COMP1531

Gourmet Burger - Report
Team Alpha

Siyin Zhou	- z5232974
Leonie Dickson	- z5215454
Peter Kerr	- z5115807

User Stories (Priority is ranked from 1 to 5. 1 story point corresponds to 2 hours.)

Epic Story 1: Customers - Place Online Orders

As a customer, I want to use an online interface to order mains, sides and drinks.

ID	US 1.1
Name	Make a customised main order

User-Story Description:

As a customer, I want to make a customised main order, so that my personal preferences are satisfied.

Acceptance Criteria:

- A customer can choose between either a burger or wrap. Error message “Cannot choose both burger and wrap” displayed if customer attempts to choose both.
- If burger is selected, customer must choose out of burger bun options including sesame bun and muffin bun. Error message “Number of buns must be between 2 and {allowable maximum number of buns}” displayed if customer attempts to choose number of buns exceeding the allowable maximum of buns. If patty is 0, maximum buns is 2, otherwise the maximum buns allowed is number of patties plus one.
- If wrap is selected, customer can choose only one wrap as either flat bread wrap or whole wheat wrap. Error message “Must select one wrap bread” displayed if customers attempt to order with no wraps selected or more than 1 wrap.
- A customer can choose customisable ingredients from a selection including tomato, lettuce, tomato sauce, cheddar cheese and swiss cheese.
- An error message displaying "Insufficient stock for {ingredient}." displayed for corresponding ingredients if there is insufficient stock level to satisfy the customer's main order.
- An error message displaying “Cannot enter negative quantity” is shown if the customer enters a negative quantity for any ingredient.
- If the “Add to Order” button is pressed and there are no errors, the page should be refreshed and a message saying “Main added to order” should be visible. The addition to the order should now be visible when reviewing order.
- After successfully adding main order to online order, the ingredients required are reserved and deducted from the current inventory for the period of allowable ordering time before the order expires in 30 minutes after creation.
- When clicked on "Review Order", a net price is displayed for the customer's gourmet main creations, which is calculated based on the chosen ingredients.

Priority	1
Size	12 story points

ID	US1.2
Name	Order sides
User-Story Description: As a customer, I want to order sides so that I can enjoy them. **Acceptance Criteria:** ● Options for nuggets displayed with 3 sizes (small = 3/ medium = 6/ large = 9 packs) ● Options are displayed as: fries, chocolate sundae, strawberry sundae (small = 100g / medium = 150g / large = 200g) ● An “Add to Order” button should be located at the bottom of the page. ● If a request is made to order more of any particular item than there is stock, then an error message “Insufficient stock of {item}” is displayed when the customer attempts to add the side to their order. ● An error message displaying “Cannot enter negative quantity” is shown if the customer enters a negative quantity for any ingredient. ● If “Add to Order” button is pressed and there are no errors, the page should be refreshed and a message saying “Side added to order” should be visible. The addition to the order should now be visible when reviewing order. ● After successfully adding sides to online order, the ingredients required are reserved and deducted from the current inventory for the period of allowable ordering time before the order expires in 30 minutes after creation.	
Priority	4
Size	6 story points

ID	US1.3
Name	Order drinks
User-Story Description: As a customer, I want to order drinks so that I can enjoy them. **Acceptance Criteria:** ● Options for drinks including Coke, Fanta, Sprite, are each displayed with two choices including can and bottle (eg. Can Coke, Bottle Coke). ● Options for fountain drinks including orange juice is displayed with options of varying sizes including small (250ml), medium (450ml), large(600ml) ● An “Add to Order” button should be located at the bottom of the page. ● If a request is made to order more of any particular item than there is stock, then an error message “Insufficient stock of {item}” is displayed when the customer attempts to add their sides to their order. ● An error message displaying “Cannot enter negative quantity” is shown if the customer enters a negative quantity for any ingredient. ● If “Add to Order” button is pressed and there are no errors, the page should be refreshed and a message saying “Drink added to order.” should be visible. The addition to the order should now be visible when reviewing order. ● After successfully adding drinks to online order, the ingredients required are reserved and deducted from the current inventory for the period of allowable ordering time before the order expires in 30 minutes after creation.	
Priority	4
Size	6 story points

