COMP1531 ASSIGNMENT – MILESTONE 2
LOGBOOK

TUES, 5th of MARCH

This was our team's first in-person meeting and mostly consisted of deciding on communication and team work plans, reviewing the specification, and starting the first iteration of our user stories.

These are the general logistic-related aspects that were established in this meeting:

COMMUNICATION:
- Facebook messenger for general communication and written stand ups
- Skype/Facebook messenger calls for detailed meetings and key decision making
- In person meetings every Tuesday during and after our weekly labs

GENERAL TEAM WORK SETUP:
- Created the team repository
- Created a folder in Google Drive to store notes, user stories, diagrams

EXPECTATIONS:
- Being responsive to messages
- Being accountable for the work each member has been assigned
- Being transparent if a work is unable to be completed, or if any issues arise
- Availabilities were established
    - Chris has 3 days of work and full-time university
    - Mel has 4 days of work and full-time university
    - Will has 2 days of work and full-time university

During this meeting, we also began writing our user stories. This involved general brainstorming, and the main outcomes and decisions made were:

- The first epic story would surround the customer's ability to customise their main, select their sides and drinks, and checkout a meal
    - Customisation of both burgers and wraps was chosen as the first user story. We had concerns about this at first since we thought the user story would be too large. In the end, we decided to go with this strategy as it was all one process for the customer.

- The second epic story would surround the customer's ability to check the status of the order after checkout.
- The third epic story would surround staff members viewing the list of orders and updating their status
- The fourth epic story would surround staff members viewing and updating inventory levels

We faced some initial obstacles concerning interpretation of the specification. We spent a decent amount of time discussing the 'checkout' component of the application. We understood that the spec was up for reasonable interpretation, but ultimately were unsure if it included payment.
- After clarification, realised that payment was unnecessary for this project

At the end of this meeting, the scaffold for the Milestone was completed, epic stories were decided upon, user story descriptions were finished.

Responsibilities going forwards:

- Chris will own the user stories related to admin/staff
    - First draft to be completed by (8/03/18)
- Mel and Will will own the user stories related to the customer together
    - First draft to be completed by (8/03/18)

FRI, 8th of MARCH

We had our first call meeting over Facebook messenger. The purpose of this call was to finalise our user stories for the first milestone.

STAND-UP:

- Chris filled in the user stories, priorities, and sizes related to staff abilities. During the call, he will include the specific inventory items as part of the user stories.
- Mel and Will had filled in the user stories, priorities, and sizes related to customer abilities. They will work together during the call to finalise these and clarify error-conditions.

DECISIONS MADE:

- Each customer will only be able to create one main per order.
- Maximum and minimum quantities of buns and patties

During this call, we spent a lot of time discussing the particular language needed for correct user story descriptions. To help us remain on the right track, we colour coded our roles, goals, and benefits in red, green, and blue respectively.

MILESTONE – user stories were completed. Chris submitted this iteration.

TUES, 12th of MARCH

This was our first lab after submitting the user stories. The main feedback we received are as follows:

- Receiving an order ID might be too small to be a user story on its own
- Allowing one staff member to use the admin functionalities at a time will be too hard to implement

Responsibilities going forwards:

- Entire team to get familiar with CRC cards, use-case diagrams, class diagrams
- Will to update the user stories for the next iteration by 19/03/19

TUES, 19th of MARCH

We got feedback for our lab 3 and 4, which consisted of our attempts at writing CRC cards and use-case and class diagrams. The main feedback we received are as follows:

Use Case Diagrams

1. Never have extend/include from any actor, only between use cases.
2. Book car could include entering booking details, maybe rename pay insurance to get insurance, and pay car to authenticate payment details.
3. Credit card company now need not participate in insurance, because we just call it “get insurance”. Payment implied to be handled in authentication or just “pay booking”.
4. Manager should not inherit admin ,they are distinct in the specs.
5. Otherwise this is fine.

