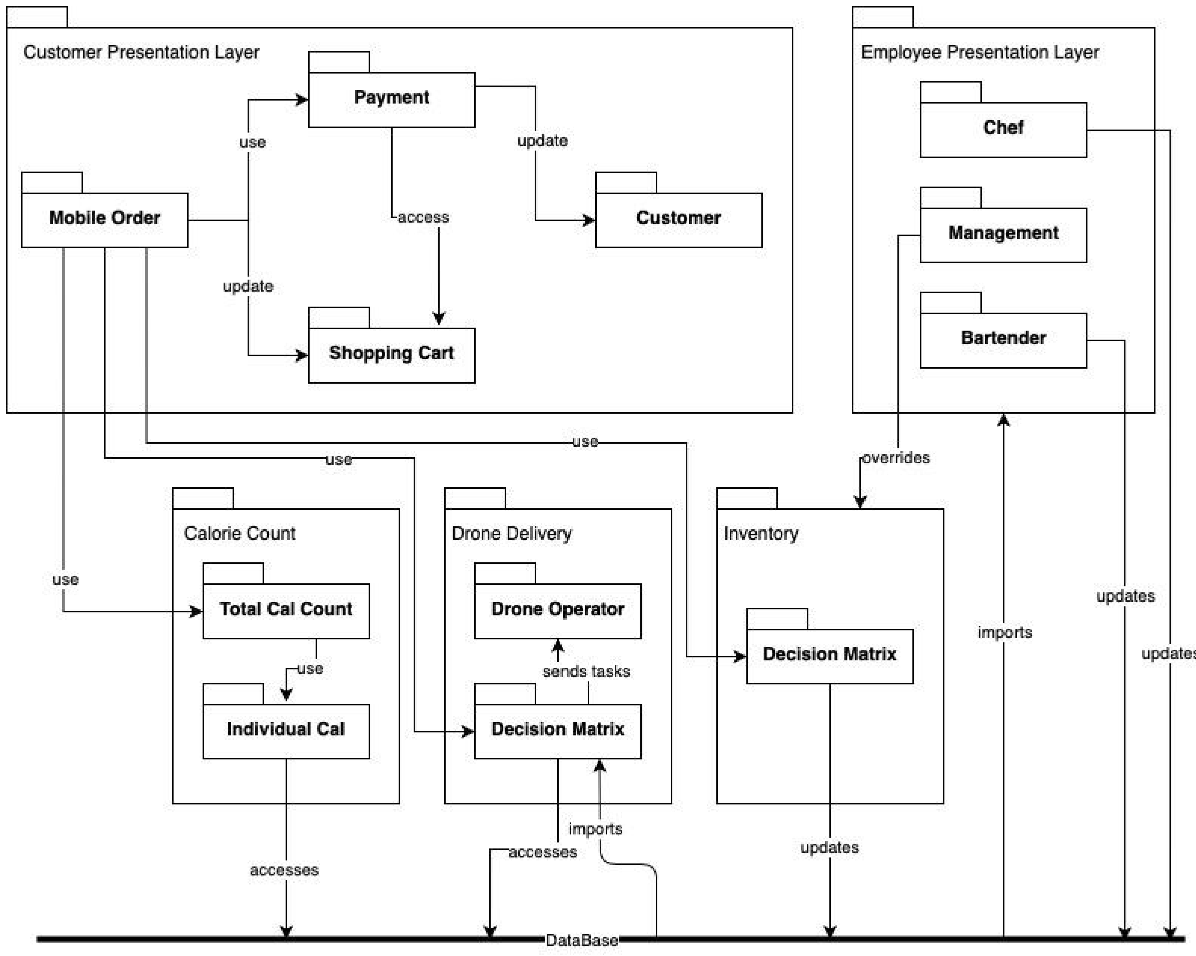
-	Order:	Orders will be sent to the database once the customer places their order.
-	Employee Info:	Stores employee’s personal information onto the database.
-	Inventory:	Stores Inventory items on the database.
-	Drone Decision Matrix:	Customer information that is used by the Drone Decision Matrix is stored in the database.