ID	US1.4
Name	Review order
User-Story Description: As a customer, I want to review my order so that I can confirm that my order is correct. **Acceptance Criteria:** ● A list of all of the items that the customer currently has added to their order is displayed for review. ● Customers should be able to access this page through a button at the bottom of any order page called “Review Order” ● This page should be updated whenever the customer makes a new addition to their order. ● A “Continue Order” button is displayed below the order summary which directs to the mains page and allows the customer to continue adding items to their order ● A "Checkout" button is displayed below the order summary which directs to checkout. ● A “Cancel Order” button is displayed below the order summary which allows the customer to abandon their order and restocks reserved ingredients in inventory.	
Priority	2
Size	4 story points

ID	US1.5
Name	Checkout
User-Story Description: As a customer, I want to checkout so that restaurant staff can see and prepare my order. **Acceptance Criteria:** ● Once checkout is confirmed, order is marked as paid and is sent to the staff interface as an active order so that they can begin preparing the order. The customer is then sent to the order status page. ● After checkout, inventory levels in the staff interface will have been decremented in correspondence to the items ordered. ● After order is paid, no further modifications can be made to order. If customer attempts to add more main or side orders to a paid order, error page "Sorry, your order is already complete. Please start a new order." is displayed ● When 30 minutes order time expires, no action will be taken as order is already paid.	
Priority	1
Size	6 story points

ID	US1.6
Name	Obtain order ID
User-Story Description: As a customer, I want to obtain an order ID, so I can refer to my order in the future. **Acceptance Criteria:** ● Order ID should be displayed in a message "Order ID: {orderID}" on the top of the order status page after successfully checking out. ● The order ID should be unique to each order. ● Order ID should increase by 1 for each successive order (eg. one customer creates an order and receives ID 27, the next customer to create order receives ID 28).	
Priority	3
Size	3 story points

ID	US1.7
Name	Check order status
User-Story Description: As a customer, I want to be able to check the status of my order with my previously issued order ID so that I can know when my order is ready. **Acceptance Criteria:** ● An insertion field should be located on the home page of the interface titled “Enter order ID:” ● On inserting their order ID into this field, customers should be able to see corresponding order status page. ● If incorrect non-existent order ID is entered, an error message "Incorrect order ID entered." should be displayed. ● If an order ID is entered that corresponds to an order that has not yet been paid for, an error message “Incorrect order ID entered.” should be displayed. ● Status page displays all items ordered, price paid, and current status - either “Your order is being prepared.” or “Your order is ready to be collected.” ● Refreshing this page should update the current status of the order.	
Priority	3
Size	3 story points

ID	US1.8
Name	Order standard main

User-Story Description:

As a customer, I want to be able to order a standard burger or warp so that I can save time by not customising ingredients.

Acceptance Criteria:

- Options are a standard burger (ingredients: sesame bun x2, beef, tomato, cheddar cheese) and a standard wrap (ingredients: flatbread, chicken, lettuce, cheddar cheese).
- Price for respective options is sum of ingredients cost.
- An error message displaying "Insufficient stock for {ingredient}." displayed for corresponding ingredients if there is insufficient stock level to satisfy the customer's main order.
- If the “Add standard wrap” or "Add standard burger" button is pressed and there are no errors, the page should be refreshed and a message saying “Main added to order” should be visible. The addition to the order should now be visible when reviewing order.
- Customers cannot customise a standard option.

Priority	5
Size	3 story points

Epic Story 2: Staff - Service Online Orders

As a staff member, I want to service orders submitted by customers through an online interface.

ID	US2.1
Name	View current orders
User-Story Description: As a staff member, I should be able to view the current orders so that I can service customers. **Acceptance Criteria:** ● Current orders confirmed by customer checkout should appear on the "Active Orders" page, belonging to the staff interface. ● Customer's orders are displayed in a list format with each order displayed in a table. ● Each table will display customer order ID and presents buttons to "Inspect Order" and "Order Prepared" ● If staff clicks on "Inspect Order" for a particular order, a page with order ID, main orders, corresponding ingredients chosen, sides and drinks order is displayed. A button is also present for "Order Prepared". ● If staff clicks on "Order Prepared" for a particular order on either the active order or the inspect order page, the order status will be update as prepared and the order status message will be updated as "Your order is ready to be collected" if customer checks the order status.	
Priority	1
Size	8 story points

ID	US2.2
Name	Update order status
User-Story Description: As a staff member, I should be able to update the status of an order so that the customer knows when their order has been prepared. **Acceptance Criteria:** ● A button labelled ‘Order Prepared’ must be displayed next to each listed order in the Active Orders page and at the bottom of each “Order Details” page. ● After the ‘Order Prepared’ button is clicked, the order will be removed from the staff list. ● The status on the corresponding customer’s status page (referenced through order ID) should be changed to “Your order is now ready to be collected,” which the customer should be able to view after refreshing their status page.	
Priority	1
Size	3 story points

Epic Story 3: Staff - Maintain Inventory

As a staff member, I want to be able to maintain the inventory of the restaurant’s various ingredients.