CRC

1. No need to say set/get in CRC, knows is just fine here.
2. Put “knows price” and “compute price” in car rather than in all the subclasses. Then you can essentially remove the subclasses from CRC, but keep them in the class diagram.
3. **You are forgetting the system class.**
4. Do not have search result, searching is just a responsibility of the system.
5. Have the system store everything important like bookings, people and cars. It can perform many actions on what it stores which is why it is useful.
6. Booking should know more (customer? Car? Booking details? Like dropoff, price, ect)
7. Just remember not to think physically.

UML Class Diagram

1. Missing cardinality of relationships. Ie. One to One
2. **System Class**
3. If you are going to mention some list put is as car_list: list<Car>, however it is in your unneeded search class for your example.
4. Admin manager don’t need to associate with car, you are thinking too physically. Real admin would just press button in real life calling method on system to generate reports (it stores everything important here).

After receiving extensive feedback, the entire team reviewed existing knowledge of CRC cards, use case and UML class diagrams.

Set the next meeting for the 21/03/19.

THURS, 21st of MARCH

This was our second call meeting.

PURPOSE:

- Review the requirements for milestone 2
- Come to a consensus about the classes we'll include in the diagram

Our first decision in this call was to start by making CRC cards in order to be able to create the UML class diagram with a strong foundation. These are some examples of the draft cards we made:

Order (US1)	
Responsibilities	Collaborator
Knows order ID Knows order status Computes total price	

MainOrder (US1)	
Responsibilities	Collaborator
Computes net price of main order	

Burger (might not be necessary) (US1)	
Responsibilities	Collaborator

OrderSystem (US3, US4, US5)	
Responsibilities	Collaborator
Knows total price Remove items from overview Begin checkout Knows order ID Knows status of order Creates order	Order

DECISIONS MADE:

- The classes that we would proceed with
    - Including their responsibilities and collaborators
- No need to include burger and wrap classes in the UML diagram

We also discussed team availabilities leading up to the next submission date:

- Will has a society induction on Sunday, and a 40% exam on Tuesday
- Melanie has a society induction on Saturday

Responsibilities going forwards:

- Decided to try and get most done before our lab on Tuesday for feedback from tutors and lab assistant
- Set next meeting for 25/03/19

MON, 25th of MARCH

During this meeting, we got started on the UML class diagram.

PURPOSE:

- To have a first draft of the UML class diagram before the next lab

STAND-UP:

- Will revised the use case diagram that we submitted for the week 3 & 4 lab to gain a better understanding of diagrams
- Similarly, Chris revised the UML diagram
- For this meeting, everyone is working together to draft up a UML diagram for milestone 2

For this first iteration, some decisions made were:

- Chris decided in the long term, it would be good to split up price and quantity, and also to introduce a total quantity class. This would be helpful for:
    - Inventory updating
    - Reducing the amount of main order instances
- Mel suggested breaking up Mains and Sides into their own classes, which would calculate the prices of the main and sides separately. This would be helpful for:
    - Separating the customisation business logic to selecting sides
    - Displaying the net price of the main

By the end of this meeting, we realised that a lot of the CRC cards we drew up were probably incorrect. We also discussed the impact of the changing specification in iteration 3 on our model.

Other things discussed were:

- The need to declare separate attributes for each ingredient
    - Ultimately unnecessary
- Maybe combining the price and quantity classes into one
    - Left it separate so we could get feedback for the original idea

We ultimately finished the first draft.

