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

Super Foods

Report #2 Part 3
Submission date: March 12, 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 8: Analysis and Domain Modeling

8.1: Conceptual Modeling

8.1.1: 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
TurboYum Enhancements	**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	InterfaceZ
	R18: Allows customers to leave a review(words+stars)	D	Interface

8.1.2: 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

8.1.3: 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
	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.

8.1.4: Traceability Matrix

Used Cases	**PW**	**Domain Concepts**										
		Controller	**Order Display**	**Game Play**	**Inventory Interface**	**Alert Display**	**Calorie Display**	**Employee Profile**	**Customer Profile**	**Payment System**	**Table Status**	**Interface**
UC - 1	**4**											**X**
UC - 2	**3.66**	**X**										
UC - 3	**3.2**	**X**						**X**				**X**
UC - 4	**5**	**X**				**X**		**X**				**X**
UC - 5	**4**		**X**						**X**			**X**
UC - 6	**4.33**					**X**		**X**				**X**
UC - 7	**3**			**X**								
UC - 8	**3**											**X**
UC - 9	**2.8**			**X**								

UC - 10	3.5			X								
UC - 11	5			X								
UC - 12	3.33				X							
UC - 13	4.33				X							
UC - 14	3.33				X							
UC - 15	2				X	X						
UC - 16	4.2						X					
UC - 17	4.5		X				X					
UC - 18	3.5	X					X					X
UC - 19	4.7				X		X					
UC - 20	3						X					X
UC - 21	4								X			
UC -	3.7		X									

22	1											
UC - 23	2								X			
UC - 24	2.33								X			X
UC - 25	3							X				X
UC - 26	2.5		X						X			
UC - 27	1								X			X
UC - 28	3.67									X		X
UC - 29	3							X				
UC - 30	4.33							X				
UC - 31	5										X	X
UC - 32	2.83							X	X			X
UC - 33	2.7											X

Drone/Driven Delivery Service

- **UC-1: Data Input -** Data is being inputted by the customer via the interface.
- **UC-2: Check Battery Level -** The battery level is being checked via the control which has access to the individual levels.
- **UC-3: Drone Tracking -** The controller tracks the drones, and displays them on the interface based on location. Only the employee has access to this specific feature tracking all drones.
- **UC-4: Track Emergencies -** The controller monitors all the drones, and alerts the employee if there are any issues regarding drone delivery. This is then displayed on the interface, and only available to the employees.
- **UC-5: Order Tracking -** In the customer profile, they are able to track their existing order via their order display which also utilizes the interface within the application.
- **UC-6: Decision Making -** The alert display sends to the drone controller which drones to use

Adult "Would You Rather" Game App

- **UC-7: Start the Game -** Allows the customer to start the game from the game menu via Game Play
- **UC-8: Tutorial -** Uses Game Play to allow the customer to migrate through tutorial slides at their command.
- **UC-9: Play -** Allows the customer to leave the tutorial and play the game via Game Play
- **UC-10: Next -** Allows the customer to skip a "Would You Rather?" question if the was repeated via Gameplay
- **UC-11: End -** Using Gameplay the customer is able to end the game and return to the main menu during a "Would You Rather?" statement

Inventory:

- **UC-12: Shipment Deliveries -** Shipment Deliveries use the inventory interface to allow managers access to viewing delivery status.
- **UC-13: View Inventory Count -** To view the inventory count, the inventory interface is used as a display mechanism.
- **UC-14: Inventory Display -**The inventory display utilizes the inventory interface to categorize the product in the inventory system.
- **UC-15: Low Stock Alert -** Low Stock Alert utilizes the inventory interface and alert display to get an alert about low inventory based on what information is held in the database.

Calories/Health Data

- **UC-16: Displaying Menu Item:** The calorie display provides calorie information for individual menu items