ID	US3.1
Name	View current inventory levels
User-Story Description: As a staff member, I should be able to see the current inventory levels in the staff section of the interface so that I know the stock status. **Acceptance Criteria:** ● Should display numerical stock information for every ingredient in mains, sides, and drinks in the staff section of the interface. ● Amounts of burger buns, wraps, patties, fillings and nuggets should be displayed in whole quantities. ● Cans and bottles should be displayed in whole quantities under the drinks section. ● Stock of drinks such as orange juice should be displayed in volume (millilitres). ● Stock of sides should be displayed in weight (grams).	
Priority	2
Size	6 story points

ID	US3.2
Name	Update inventory levels
User-Story Description: As a staff member, I should be able to update the inventory levels so that the physical state of the inventory is reflected in the online interface. **Acceptance Criteria:** ● Should be able to enter a numerical value in a field beside the ingredient they wish to update stock for (positive value for refilling, negative value if, eg. stock is damaged and removed) ● The inventory level for each ingredient is unable to decrement below 0. If staff attempts this, an error message "Error: Insufficient stock for {ingredient}" is displayed. ● After selecting 'update', the staff member should be able to see the changes made reflected in the inventory. ● If the ingredient being updated is stocked in whole quantities, and the staff enters a non-integer value, an error message "Error: Can not enter non-integer quantity" should be displayed.	
Priority	2
Size	4 story points

ID	US3.3
Name	Add new ingredients

User-Story Description:

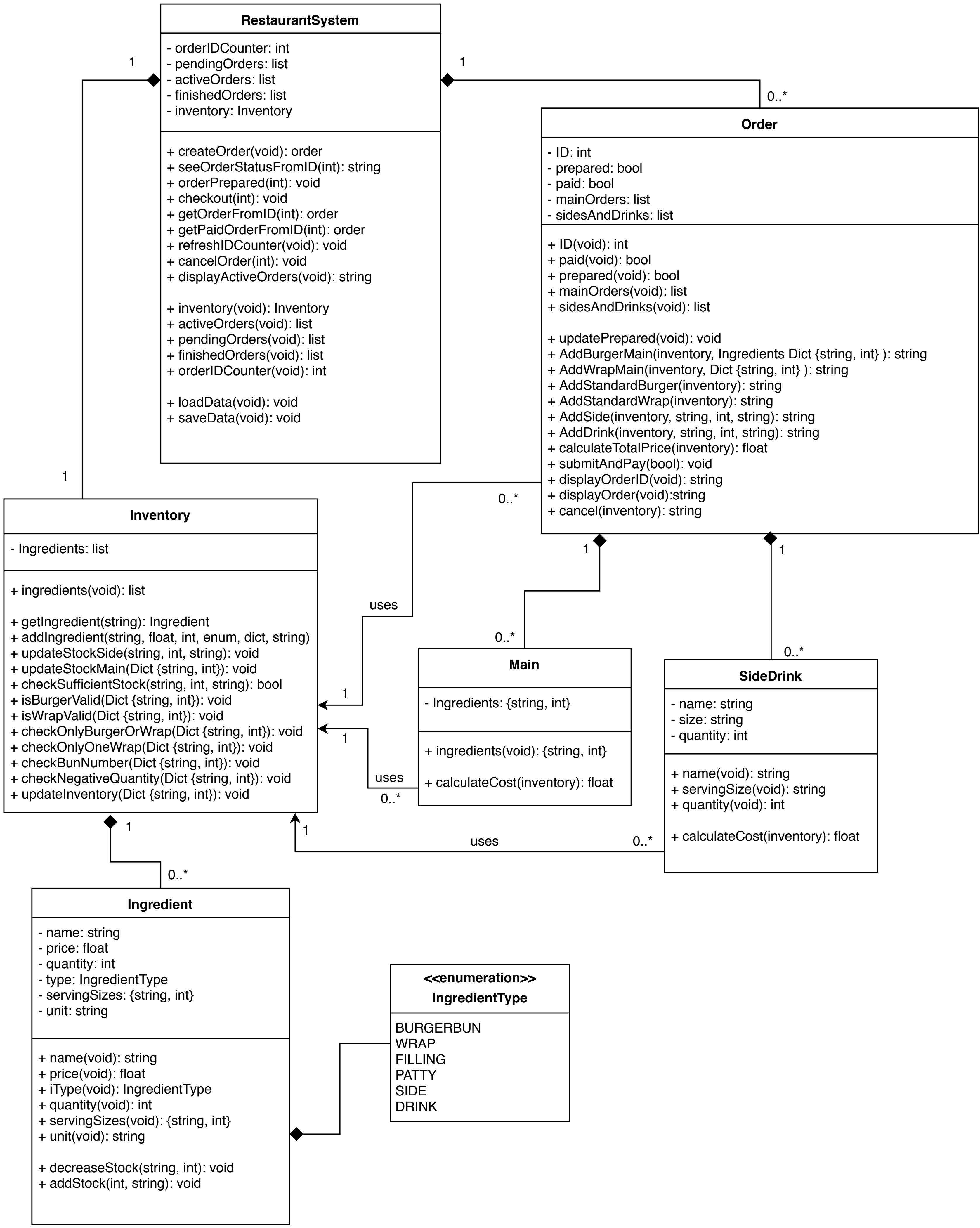
As a staff member, I should be able to add ingredient to inventory so that it can be used to reflect the physical state of the inventory.