Table Status

-	Employee Info:	Allows Employees to change the status of a table.
-	Table Layout:	Allows users to select and view the status of a table

Section 9: System Architecture

9.1: Identifying Subsystems



The business component in the project comes under two umbrellas.

1. Under Inventory subsystem, the manager computer is able to access the Supplier Database. Within the supplier database is the inventory data: the cost of ordering an ingredient, cost of shipping and handling, and payment method. This part of the business is the backend where it shows the ordering and handling costs.
2. Under the App subsystem, customers are able to order the food. This part of the business is the customer facing where customers can order food and use their payment method to pay for the food.

How it works with each component:

Drone:

The drone section makes use of the app subsystem and the inventory subsystem. Customers ordering food will be allowed to choose the method of delivery if they are eligible for drone delivery. The decision of eligibility will be done in the background within the app and will access information from the inventory subsystem to inform itself.

Games:

The game section uses the Customer Presentation Layer. Customers who have created an account have the ability to access the Game System. Customers can play a game called “Would You Rather” where customers are given a series of questions and asked to make a decision on two options. One of the options is displayed by the system and customers who choose that option are asked to take a shot. All steps taken by the Game System are handled by the Customer Presentation Layer.

Inventory:

Inventory section integrates the Inventory system. Starting from the mobile order, it uses the Decision Matrix to inform itself about the new order. The “Decision Matrix” then accesses the Database (which is an external system) to access the initial values and then the “Decision Matrix” sends tasks to update those values with the new ones depending on the order. This then informs the other subsystems to make what it must, but the importance of the inventory section is so they can keep track of all the inventory and automatically order new inventory in case it's low.

Health Data:

The health data section integrates the calorie count system and the customer presentation layer. Starting from the mobile order, the calorie count system is used to calculate and display the caloric value for each menu item. The individual caloric values for the ingredients that comprise a menu item are retrieved from the Database(external system) and then summed. As the customer eliminates or substitutes ingredients in their order, the calorie count is updated accordingly.

Rest of Subsystems:

One last business logic is the overhead cost of running the kitchen, but none of our subsystems handle that. This part of the business logic comes under the manager/owner and their method of paying the overhead. Therefore this umbrella is not included in the business components for the project.

9.2: Architecture Styles

Our project's architectural style will be composed of a few common architectural styles. Primarily, the application will make use of the client-database model. The application client will read and write data to the database according to rules set by the application. Customers interact with the client to do things like order their food, play a game, or view the status of their delivery. Restaurant staff interact with the client to do things like clock in/out and view & manage orders. The database will store all these data points, and the application client can read them as requested. Clients can only interact with the database, and not directly to other clients.

The Super Foods application will make use of layered architecture to easily separate the features needed by customers from features needed by restaurant staff. This will simplify the design considerably and allow us to make changes to one layer without affecting the others. It also allows the different roles to interact with the application at different levels of abstraction. For example, while managers should be able to view and modify the inventory subsystem, other staff and customers should not see this feature in their application layer.

Lastly, the application will follow a reactive style architecture. Reactive systems are responsive (low latency), resilient (stays responsive in face of failure), elastic (stays responsive under varying workload), and message driven (pass messages asynchronously between components). The application will be composed of smaller components that will work as independently as possible in order to ensure responsiveness, resiliency, and elasticity.

9.3: Mapping Subsystems to Hardware

Our project will have one component that will run on hardware, as explained above: application clients. The many clients will connect and interact with the server. The server will manage the database and its components, and process requests from the clients. The clients will run as applications on mobile devices. Most modern smartphones will be able to use the application.

9.4: Connectors and Network Protocols

Our system will be having the client interact with the database to load and store data. To accomplish this, our group is using a Firebase database, which is a hosted database solution with full support for mobile development.

9.5: Global Control Flow

Executive Orderness: The SuperFoods main system is *event-driven*. Excluding the linear fashion of logging in, the user can take a variety of actions in whatever order they please (View menu, place an order, view order history, leave a review, play a game) but when breaking it up into subsystems, the process of ordering and the “Would You Rather?” Adult Game are *procedure driven* where the user only has a few paths to take, either continue or cancel/exit.