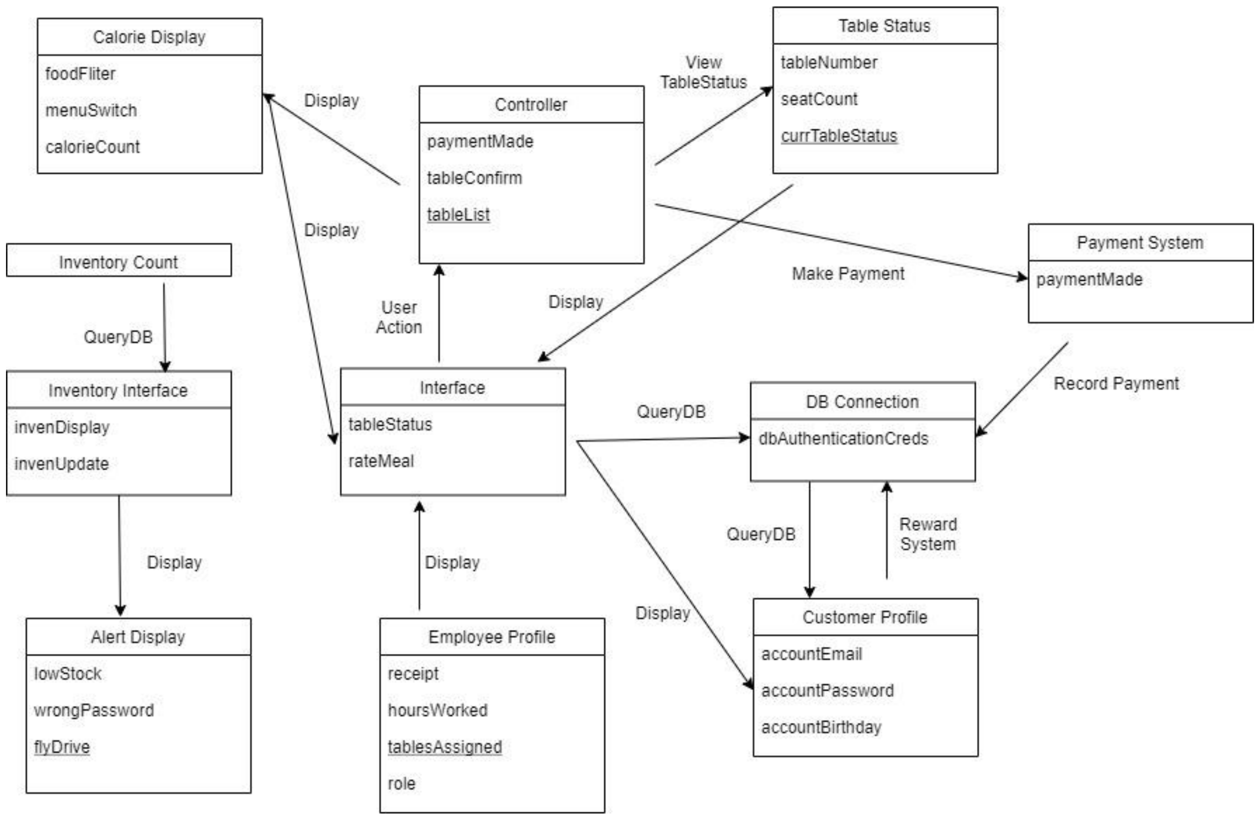
  UC-17: Display Total Calories: The total calories are available on the menu and at order display
- **UC-18: Switch Menu:** The toggle to switch is done by the customer through the interface
- **UC-19: Add New item:** Allows employees to add new ingredients and their caloric values through the inventory display

- **UC-20: Display Food Filters:** Allows customers to limit the display of menu items containing/not containing certain ingredients according to dietary restrictions

Turbo Yum Enhancements

- **UC-21: Account creation-** The customer is able to create an account and add information such as username, birthday, password, and email to the Customer Profile.
- **UC-22: Order-** When consumers view and manage their order, it's displayed on the order display.
- **UC-23: Reward Display-** A customer may earn rewards and it is done while the customer is paying for their meal.
- **UC-24: Point Redemption System-** A customer may redeem rewards if they have enough points and it is done while the customer is paying for their meal.
- **UC-25: Menu Editor-** In order to change the menu, a manager must interact with the application and the display using the interface.
- **UC-26: History Display-**Uses Customer Profile to allow customers to view past orders that were made through their account and displays them in the interface
- **UC-27: Review Menu-** Allows the customer to leave a star rating and comments on different categories therefore customer profile and interface are involved
- **UC-28: Purchase Menu-** Allows customers to input card information and complete a transaction when placing an order via the payment system
- **UC-29: Stop Delivery-** Allows managers to select a time window that will stop customers from selecting delivery when ordering through their employee profile which contains those privileges, and the interface to display the changes.
- **UC-30: Employee Creation-** Managers are able to create employee accounts and grant benefits therefore employee profiles are involved.
- **UC-31: Tables Display-** Displays the table layout using the interface to the host. This layout is color-coded through the Table Status to inform hosts of status.
- **UC-32: Customer and Employee Login-** Any user must log in order to access their accounts, so customer and employee profiles are involved, in addition to this user login by typing on the screen and reading the information displayed by the interface.
- **UC-33: Access Menu-** Lets users view the menu, food and beverage selection, and the reviews made from other users through the interface