Acceptance Criteria:

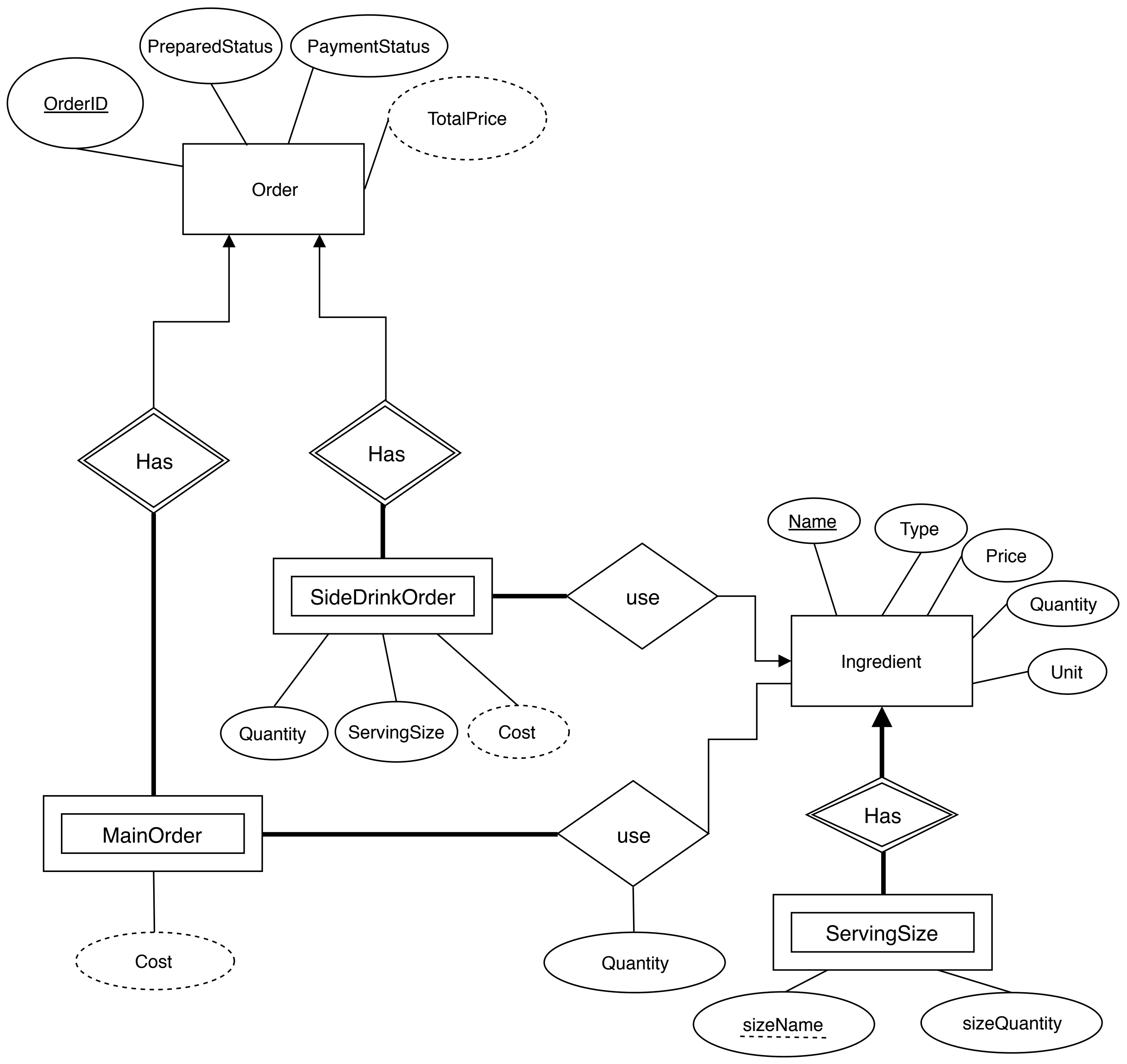
- Must enter ingredient name with alphabetical letters and space with no numbers. Message "Error: Please enter valid ingredient name" shown if name does not match criteria. Message “Error: Please enter ingredient name” shown if no name is given.
- Should be able to enter a non-negative numeric value as quantity. If no value is entered, the quantity is set to be zero. Error message "Please enter valid ingredient quantity” is displayed if a non-integer value or negative value is entered.
- Must select the Ingredient Type of the ingredient out of displayed options: Burgerbun, Wrap, Filling, Patty, Side, Drink. If invalid input is provided, error message "Error: Please enter valid ingredient type" is displayed.
- Must enter ingredient name that does not already exist in the inventory. If so, error message “Error: Ingredient is already in inventory” is displayed.
- Must enter ingredient price as a non negative float less than or equal to $30. If input given does not match criteria, error message "Error: Please enter valid ingredient price" is displayed.
- Must enter unit selected from “regular unit”, “g” or “ml”. If another unit is given, error message “Please enter valid unit” is displayed.
- If regular serving size is chosen, must choose "regular unit" as unit. Otherwise error message "Error: If choosing regular serving size, must choose regular unit" is shown. Can only select "ml" as unit if choosing type Drink, otherwise error message "Error: Must be type drink to be stored in ml" is shown. If ingredient is not chosen as type side or drink, must select regular unit, otherwise error message "Error: Must be stored as regular unit if ingredient is not side or drink" is shown. If choosing to store ingredient as type Drink, must not store it in grams, otherwise error message "Error: Drink can not be stored in grams" is outputted.
- After selecting ‘update’, the staff member should be able to see the changes made reflected in the inventory.
- If the ingredient being updated is stocked in whole quantities, and the staff enters a non-integer value, an error message "Error: Can not enter non-integer quantity" should be displayed.