Time Dependency: The subsystem managing drone operation is the only subsystem that involves *time dependency*. The manager is able to set a time window for customers to place an order. When a customer includes their shipping address for a delivery order, the system will also prompt a request for the order arrival time. If the order arrival time exists within the time window set by the manager, then the system needs to calculate how long it will take for the kitchen to finish preparing the food and for the drone to deliver it. Subtract the order arrival time by those values and the system will know when to prompt the kitchen to start preparing the food and allocate a drone for the specific time of the delivery. Implementing time-dependency into the order and drone subsystems is paramount for a restaurant to deliver food effectively.
The “Would You Rather?” Adult Game subsystems will also be *time dependent*, constrained by another time window set by a manager. This is important when the restaurant is held up from closing because customers are playing the game and a manager is unavailable to turn the game off right away, the manager can be proactive by setting a time for the game to be locked from users automatically. Whenever a customer presses the next button (UC-10) the system will compare the real time with the manager's time window. If it is outside the window, the game will redirect to the main

menu. At every entrance to the main menu, the system will also disable the game button if the real time is outside the time window.

Concurrency: Since multiple clients can be running at any given time, all concurrency is handled by the application client and database. When the new data is loaded into the Firebase database, the application automatically updates

9.6: Hardware Requirements

Our SuperFoods application will be able to run on mobile devices such as smartphones and tablets that are supported by a network connection via Wifi, ethernet, or Mobile Data. The layout of the application will be identical for all device types and models. The application is not intended to be used on a computer, so there is no developed client or layout for laptops or desktops. The application will be compatible with the android operating system and able to run on a range of phones such as the Google Pixel and Nexus phones. A minimum storage capacity of 100 mb is required. Our application would require mobile devices to be at least operating in the android operating system “Android Pie”. Also to be able to run our application a minimum of a 5.6” phone screen is required with a resolution of 1080x 2220. A minimum recommended internet connection is 2.5 Mbps to run SuperFoods smoothly. For phone specifications we would require the phone to at least Qualcomm Snapdragon 835 processor, the phone to be equipped with a gyrometer and accelerometer, and wireless audio capabilities.

Section 10: Algorithms and Data Structures

10.1: Algorithms

Drone Delivery: Drone delivery will be using an algorithm to determine whether or not a user's order will satisfy the requirements for drone delivery. Our process check will take into account the user's pick-up location, the weather, as well as the weight of the order. This will make sure that all orders sent out by drone will be able to make it to the target location and back to the restaurant.

Game App: The Game App will not use any complex algorithms.

Inventory: The inventory portion of the application will not be using any algorithms.

Calories/Health Data: The calories portion will not be using any algorithms as it mainly involves the database and interface.

Menu/Order Systems and Customer Interface: The Enhancements to the menu and orders do not require any algorithms since the system will only be fetching information in the database to make a decision and will not make any complex calculations aside from summing the total price of an order.

10.2: Data Structures

Drone Delivery: Drone delivery will use simple data structures in the implementation of the code. We will be using a linked list to calculate the weight of the order based on the ingredients that are being used to make the dishes. This will allow us to efficiently look up and calculate the total weight of the entire order to determine whether or not drone delivery is available for the customer. We will use Firebase to hold all of our information for the linked list.

Game App: The game app will contain a Data Structure storing a list of "Would you Rather" images. After play() or nextround() the system will fetch a random image to be displayed.

Inventory: Inventory will be utilizing data structures in the implementation of the code. We will be using such arrays to be able to store all of the information pertaining to a specific ingredient. This would include the name and quantity of each item. We chose to use arrays because that will be the easiest way to organize and access the information. This also allows us to easily expand on the table, giving us room to add new ingredients to the inventory when necessary. We will be using the NoSQL database Firebase in order to hold all of our information, and sync between the users and the application.

Calories/Health Data: Calories will be utilizing data structures in order to store information such as caloric values corresponding to the menu items. The interface will then display that data to the interface.
There will also be an array containing the menu items that include certain food filters such as allergies and dietary restrictions to be displayed exclusively or sorted out from the menu.

Menu/Order Systems and Customer Interface: In order to readily fetch the most recent menu, Data Structures need to include menu items from the menu editor including a variable corresponding to each variable that, determined by the inventory, will be checked before a menu item can be displayed.