8.1.5: Domain Model Diagram



8.2: System Operation Contracts

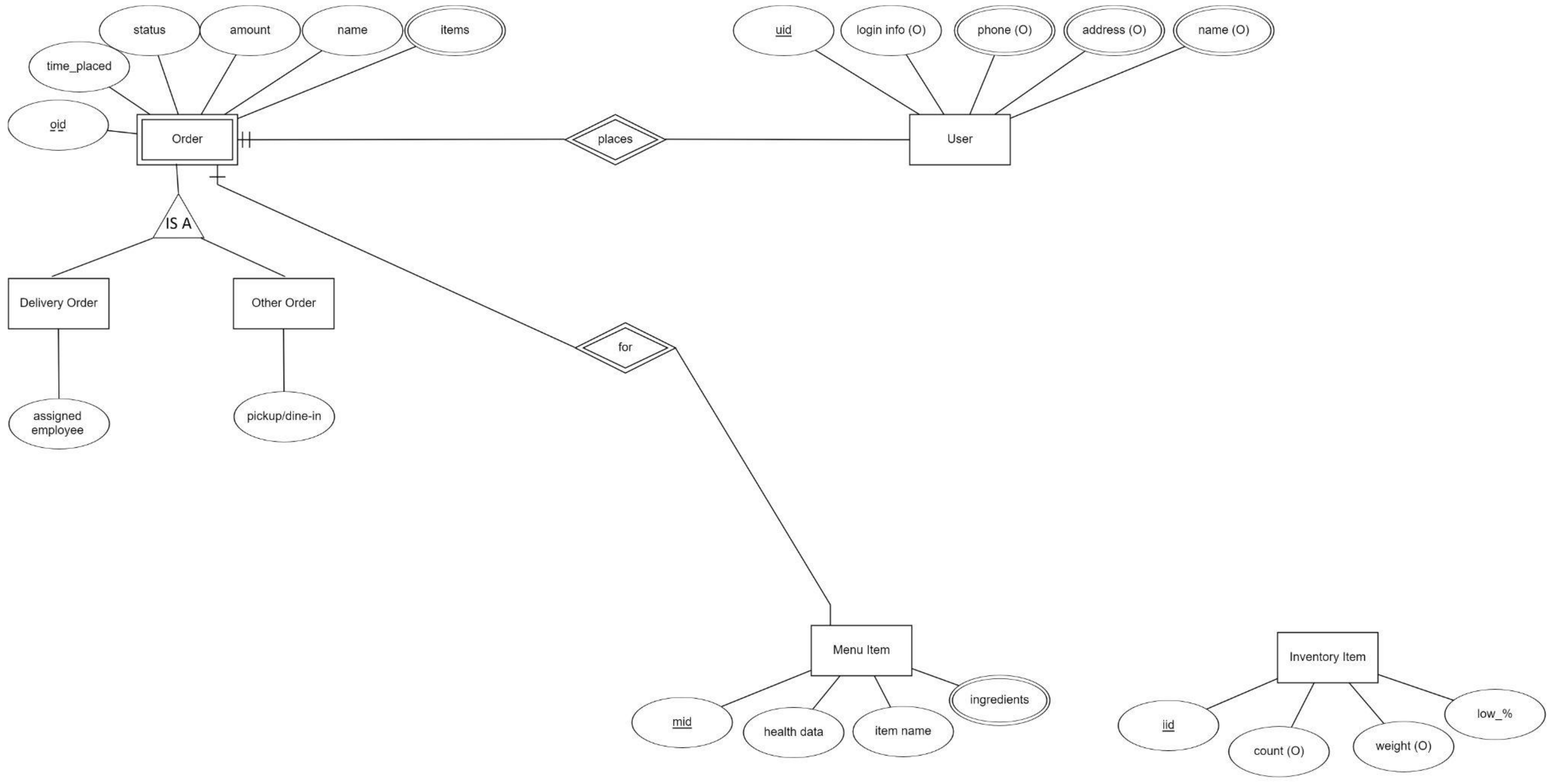
Operation	Decision Making
Use Case	UC-6
Preconditions	Customer must place an order Customer must enter address being delivered to
Postconditions	Customer will have order confirmation after processing the payment

Operation	Order Display
Use Case	UC-22
Preconditions	Customer logs into application Customer must click order from the drop-down menu
Postconditions	Customer is able to view and manage their order

Operation	Customer and Employee Login
Use Case	UC-32
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)