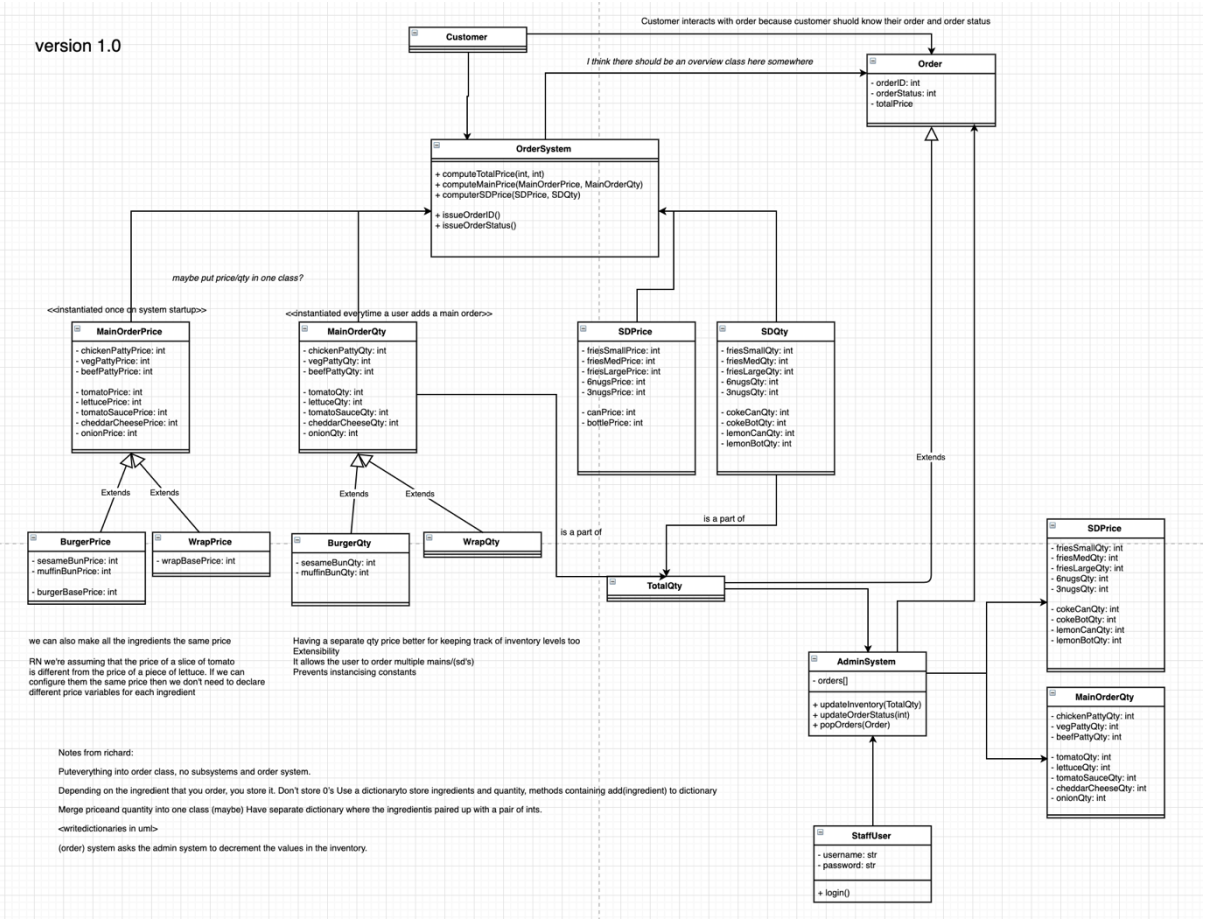
TUES, 25th of MARCH

Chris asked Richard for feedback on the first draft of our UML diagram. The feedback was basically that the diagram was wildly overengineered. The key points of feedback were to:

- Put everything into an order class and system -> subclasses were unnecessary
- Store ingredients depending on what is ordered. Don't store zeros and use a dictionary to store ingredients and their quantities. Methods should be written to update these dictionaries
- Merge price and quantity into one class
- Write dictionaries in UML
- The order system asks the admin system to decrement the values in the inventory

After receiving this feedback from Richard, we set the next meeting for the day after (26/03/19) to discuss and make edits accordingly

MILESTONE - This was the first version of our UML diagram:



WEDNESDAY, 27th of MARCH

This was our in-call meeting after receiving feedback for the first draft of our UML diagram

PURPOSE:

- To reflect on the comments made on the first draft of the diagram, and to make changes accordingly

STAND-UP:

- The last thing everyone worked on was the first draft of the UML
- Chris to go through the comments with Mel and Will

REFLECTIONS:

- The first draft was unnecessarily complex
- Splitting classes into burger price, burger quantity, wrap price, wrap quantity was not needed, and probably incorrect since the methods defined in them can all be done within an order class
- The admin system would not need to manage side prices and main order quantities directly.
- A total quantity class is unnecessary, since we'll be introducing a dictionary

This was our second meeting working together to build a UML diagram. We worked on about four iterations of the diagram.