10.3: Concurrency

Drone Delivery: The Drone Delivery App will not use multiple threads.

Game App: The Game App will not use multiple threads.

Inventory: The inventory app will not be using any concurrency.

Calories/Health Data: The calories app will not be using any concurrency.

Menu/Order Systems and Customer Interface: When an order is purchased, the system will need to update the inventory. And in turn, when a customer accesses a menu, the inventory may be empty for a particular item that had just been ordered. Threads using the menu and menu editor have to be refreshed.

Section 11: User Interface Design and Implementation

Drone Delivery: The interface of the drone delivery system prompts the user on what type of delivery they prefer. The system will then determine whether or not that delivery option is available to the customer. The customer will have a confirmation page of their delivery/pick-up status associated with their order.

Game App: We stayed true to the original design found in Report 1, to provide an easier form of displaying our “Would you Rather” statements, we stored the statements on images for them to be displayed.

Inventory: The core functionality of the inventory system will remain the same as proposed in the original report (Report 1). The formatting also will not be changed and will be implemented in an organized fashion, keeping all ingredients in specific categories. Originally, there was little documentation on the way the inventory interface was going to look.

Calories/Health Data: The essential functions outlined in the original report have not been changed and will be implemented into the main interface. Caloric values will be easily accessible and displayed alongside the menu items.

Turbo-Yum Enhancements: Compared to our images in Report 1, we stayed true to the straight forward style and display of our app, we changed the initial home page into a drop-down bar at the bottom of the page for the customer to move easier within parts of the app.We decided at a phone number to the Log in as a easier way for the system to recognize & authenticate the account and employee status, it acts as a virtual address for the account

Section 12: Design of Tests

Drone Delivery:

Test Case Identifier: TC - 1 **Use Case Tested:** UC - 1 **Pass/Fail Criteria:** The test will pass if there is a valid address inputted into the specified places. The test will fail if the address inputted is unknown or invalid. **Input Data:** Customer Address	
Test Procedure	**Expected Result:**
Step 1: User inputs address into the delivery section.	The system will notify the customer weather or not their address is valid.

Test Case Identifier: TC - 2 **Use Case Tested:** UC - 2 **Pass/Fail Criteria:** The test will pass if the operator inputs a drone's status and the status updates in the system. The status includes the battery level and any issues.The test will fail if the the operator inputs a drone status and the status does not update **Input Data:** Drone Status	
Test Procedure	**Expected Result:**
Step 1: Operator inputs the drone's status.	The system will update and show that the drone is either on delivery or in the restaurant.

Test Case Identifier: TC - 3 **Use Case Tested:** UC - 3 **Pass/Fail Criteria:** The test will pass if the user can track their order using the App. The test will fail if the user cannot track their order. **Input Data:** N/A	
Test Procedure	**Expected Result:**
Step 1: User logs in to the app Step 2: User clicks, "Track my order"	The system will update and show the drone's location throughout the order delivery.

Test Case Identifier: TC - 4
Use Case Tested: UC - 4
Pass/Fail Criteria: The test will pass if it is able to correctly determine if drone delivery can be done. The test case fails if the program cannot read the data required to perform the calculations.
Input Data: Address of delivery (by the user). Weather, map, and order data fetched automatically.

Test Procedure	**Expected Result:**
Step 1: User inputs delivery data	The system will keep track of the address.
Step 2: Program fetches weather and map date from API and fetches weight data from the database	The program will store all of the data from API for later use
Step 3: Automatically calculate drone eligibility	The system will combine all the data and output a decision if the drone should be used or not.

Game App:

Test Case Identifier: TC - 5 **Use Case Tested:** UC - 5 **Pass/Fail Criteria:** The test will pass if the user can play the game and earn rewards if they have an account. The test will fail if the game crashes, or the user doesn't earn any rewards if they have an account. **Input Data:** N/A	
Test Procedure	**Expected Result:**
Step 1: User logs in to their account, or continues as guest Step 2: User starts the game Step 3: User goes through the tutorial, or skips the tutorial Step 4: Plays the game Step 5 (Optional): If the user has an account, check to see if they received any rewards after winning.	Users can play the game with no hassle. Users can log in to their account and earn rewards.