Priority	2
Size	4 story points

GourmetBurger Online System Class Diagram


RestaurantSystem
- orderIDCounter: int
- pendingOrders: list
- activeOrders: list
- finishedOrders: list
- inventory: Inventory
+ createOrder(void): order
+ seeOrderStatusFromID(int): string
+ orderPrepared(int): void
+ checkout(int): void
+ getOrderFromID(int): order
+ getPaidOrderFromID(int): order
+ refreshIDCounter(void): void
+ cancelOrder(int): void
+ displayActiveOrders(void): string
+ inventory(void): Inventory
+ activeOrders(void): list
+ pendingOrders(void): list
+ finishedOrders(void): list
+ orderIDCounter(void): int
+ loadData(void): void
+ saveData(void): void
1
1
0..*
Order
- ID: int
- prepared: bool
- paid: bool
- mainOrders: list
- sidesAndDrinks: list
+ ID(void): int
+ paid(void): bool
+ prepared(void): bool
+ mainOrders(void): list
+ sidesAndDrinks(void): list
+ updatePrepared(void): void
+ AddBurgerMain(inventory, Ingredients Dict {string, int}): string
+ AddWrapMain(inventory, Dict {string, int}): string
+ AddStandardBurger(inventory): string
+ AddStandardWrap(inventory): string
+ AddSide(inventory, string, int, string): string
+ AddDrink(inventory, string, int, string): string
+ calculateTotalPrice(inventory): float
+ submitAndPay(bool): void
+ displayOrderID(void): string
+ displayOrder(void):string
+ cancel(inventory): string
1
0..*
uses
1
Inventory
- Ingredients: list
+ ingredients(void): list
+ getIngredient(string): Ingredient
+ addIngredient(string, float, int, enum, dict, string)
+ updateStockSide(string, int, string): void
+ updateStockMain(Dict {string, int}): void
+ checkSufficientStock(string, int, string): bool
+ isBurgerValid(Dict {string, int}): void
+ isWrapValid(Dict {string, int}): void
+ checkOnlyBurgerOrWrap(Dict {string, int}): void
+ checkOnlyOneWrap(Dict {string, int}): void
+ checkBunNumber(Dict {string, int}): void
+ checkNegativeQuantity(Dict {string, int}): void
+ updateInventory(Dict {string, int}): void
1
1
0..*
Main
- Ingredients: {string, int}
+ ingredients(void): {string, int}
+ calculateCost(inventory): float
1
uses
0..*
1
0..*
SideDrink
- name: string
- size: string
- quantity: int
+ name(void): string
+ servingSize(void): string
+ quantity(void): int
+ calculateCost(inventory): float
1
uses
0..*
1
0..*
Ingredient
- name: string
- price: float
- quantity: int
- type: IngredientType
- servingSizes: {string, int}
- unit: string
+ name(void): string
+ price(void): float
+ iType(void): IngredientType
+ quantity(void): int
+ servingSizes(void): {string, int}
+ unit(void): string
+ decreaseStock(string, int): void
+ addStock(int, string): void
<<enumeration>>
IngredientType
BURGERBUN
WRAP
FILLING
PATTY
SIDE
DRINK

Entity Relationship Diagram for Gourmet Burger System:



Log Book:

Meeting minutes: Mon 4 Mar

Progress - Initial discussion on user stories (Milestone 1):

- 3 Epic stories: Customer - online orders; Staff - service online orders; Staff - maintain inventory
- Each user story -> one web page

Next meeting: *Wednesday 6 March* 6-8pm, Location: ground J17

- Plan for meeting: finish user stories, log book entry, setup github usage guidelines
- Pre-meeting tasks:
    - Everyone:
        - Catch up on lectures
        - Understand assignment requirements
        - Note any points to discuss in next meeting
    - Peter: Log book template
    - Amy:
        - Find unsure points to ask lecturer
        - Github workflow diagram (ask about what strategy to use)

Meeting : Wednesday, 6 March 2019

Progress:

- Confirmed three epic stories from last meeting
- Wrote majority of user stories and acceptance criteria
- Decided on priority scale - using ranking
- Decided 1 story point corresponds to 2 hours

Difficulties and obstacles:

- Some difficulty in writing good "benefits" for user stories that did not restate goal or introduce goal of another user story
- Some review needed of user stories eg. regarding how staff will access the online interface

To do:

- Confirm user stories including staff - online access
- Insert priority & story points for each user story (via Discord call on Saturday)
- Check and confirm individually by *Friday 9 March*
- Final review of user stories (via Discord call) and submit - *Saturday 10 March (AM)*

Meeting: Wednesday, 27 March

Progress - Initial brainstorming for UML Class Diagram:

- Brainstormed possible classes for online order system implementation (rough ideas only)
    - Established RestaurantSystem and OrderReceipt classes
    - Established Ingredient class and 5 subclasses: Bread, Patty, Filling, Side, Drink
    - Inserted StaffOrderSystem class and Customer class (but unsure of this representation)
    - Created MainOrder class
- Created rough draft of UML Class Diagram with draw.io
    - Inserted brainstormed classes and began adding attributes and methods (incomplete)

Difficulties and obstacles:

- Unsure about how customers/customer user interface should be represented in class diagram
- Unsure about how classes relating to Ingredients, Inventory, Menu Items (eg. a burger main order) should be represented.
- Unsure of how to organise classes such that functions such as calculating cost and validating order (eg. ensuring only one wrap) can be efficiently implemented.

