Comp1531 Logbook

Group Members:

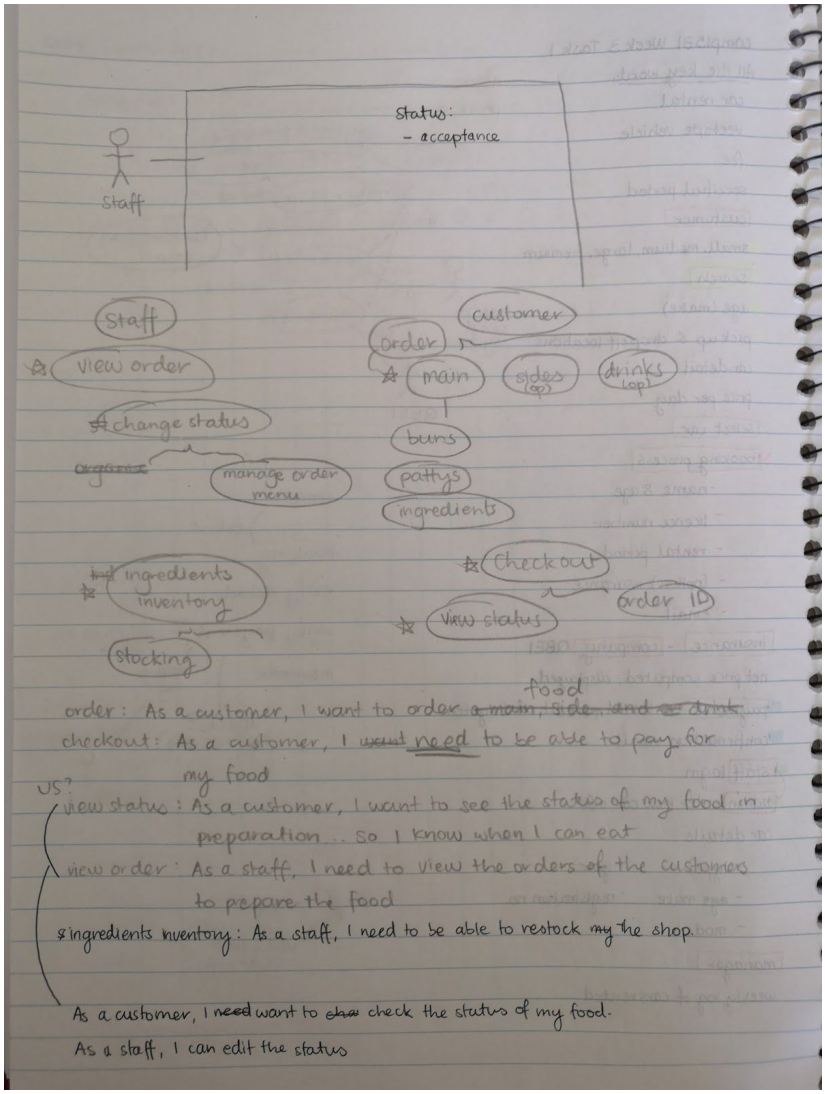
Daniel, Andre, Xinyi Sun

Meeting: 5th March

Xinyi and Andre met to overview and discuss assignment specifications.

Brainstormed ideas briefly about assignment:

- Maybe around 4 to 5 epic stories for Order, Checkout, View Status (for customer), View Order (for staff), and Inventory
- Explored if it was possible to combine view status and view order into one epic story and have a conjoined actor. Decided against this as both customer and staff were absolutely necessary actors and we shouldn't create a third actor that somehow is a combination of both to exist at the same time.
- Here is a picture of a rough draft of the epic stories:



Created user stories for the 5 epic stories: (1) Order, (2) Checkout, (3) View Status (for customer), (4) View Order (for staff), and (5) Inventory

Meeting: 20th March

Andre and Daniel work on beginning assignment milestone 2
Created in depth CRC cards without contributions in preparation of UML diagram.
Classes include (1) staff, (2) plate (customer), (3) Queue, (4) Order, (5) Inventory.
Responsibilities include attributes and functions contained within each class, with expected parameters for each function. Posted as notes in our group project.