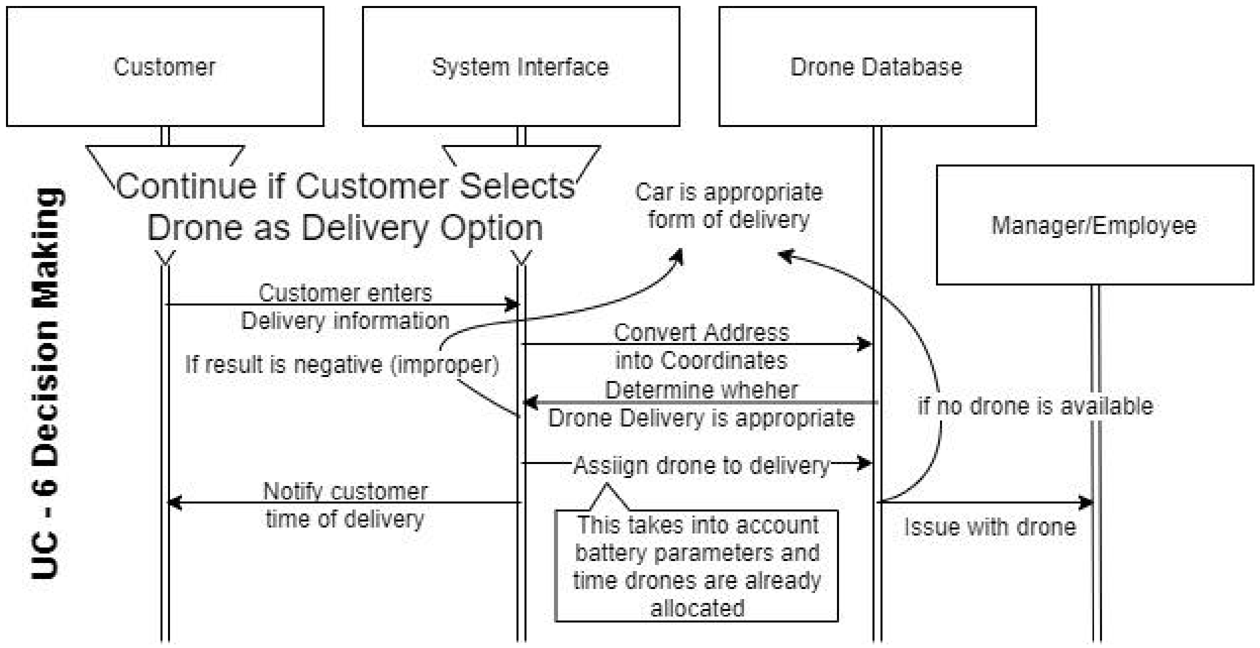
Operation	Accessing Menu
Use Case	UC-33
Preconditions	Customer must log in to the system using their credentials or chose an option as a guest Customer must select Menu from the drop-down bar
Postconditions	The user is shown the available menu