To do:

- Clarify classes and relationships in class diagram
- Clarify where methods should go (ie. what class they should fall under)

Meeting: Friday, 29 March

Progress:

- Consulted for class diagram and clarified ideas on class attributes, methods and relationships
- Revised class diagram:
    - Removed Customer/Customer UI and StaffOrderSystem representations
    - Removed subclasses of Ingredient class and substituted for enum to represent type of ingredient
    - Added inventory (list of ingredients)
    - Added SideDrink class (which composes Order, along with Main) to reflect essential differences in class methods

Difficulties and obstacles:

- Difficulties in determining most ideal representation of inventory, ingredients and their types - considered dictionaries, subclasses, enum

To do:

- Finalise class diagram (by *Sunday 31 March*)
- Check that all user stories (except Epic Story 3: Staff - Maintain Inventory) have been covered in class diagram
    - Add any missing attributes/methods accordingly

Meeting: Wednesday, 3 April

Progress and decisions made:

- Discussed implementation of backend (as planned in class diagram)
- Determined branches to be used on GitHub (inventory, order, system, master)
- Decided to use a Trello board to keep track of tasks and roles
- Separated remaining implementation and testing into more granular tasks
- Decide to adjust plan to reinitialise order ID at end of business day as this will make order ID no longer unique and can cause system confusion. Justification: integer overflow is unlikely for python as python has arbitrary precision
- Delegated tasks to each team member in order to achieve milestone:
    - Amy:
        - (Had previously implemented Inventory, Ingredients, Main, SideDrink, Order classes, and written initial testing for ingredients, inventory and order classes)
        - Complete testing for inventory and order
    - Peter:
        - Write extensive tests for Main and SideDrink classes
    - Leonie:
        - Implement RestaurantSystem class and conduct extensive testing for this class
        - Organise log book

Next meeting: *Sunday 7 April*

- Review and finalise backend implementation
- Check that all user stories (with the exception of Epic Story 3: Staff - Maintaining Inventory) have been fully implemented and all acceptance criteria has been met.
- Compile all elements for Milestone 2 (product backlog, working software, logbook) and submit.

Meeting: Sunday, 7 April

Progress and decisions made:

- Check progress of implementation and testing of backend
    - Ensure all tasks on Trello board is completed
- Finalise integration testing for entire restaurant system
- Merge finished backend implementation into master and then merge into release branch for submission
- Evaluation of progress using velocity chart as shown below
- Check testing traceability back to user story acceptance criterias
    - Ensure all acceptance criteria associated with backend implementation has been tested
- Add negative quantity order input checks
- Adjust user story

Milestone achieved:

Epic Story 1: Customer - Place online order

US 1.1: 10 out of 12 points
US 1.2: 4 out of 6 points
US 1.3: 4 out of 6 points
US 1.4: 3 out of 4 points
US 1.5: 5 out of 6 points
US 1.6: 3 out of 3 points
US 1.7: 2 out of 3 points

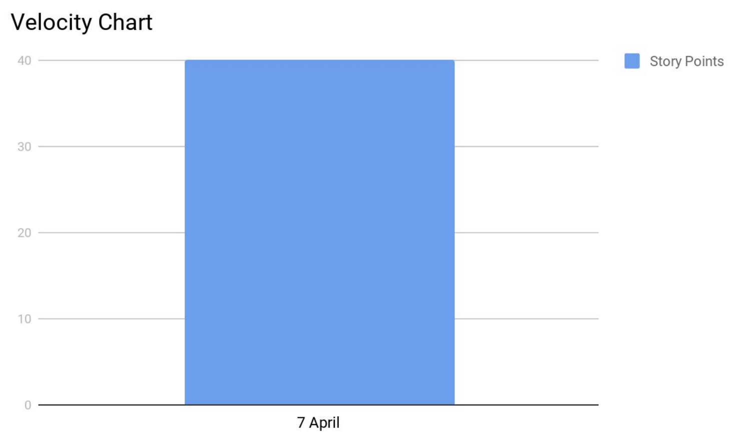
Epic Story 2: Staff - Service online order

US 2.1: 7 out of 8 points
US 2.2: 2 out of 3 points

Reflection:

All backend user story acceptance criteria accomplish.
Remaining story points all associate with front end of restaurant system.