Inventory:

Test Case Identifier: TC - 6 **Use Case Tested:** UC - 6 **Pass/Fail Criteria:** The test will pass if the system properly orders the new inventory and updates the database when the shipment arrives. The test will fail if it orders the wrong inventory item, if it updates incorrectly, or if it updates before Manager approval. **Input Data:** N/A	
Test Procedure	**Expected Result:**
Step 1: Manager proceeds to click the inventory tab. Step 2: The tab shows all inventory items that need resupply. Step 3: Manager clicks the order button and the order is placed. Step 4: System updates inventory when shipment is fulfilled and manager approves the purchase.	The manager can order new shipment for inventory which will auto-update when the shipment arrives.

Test Case Identifier: TC - 7 **Use Case Tested:** UC - 7 **Pass/Fail Criteria:** The test will pass if the system properly displays updated inventory count, and categorizes them accordingly. The test will fail if the information is not loaded from the database. **Input Data:** N/A	
Test Procedure	**Expected Result:**
Step 1: Employee proceeds to click the inventory tab.	The employee will see the updated inventory count as well as an organized, color-coded interface.

Test Case Identifier: TC - 8 **Use Case Tested:** UC - 8 **Pass/Fail Criteria:** The test will pass if a notification is displayed indicating that there is low stock of a specific item in inventory. The test will fail if an item is considered to be low stock and the manager is not being notified. **Input Data:** N/A	
Test Procedure	**Expected Result:**
Step 1: Manager proceeds to click the inventory tab. Step 2: An alert will automatically be displayed if there is an item considered to be lowstock.	The manager will receive a pop up notification where they are able to click “okay” if they acknowledge the low stock alert.

Test Case Identifier: TC - 9 **Use Case Tested:** UC - 9 **Pass/Fail Criteria:** The test will pass if the database accurately keeps track and updates, inventory count. The test will fail if the database does not update the inventory count accordingly. **Input Data:** Inventory Count	
Test Procedure	**Expected Result:**
Step 1: Manager updates inventory count according to the shipment delivery	The database will update inventory count properly in the database.

Calories/Health Data:

Test Case Identifier: TC - 10
Use Case Tested: UC - 10
Pass/Fail Criteria: Pass if the calorie count displayed is equal to the sum of the calorie counts of all menu items in the order, taking account of quantity as well. The test will fail if the calorie count is incorrect or nonexistent.
Input Data: N/A

Test Procedure	**Expected Result:**
Step 1: Customer adds a menu item to the cart	The total caloric values of all current menu items are displayed on the menu and at checkout

Test Case Identifier: TC - 11
Use Case Tested: UC - 11
Pass/Fail Criteria: The test will pass if the system properly toggles between the two menus. The test will fail if the menus are not being displayed when prompted to by the customer.
Input Data: N/A

Test Procedure	**Expected Result:**
Step 1: Customer toggles menu to show caloric values under each menu item.	The caloric values will be displayed above the menu item name once the toggle button is clicked. If the toggle button is clicked again, caloric values cannot be seen.

Menu / Order System and Customer Interface:

Test Case Identifier: TC - 12 **Use Case Tested:** UC - 12 & UC - 13 **Pass/Fail Criteria:** The test passes if the user and an employee can create an account and can logout and log back in. The test fails if the account isn't created or if the user or employee cannot log back in. **Input Data:** Customer and Employee information	
Test Procedure	**Expected Result:**
Step 1: Person goes to app and creates an account Step 2: Person inputs information depending on if they are a user or an employee Step 3: After verifying their email, they log out of their account Step 4: They log back in	System uses phone number to find the account, check if the username and email match the account database. Then the system checks user employee status and displays buttons accordingly.

Test Case Identifier: TC - 13 **Use Case Tested:** UC - 14 **Pass/Fail Criteria:** The test will pass if the customer is able to view all of the required information including rewards, past/current orders, reviews, and card information. The test will fail if at least one of the sections are unable to to be viewed properly or are not being displayed. **Input Data:** Customers must have placed an order to view past/current orders, must enter card information to be able to view the information pertained to that, or have earned rewards to view reward status.	
Test Procedure	**Expected Result:**
Step 1: User sets quantity and adds menu item to cart. Step 2: User clicks shopping cart button to view Order. Step 3: Customer views account information in the Customer Portal.	System displays a list of selected menu items and, calorie count, total price. System will also display reward status, past/current orders, and card information.