8.3: Data Model



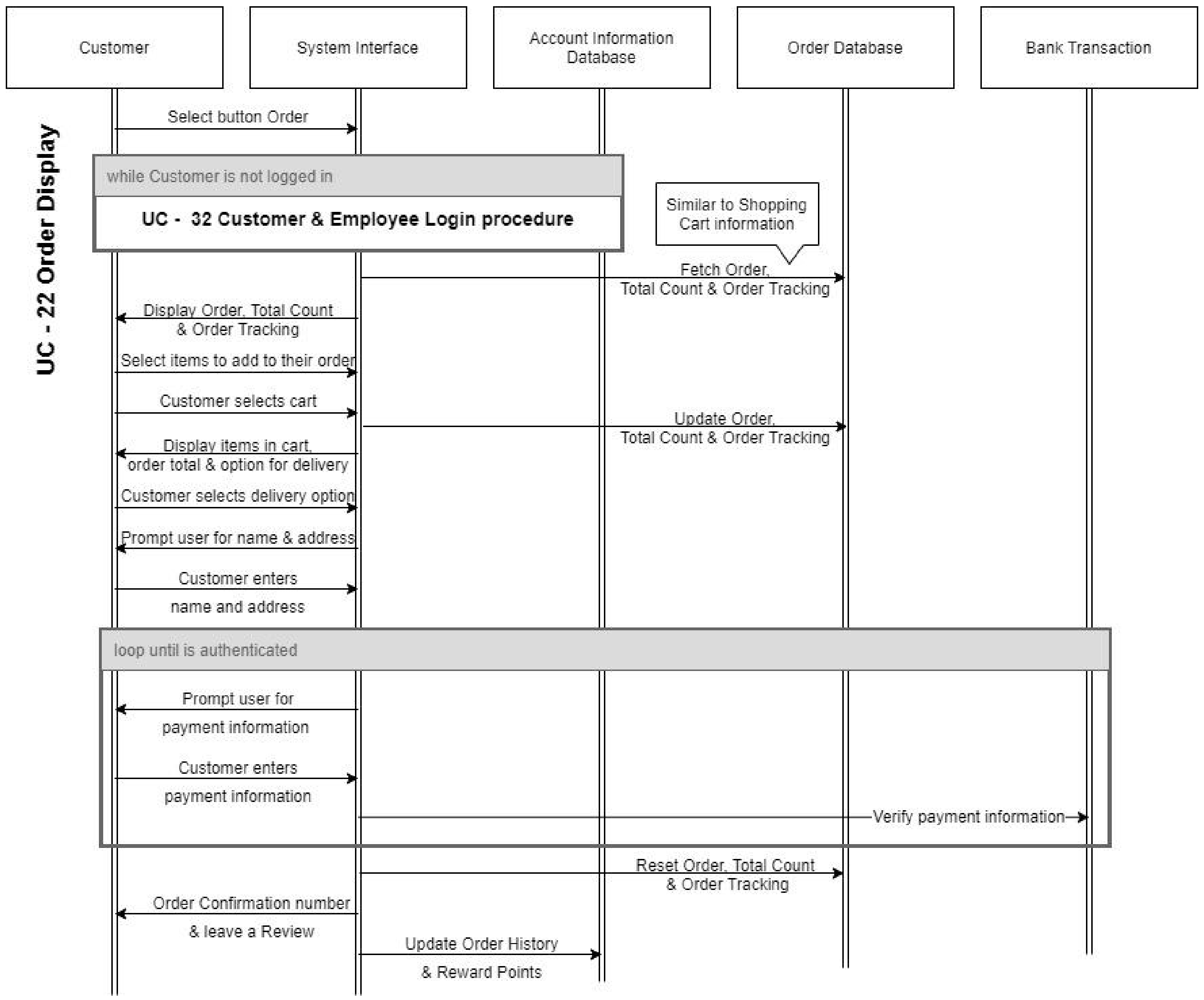
Section 9: Interaction Diagrams

UC-6: Decision Making


Customer
System Interface
Drone Database
Manager/Employee
UC - 6 Decision Making
Continue if Customer Selects Drone as Delivery Option
Car is appropriate form of delivery
Customer enters Delivery information
If result is negative (improper)
Convert Address into Coordinates
Determine wheher Drone Delivery is appropriate
if no drone is available
Assiign drone to delivery
Notify customer time of delivery
This takes into account battery parameters and time drones are already allocated
Issue with drone

Decision Making refers to the system's decision on whether a drone is optimal for the delivery method, when the user inputs their delivery information for their order, the system will perform some operations to decide the best form of delivery, it does this by transferring the address to a coordinate-based system. By calculating the distance and the timestamp of the order, the system will determine if a drone can be allocated for a timeslot in order to deliver the order. The system will notify the manager and drones accordingly and return the order confirmation order to the customer.