Meeting: 27th March

Andre, Daniel, Xinyi worked on assignment milestone 2:

- Divided the workload
- Finished creating class diagram and it is ready to submit.
- Daniel completed class inventory and food.
- Andre has been assigned class order
- Xinyi has been assigned class queue, staff

Meeting: 29th March

Group collaborated to work through some conflicts about classes:
Resolved issues:

- Should there be a composition association between inventory and order?
    - Resolved: No: It should just be regular association, because inventory is larger than order, so it would not be a good idea to have order as the container
- Should queue have change_status()?
    - Resolved: Yes: Queue should have change_status because queue oversees all orders

Meeting: 2nd April

Milestone 2 backend update:
Xinyi and Andre met to work on milestone 2.
Andre has completed order, just require testing now
Xinyi has completed staff, queue, and additional class error which handles errors

Discussion:

- New class might be needed for admin systems
- How will the validity of the user be remembered? Right now staff login is called in every instance where staff only methods are accessed (e.g. changing the status of the order - change_status())
- Added a new list in Queue for storing history of completed orders
- Should queue be responsible for removing items from inventory or should order? Or should we create new class which does that?
    - Resolved: Order is fine. Just have a confirm order and then subtract relevant items from inventory. This does not need to happen simultaneously to add_order from Queue, which was what was originally thought

Meeting: 3rd April

Milestone 2 update:

Classes queue, staff, inventory, food has been implemented. Xinyi has begun working on and mostly finished the main skeleton of main. Encountered bugs were minor: syntax, value errors, wrong comparisons, and similar. Order is still being worked on by Andre: encountering difficulties with function delete_item because can't use remove() for dictionaries.

Individual Work: 5th April

Milestone 2 update:

Xinyi created a new class called menu to reduce the complexity of main.py. Main.py was getting too long, so menu.py was created. It contains the functions view_menu and order_from_menu. View menu prints out available foods from inventory separated into correct categories of mains, sides, and drinks. Order from menu appends food customer orders to a new order.

Individual Work: 6-7th April

Group realised that wraps and burgers must be separated, as the specs state that there can only be a choice between one. Daniel edited inventory, food to separate all the main menu foods into main (non-burger/wrap specific, such as cheese and lettuce), burger main (buns, patties), and wrap main (wraps). Andre updated order to pick up difference between ordering

mains of burger and wraps. Xinyi added checks to menu.py to check that when a burger is chosen, customer can't order from wrap menu, and vice versa.
Xinyi separated parts of main function to add to staff.py under a new function called staff_interface. This shortens the main, and puts all the staff processes like change status, check order history calls to staff.py instead of main. Daniel has written pytests, raised an issue where queue.py accepts junk strings as order. This issue is resolved without making changes to queue.py, because order has no specific format for queue to check what its taking in against. Checks have been implemented for when new orders are initialised to ensure orders are not junk.

Meeting 10th April

Meeting held to allocate sections for implementing front end and commence work. The front end task was split into the following 3 portions:

1. Customise unique orders, menu, create and store order
2. Premade orders, show price, front page, setting up inventory to use
3. Staff login, logout, view orders, see and update status of orders, view and update inventory

It was easy to split up the tasks, as they had already been implemented in the back end and so we anticipate there won't be much time and workload that this front end would require. Andre worked on portion 1, Daniel worked on portion 2, and Xinyi worked on portion 3 for an even split of responsibilities.

Meeting 17th April

Meeting held to overview what each group member had committed for the front end. Most of the work was done, and together our routes were combined to create the main route.py, and everything was linked together. Daniel and Andre had spent a lot of time working on the impressive CSS of the site.

Some challenges that we had encountered included needing to rewrite some older functions that had worked for the back end implementation. Namely, main.py, which Xinyi had later found out was unnecessary (it provided a 'front end' view from terminal on what the site should look like - taking in user input to commence certain commands, such as login as staff, customise a burger, display the menu, customise inventory… etc for all the implemented functions). Main.py

required a lot of the functions to output boolean values instead of lists and other structs. Xinyi worked on rewriting the outputs of staff,py, queue.py, and Andre worked on rewriting the outputs of order.py. This ensured that the front-end was able to retrieve the correct information and not be reliant on prints that had been needed for outputting the orders, etc. in terminal. Xinyi realised that the way main.py was written was highly inefficient and an outstanding example of low cohesion. A lot of reflection had been made to ensure the new code is efficient and follows the Single Responsibility Principle, though there are room for improvement for more reusability (some functions still have overlapping uses).

Daniel created more food options, including sundae, to fulfil marking criteria for extra credit. He implemented the option into customisable orders. Another challenge that Daniel had to fulfil was rewriting the layout of inventory such that it can be presented in a presentable, readable manner for flask instead of terminal. Where his previous inventory had looked neat for an output in terminal, it has lost all of its structure when it was attempted to be printed to the site. As a result, the entire inventory had to be reformatted.

Meeting 19th April

The last meeting to finalise the front end implementation, create pytests. Some challenges we did run into was mostly due to miscommunication: we had so many different branches on github that on some occasions, we would pull from the wrong branch to write our pytests for, only to find out that it was incompatible with what would be pushed to the final release branch and so pytests had to be edited. But overall, the team had good communications on how our visions for the assignment and how our own portions of the program would come together.

Overall, the group exhibited good teamwork and dedication. XxxDinerDashxxX created a beautiful (credits to Daniel and Andre for the CSS) and functional site.

Velocity Chart:

The above chart is the for the contributions to master over the course of February 24th until April 28th.