Test Case Identifier: TC - 14 **Use Case Tested:** UC - 15 **Pass/Fail Criteria:** The test will pass if the user can log in and redeem their points for a coupon for their next order. The test will fail if they are unable to redeem their points or if the

system does not reset their points. **Input Data:** User Card information	
Test Procedure	**Expected Result:**
Step 1: User logs in to their account Step 2: User goes to redeem their points Step 3: User clicks “Redeem Points for Coupon” Step 4: User uses coupon for a new order purchase	System resets their points after they redeem it. Then successfully uses the new coupon for the new order.

Test Case Identifier: TC - 15 **Use Case Tested:** UC - 16 **Pass/Fail Criteria:** Test will pass if the manager is able to put a hold on the deliveries coming in at a specific time. Test will fail if customers are able to place orders after desired end time set by the manager. **Input Data:** Time and Order	
Test Procedure	**Expected Result:**
Step 1: Manager sets a time that deliveries Step 2: Customer places an order that needs to be delivered	System will either tell the customer that their order has been placed or their order cannot be placed because it is past the time for delivery.

Test Case Identifier: TC - 16 **Use Case Tested: UC - 17** **Pass/Fail Criteria:** Test will pass if employees are able to properly view table layout, adn view available tables based on colors. The test will fail if the table status is not being updated properly or not being displayed to the employee. **Input Data:** Table Status	
Test Procedure	**Expected Result:**
Step 1: Employee looks at the table status via the Employee Portal.	System displays the table status, categorized by color -- color indicating whether or not the table is available, taken, or dirty.

Section 13: History of Work, Current Status, and Future Work

	Demo 1						Demo 2			
Group	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
	2/21 - 2/27	2/28 - 3/6	3/7 - 3/13	3/14 - 3/20	3/21 - 3/27	3/28 - 4/3	4/4 - 4/10	4/11 - 4/17	4/18 - 4/24	4/25 - 5/1
Group 1	UC-1	UC-1	UC-4	UC-4	UC-4	Demo Week	UC-2	UC-2	UC-3	Demo Week
Group 2	UC-5	UC-12	UC-13	UC-14	UC-14	Demo Week	UC-15	UC-16	UC-17	Demo Week
Group 3	UC-7	UC-7	UC-7	UC-9	UC-9	Demo Week	UC-6	UC-8	UC-8	Demo Week
Group 4	UC-11	UC-11	UC-11	UC-11	UC-11	Demo Week	UC-10	UC-10	UC-11	Demo Week

Future Work:
In the future, we wish to continue working on the App enhancements. We also wish to use a drone and integrate it within the project for a full demo. We also wish to work on increased security since there is people's information saved within the database. Hopefully, either starting our own restaurant or selling our product to other restaurants that wish to utilize our project to get revenue off of it.

Section 14: References

- TurboYums Project
  https://turboyums.github.io/SEWebsite
- The Coding Train
  Author: The Coding Train
  Title: 8.1: What is HTML? - p5.js Tutorial
  Link:
  https://www.youtube.com/watch?v=URSH0QpxKo8&list=PLRqwX-V7Uu6bI1SlcCRfLH79HZrFAtBvX&index=1
- The Reactive Manifesto
  Author: Jonas Bonér, Dave Farley, Roland Kuhn, and Martin Thompson.
  Title: The Reactive Manifesto
  Link: https://www.reactivemanifesto.org/

Section 15: Project Management

Subgroups, Members, and Members' Skills

Sub-Groups	Functionalities	Members	Skills
Group 1	Drone/Driven Delivery System	Keith Lo	C++, Python, Java, Solidworks, Impact
		Guilherme Silva	C++, Python, Java
Group 2	Tablet Games Turbo-Yums Enchantments	Rawad Sayah	C, Java, UX/I Design, Gamer
		Gian-Soren Morici	Linux, C++, Management, Creativity
		Brianna Solano Aguilar	C++, Java, SolidWorks, Arduino, Raspberry Pi
Group 3	Inventory	Saurabh Bansal	C++, Python, Java
		Robert Kulesa	Linux, C, Java, Python, Git/Version Control
		Lindsay Wisner	C++, Design, AutoCAD, SolidWorks, Team Management
Group 4	Calories/Health Data	William Basanaga	C++, Java, Python
		Jeremy Kim	Java, C++, AutoCad
		Justin Chan	C++, Java