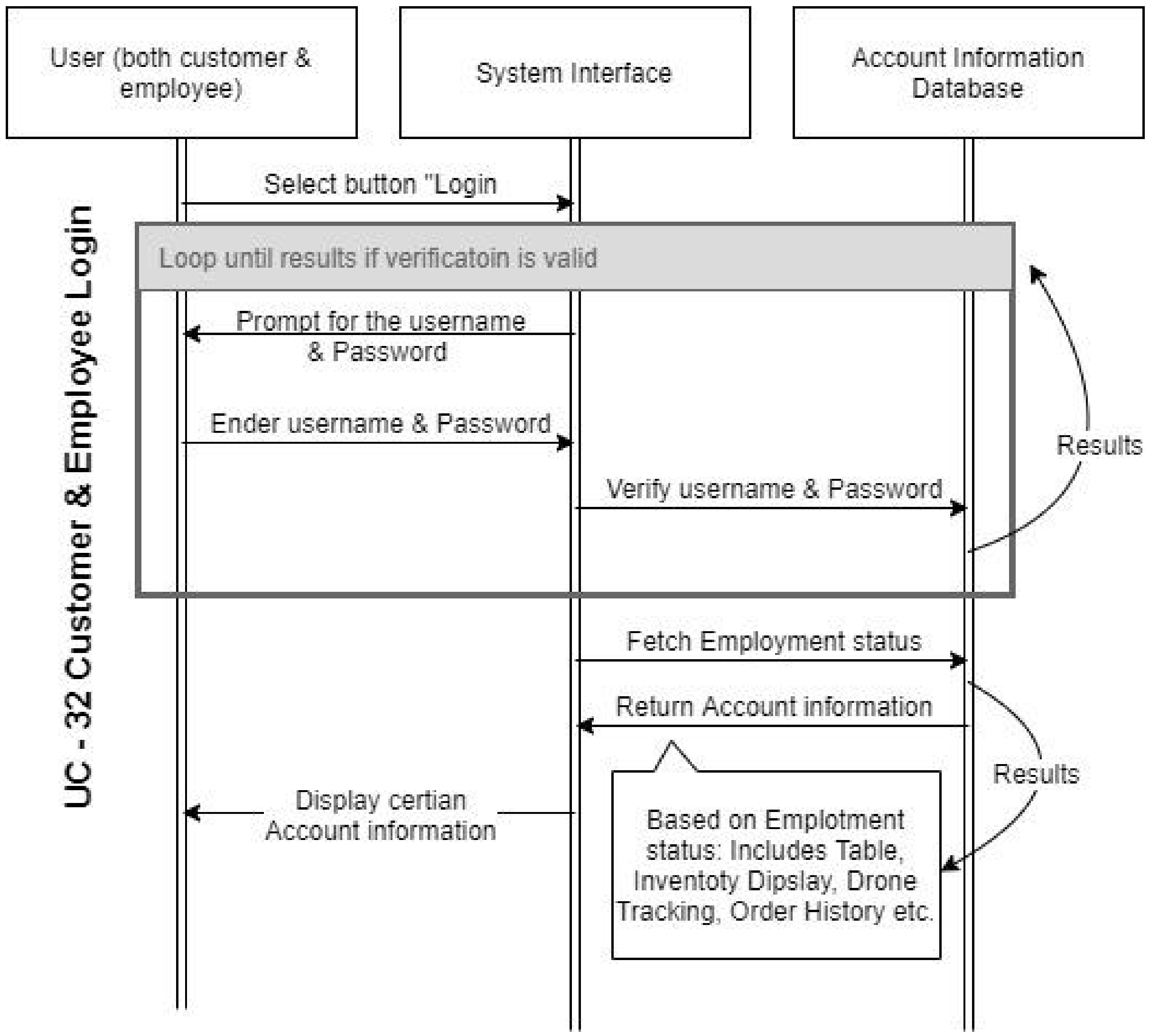
UC-22: Order Display



Customer
System Interface
Account Information Database
Order Database
Bank Transaction
UC - 22 Order Display
Select button Order
while Customer is not logged in
UC - 32 Customer & Employee Login procedure
Similar to Shopping Cart information
Fetch Order, Total Count & Order Tracking
Display Order, Total Count & Order Tracking
Select items to add to their order
Customer selects cart
Update Order, Total Count & Order Tracking
Display items in cart, order total & option for delivery
Customer selects delivery option
Prompt user for name & address
Customer enters name and address
loop until is authenticated
Prompt user for payment information
Customer enters payment information
Verify payment information
Reset Order, Total Count & Order Tracking
Order Confirmation number & leave a Review
Update Order History & Reward Points

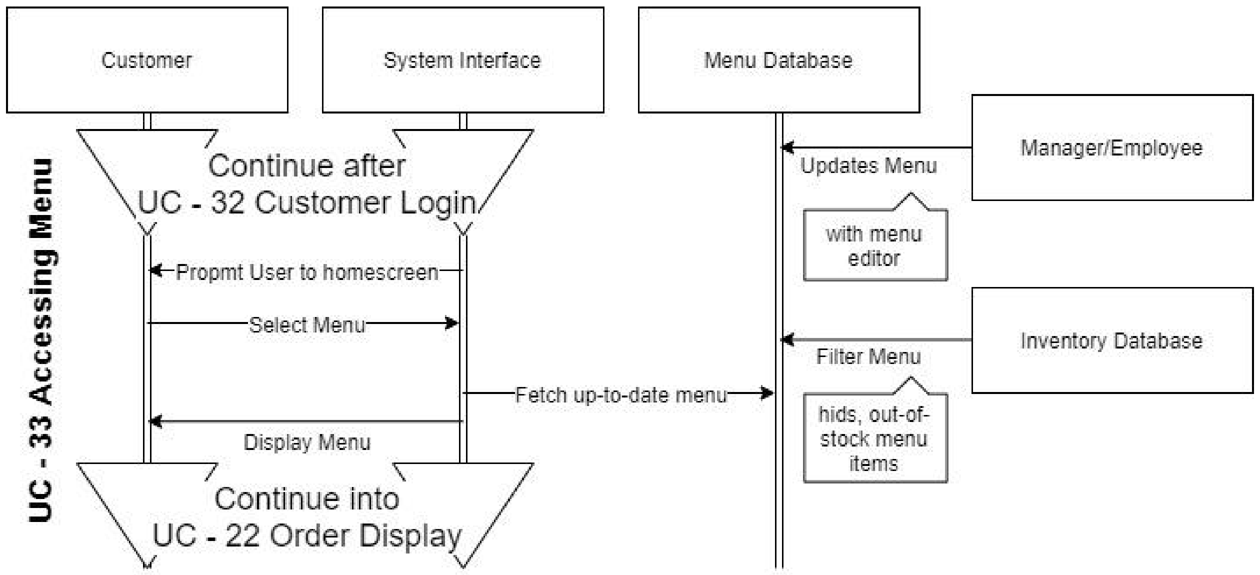
Order Display requires the user to log in (and create an account if they have not already) because the system will fetch the recently saved order information, this includes the shopping cart of menu items added to the order, a calculation of current total cost, and calorie count. Here the customer can remove items from their order, if they leave the shopping cart, the system will update the changes the customer has made to the order back into the database, they can add menu items to the order and the database will also be updated accordingly. When the customer wants to complete and purchase their order, the system will execute some liner operations, which involve prompting the customer for their shipping address and payment options, when both of those options are valid, the customer is sent an order confirmation. The system updates their order history, rewards the customer points based on their order, resets their current shopping cart, and allows them to track and leave a review on their order.

UC-32: Customer & Employee Login


User (both customer & employee)
System Interface
Account Information Database
UC - 32 Customer & Employee Login
Select button "Login
Loop until results if verificatoin is valid
Prompt for the username & Password
Ender username & Password
Verify username & Password
Results
Fetch Employment status
Return Account information
Results
Display certian Account information
Based on Emplotment status: Includes Table, Inventoty Dipslay, Drone Tracking, Order History etc.

The Customer and Employee login is separated by customer and employee but the operation is the same, the system will prompt the user with a request for their username and password. The system will keep making this prompt until the users information is valid and links to an account. The system will then fetch the employment status of the user and return the proper account information for the system to display accordingly. If it is a customer, the user will be able to view their order history and track any current orders. If it is an employee they will be able to view the tables, inventory, or track drones. Higher-level employees will be able to edit the menu and update inventory or schedule a time where customers are locked from placing orders or playing games.