Velocity chart: Velocity = 40



Meeting: Wednesday, 10 April

<u>Progress and decisions made:</u>

- Reviewed customer feedback from recent iteration and new functionalities required
- Updated user stories to reflect the changes
    - Added new user story: US1.8: Order standard main
    - Adjusted acceptance criteria of US1.2 to account for new menu addition of sundaes
- Updated class diagram to reflect the changes
    - Added two new methods in Order class: AddStandardBurger and AddStandardWrap
- Updated backend to implement new changes
    - Implemented new functions (as above) in order.py in order branch
- Decided to store all ingredient names with lower case in inventory
- Make a CSS style sheet (Low priority)
- Determined app routes for Flask:
    - index ("/"):
        - make an order: createOrder
        - POST: check order status -> "/order/<ID>/status"
    - "/order/<ID>/main":
        - add standard wrap -> "/order/<ID>/sidedrink"
        - add standard burger -> "/order/<ID>/sidedrink"
        - add customised burger -> "/order/<ID>/burger"
        - add customise wrap -> "/order/<ID>/wrap"
        - add side or drink -> "/order/<ID>/sidedrink"
    - "/order/<ID>/review" -> "/order/<ID>"
    - "/order/<ID>/status"
    - "/staff"
        - display active orders
    - "/staff/<ID>"
        - display order details
        - update prepare status
    - "/staff/inventory"
- git branch frontend

<u>To do:</u>

- Add more tests for addStandardBurger and addStandardWrap
- User Acceptance Testing
- Unit Testing
- ER design
- Front end
    - Become familiar with Flask and Jinja
- Ask about Epic Story 3: Maintain inventory implementation
- Decide on required global variable
- Persistence implementation?

Meeting: Tuesday, 16 April

<u>Progress and decisions made:</u>

- Continued writing html for GourmetBurger system
- Discussed how to implement errors in frontend
- Decided to use html input constraints to implement sufficient stock and positive quantity validations for wrap and burger.
- HTML files split between group for completion by Sunday (as assigned on Trello)
- Refactored order validation code

<u>Obstacles:</u>

- Issues with how to pass order through different routes of system
- Issues with considering whether or not to create a form class - necessary?
- Unsure if the use case of staff updating stock is required

<u>To do:</u>

- Finish html
- Ask questions to clarify understanding of assignment requirements
    - Request.form
    - Error handling
    - How to pass order through url and form - post
    - Update stock use case

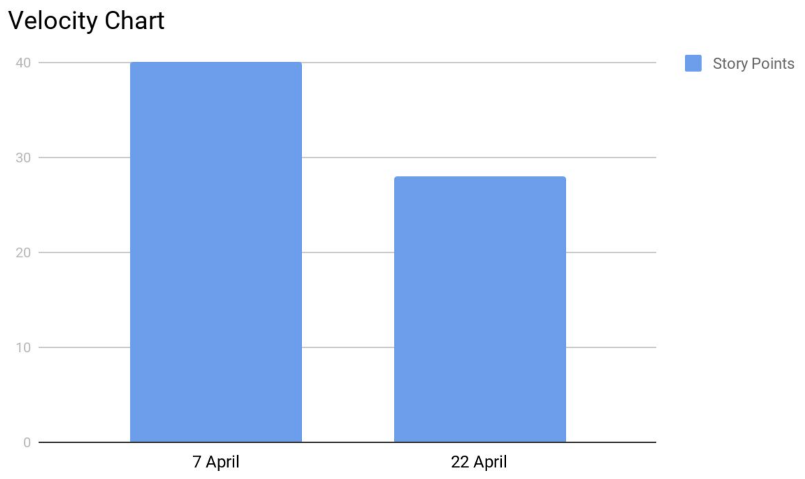
Meeting: Wednesday 17 April

<u>Progress and decisions made:</u>

- Decide to add unit attribute to ingredients class and adjust functions/tests
- Epic Story 3: staff - maintain inventory frontend
    - Update inventory
    - Add Ingredients
- Completed forms for error handling in frontend

<u>To do:</u>

- Comprehensive user testing of interface
- Update class diagram
- Update product backlog



Meeting: Wednesday 24 April

<u>Progress and decisions made:</u>

- Update class diagram
- Update product backlog
- Plan product presentation
- Begin ER diagram

<u>To do:</u>

- Finish ER diagram
- Compile report