Used Cases

Use Cases	Rob	Justin	Will	Lindsay	Kenji	Saurabh	Keith	Brianna	Rawad	Jeremy	Soren
UC - 1	0%	20%	0%	0%	40%	0%	40%	0%	0%	0%	0%
UC - 2	0%	20%	0%	0%	40%	0%	40%	0%	0%	0%	0%
UC - 3	0%	20%	0%	0%	40%	0%	40%	0%	0%	0%	0%
UC - 4	0%	20%	0%	0%	40%	0%	40%	0%	0%	0%	0%
UC - 5	0%	0%	0%	0%	0%	0%	0%	33%	33%	0%	33%
UC - 6	33%	0%	0%	33%	0%	33%	0%	0%	0%	0%	0%
UC - 7	33%	0%	0%	33%	0%	33%	0%	0%	0%	0%	0%
UC - 8	33%	0%	0%	33%	0%	33%	0%	0%	0%	0%	0%
UC - 9	33%	0%	0%	33%	0%	33%	0%	0%	0%	0%	0%
UC - 10	0%	40%	20%	0%	0%	0%	0%	0%	0%	40%	0%
UC - 11	0%	40%	20%	0%	0%	0%	0%	0%	0%	40%	0%
UC - 12	0%	0%	0%	0%	0%	0%	0%	33%	33%	0%	33%
UC - 13	0%	0%	0%	0%	0%	0%	0%	33%	33%	0%	33%
UC - 14	0%	0%	0%	0%	0%	0%	0%	33%	33%	0%	33%
UC - 15	0%	0%	0%	0%	0%	0%	0%	33%	33%	0%	33%
UC - 16	0%	0%	0%	0%	0%	0%	0%	33%	33%	0%	33%

UC - 17	0%	0%	0%	0%	0%	0%	0%	33%	33%	0%	33%

Report 3 Breakdown

Topics	Rob	Justin	Will	Lindsay	Kenji	Saurabh	Keith	Brianna	Rawad	Jeremy	Soren
Section 1	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 2	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 3	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 4	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 5	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 6	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 7	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 8	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 9	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 10	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 11	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 12	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 13	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 14	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%
Section 15	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%	9.09%

How we Worked as a Group

The group coordinated with each other and set meetings that accommodated everyone's schedule. Websites like when2meet provided a method to apply everyone's schedule for a set amount of days onto a table format. This allowed for easy selection of a timeframe with all members. Applications like discord are used to have project-related conversations divided by sections such as groups, ideas, and announcements. Discord also allowed us to have a platform to conduct meetings through voice and video calls. Meetings are conducted once a week with everyone to coordinate the direction of where the project is heading. Meetings are also conducted between respective groups as needed by members. Documentations for each meeting and reports are stored onto Google Docs, separated into their respective folders such as by group number, large meetings, and reports.

The report was done equally as a collective group. A meeting is scheduled such that every member of the project meets on Discord and completes the report together. Each subproblem group is in charge of describing and writing their system requirements. The user interface is completed together as a project team to incorporate everyone’s ideas. This gives us a foundation such that no conflicting problems occur when all subproblems are brought together. Upon completion of the report, everyone conducts a review of the report and provides comments. The comments are taken into consideration and incorporated into the final product. A final review of the report is conducted and later on submitted.

Meetings are objective-oriented such that they can be as productive as possible. During our initial week, we were able to create the basic framework of our project design. Subgroups were created and members were assigned to each group based on member’s interests. Meetings are conducted between each sub-group and each group sets objectives for themselves. Documentation for each meeting conducted for each sub-group is stored in the goal drive and contains the progress and status of the group. These documentations can be viewed by all members so they can view the activities of the other groups. Code created by the group will be stored in GitHub such that everyone has access to it.