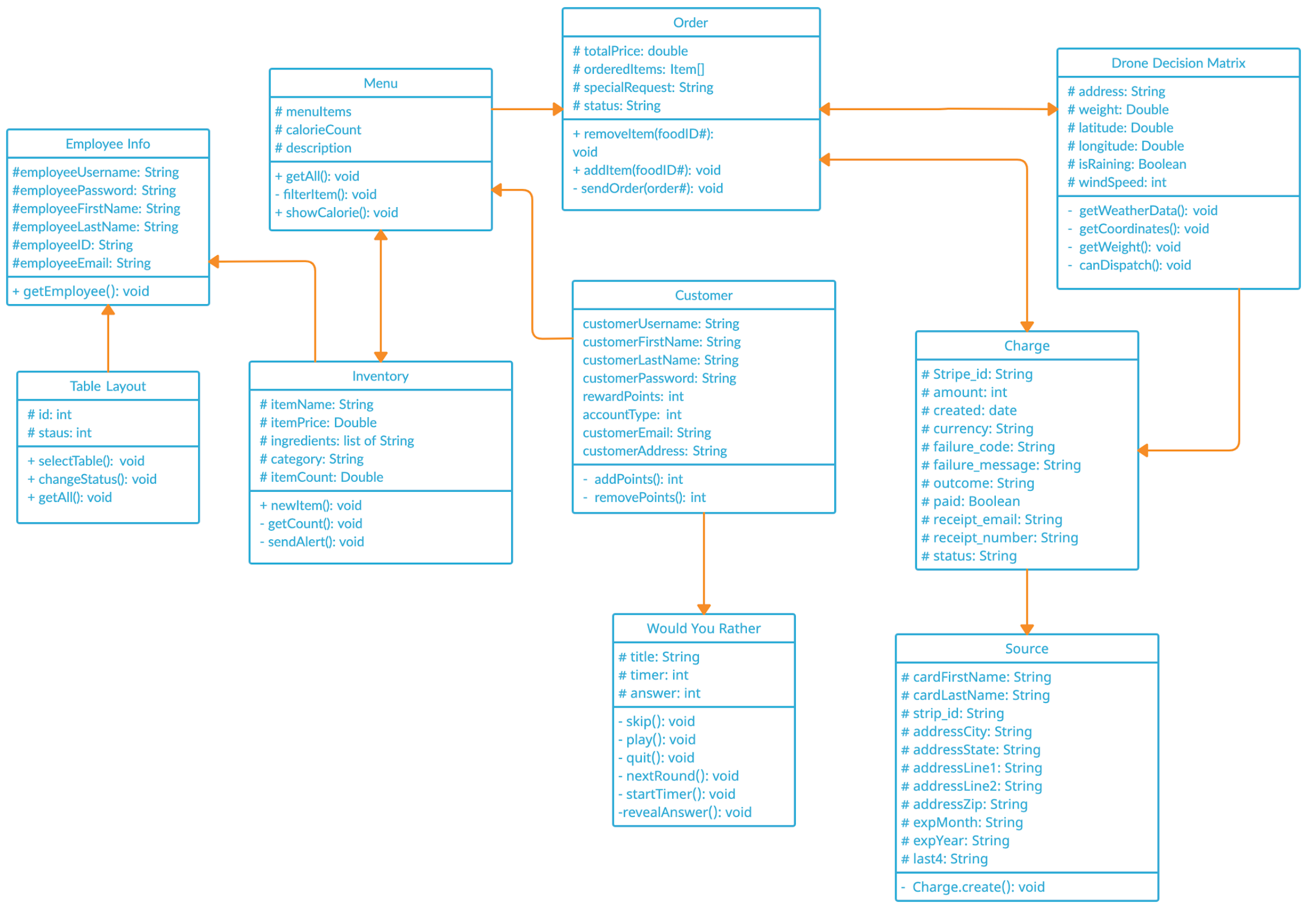
UC-33: Accessing Menu



When the menu button is selected, the system has to fetch the most recent update of the menu, filtered by the manager and the inventory database the system will display the menu accordingly. While viewing the menu, the customer can view the calorie count, add items from the menu into their order and select certain features to the said menu item (Like sauce on the side, extra pickles, etc.)

Section 10: Class Diagrams and Interface Specification

10.1: Class Diagram



10.2: 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 charged

- cardLastName: String - The last name of the person being charged
- stripe_id: { type: Sequelize.STRING, primaryKey: true }, - The charge’s unique stripe ID
- addressCity: String - The city of the billing address.
- addressState: String - The state of the billing address.
- addressLine1: String - The address of the billing address.
- addressLine2: String - Alternate address of the billing address.
- addressZip: String - Zip code associated with the billing address.
- cvcCheck: String - Makes sure the cvc code matches the record.
- expMonth: String - The expiration month of the payment card
- expYear: String - The expiration year of the payment card
- last4: String - The last 4 digits of the payment card

Methods:

- Charge.create() - Creates a new charge for the table and defines all associated variables

10.3: 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 11: Algorithms and Data Structures

11.1: 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.

Turbo-Yum Enhancements: 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.

11.2: 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.

Turbo-Yum Enhancements: 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.

11.1: 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.

Turbo-Yum Enhancements: 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 12: 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 has 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 13: Design of Tests

Drone Delivery:

Test Case Identifier: TC-1
Use Case Tested: UC-6
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 performs 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.

Test Case Identifier: TC-2
Use Case Tested: UC-3
Pass/Fail Criteria:The test will pass if the operator inputs a drone's status and the status updates in the system.The test will fail if the the operator inputs a drone's and the status does not update
Input Data:

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.

Game App:

Test Case Identifier: TC-4
Use Case Tested: UC-7 "start the Game"
Pass/Fail Criteria: PASS
Input Data: none

Test Procedure	**Expected Result:**
User clicks on a button on the drop-down bar	System displays WYR title and buttons that

that prompts the WYR game.	lead to the tutorial

Test Case Identifier: TC-5 **Use Case Tested: UC-9 “Play”** **Pass/Fail Criteria: PASS** **Input Data: none**	
Test Procedure	**Expected Result:**
User clicks on a button that prompts a “Ready?” statement	System displays the WYR image and, when the time bar is completed, displays the resulting image.

Test Case Identifier: TC-6 **Use Case Tested: UC-10 “Next WYR”** **Pass/Fail Criteria: PASS** **Input Data: none**	
Test Procedure	**Expected Result:**
User clicks on a button that prompts a “Ready?” statement	System displays the WYR image and, when the time bar is completed, displays the resulting image.

Test Case Identifier: TC-7 **Use Case Tested: UC-11 “End”** **Pass/Fail Criteria: PASS** **Input Data: none**	
Test Procedure	**Expected Result:**
User clicks on a button that prompts the WYR game.	System displays WYR title and buttons that lead to the tutorial

Test Case Identifier: TC-8 **Use Case Tested: UC-8 “Tutorial”** **Pass/Fail Criteria: PASS** **Input Data: none**	
Test Procedure	**Expected Result:**
User clicks on a button that prompts the Tutorial slides	System displays an image slideshow of directions of games.

Inventory:

Test Case Identifier: TC-12 **Use Case Tested: UC-13** **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.

Test Case Identifier: TC-13 **Use Case Tested: UC-14** **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.

Calories/Health Data:

Test Case Identifier: TC-14 **Use Case Tested: UC-18** **Pass/Fail Criteria:** The test will pass if the system properly toggles between the two menus **Input Data:** N/A	
Test Procedure	**Expected Result:**
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.

Test Case Identifier: TC-15 **Use Case Tested: UC-17** **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.** **Input Data: quantity**	
Test Procedure	**Expected Result:**

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-16 **Use Case Tested: UC-19** **Pass/Fail Criteria: Pass if database updates successfully after inputting corresponding data.** **Input Data: Food class including calorie count, price, name, description**	
Test Procedure	**Expected Result:**
Manager adds a menu item through the manager computer	The menu item is added the menu and displays the corresponding calories

<u>Turbo-Yum Enhancements:</u>

Test Case Identifier: TC-3 **Use Case Tested: UC-32 “Customer and Employee Login”** **Pass/Fail Criteria: PASS** **Input Data: username, email, phone number (given that an account has already been created with the given credentials)**	
Test Procedure	**Expected Result:**
User inputs email username, email and phone number. User then clicks on a “sign in” button	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-9 **Use Case Tested: UC-33 “Access Menu”** **Pass/Fail Criteria: PASS** **Input Data: None**	
Test Procedure	**Expected Result:**
Hit takeout and view menu, dine in will show table number, calorie count, price, name, display quantity and filter.	System displays menu that displays all menu items, calorie count, price, name.

Test Case Identifier: TC-10 **Use Case Tested: UC-22 “Order”** **Pass/Fail Criteria: FAIL (Total is not being displayed properly)** **Input Data: None**	
Test Procedure	**Expected Result:**
User sets quantity and adds menu item to cart, User clicks shopping cart button to view Order	System displays a list of selected menu items and, calorie count, total price.

Test Case Identifier: TC-11 **Use Case Tested: UC-28 “purchase”** **Pass/Fail Criteria: FAIL** **Input Data: User Card information**	
Test Procedure	**Expected Result:**
User clicks shopping cart and clicks purchase button, User inputs credit card credentials	System prompts payment system and sends order to kitchen and drone delivery system.

Project Management

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.

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

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-V7Uu6bl1SlcCRfLH79HZrFAtBvX&index=1
- The Reactive Manifesto
  Author: Jonas Bonér, Dave Farley, Roland Kuhn, and Martin Thompson.
  Title: The Reactive Manifesto
  Link: https://www.reactivemanifesto.org/