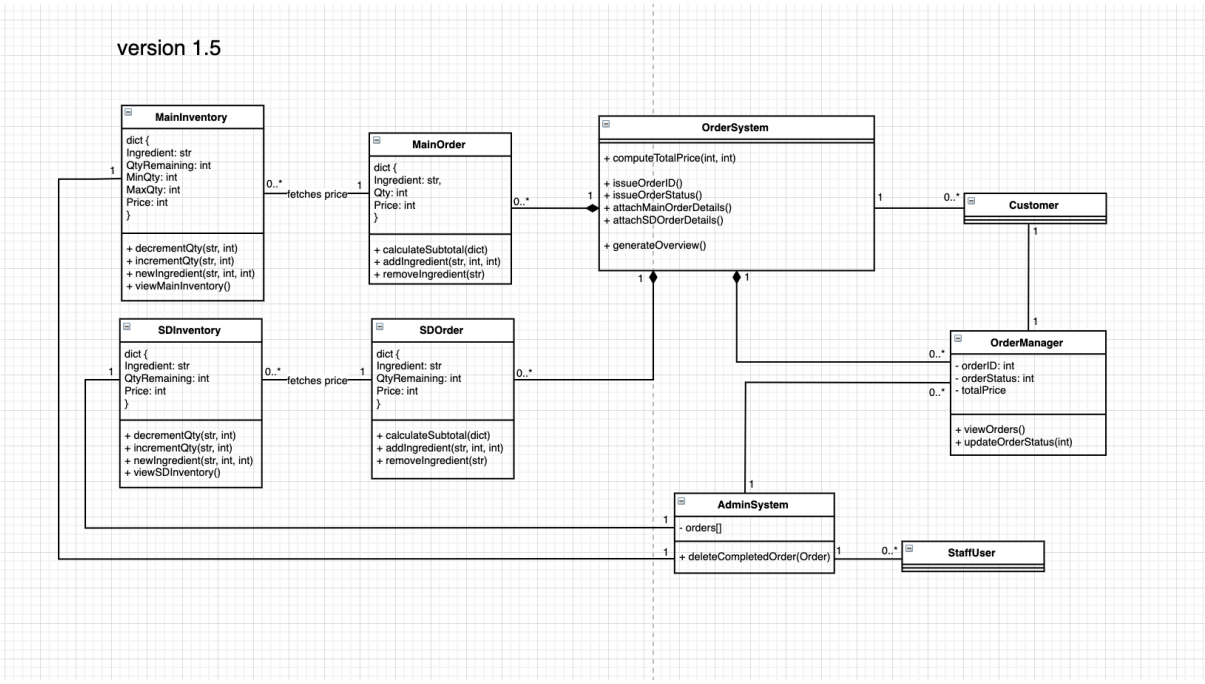
OBSTACLES:

- Using draw.io simultaneously proved to be slightly inefficient, as saving was manual. Sometimes, a lack of communication would result in multiple team members editing the same part of the diagram unknowingly.

Responsibilities going forwards:

- Set a meeting for 30/03/19 for final thoughts and the chance to make edits before submission

MILESTONE – version 1.5 of our UML diagram was the first diagram we all agreed upon after Richard's feedback:



SATURDAY, 30 of MARCH

This was our last meeting before the submission date for milestone 2.

PURPOSE:

- To make final edits to the UML diagram

STAND-UP

- Everyone worked together on the last version of the UML diagram
- Mel wants to bring up some points that we didn't consider earlier

We took this time for a final reflection on the UML diagram. These are the key decisions made:

- Combined the price and quantity classes to track a dictionary
- The Order class is instantiated by the system every time a user starts an order
- Removed the total quantity class
- Introduced idea of having a min and max quantity in the dictionary
- Removed the staff user and customer classes, since they don't need to be managed

Some other things we discussed during this call were:

- Will brought up the issue of how to interpret the base prices of the mains
    - In the end, we had a different base price for each possible main

During this call, we also filled in the multiplicities and cardinalities.

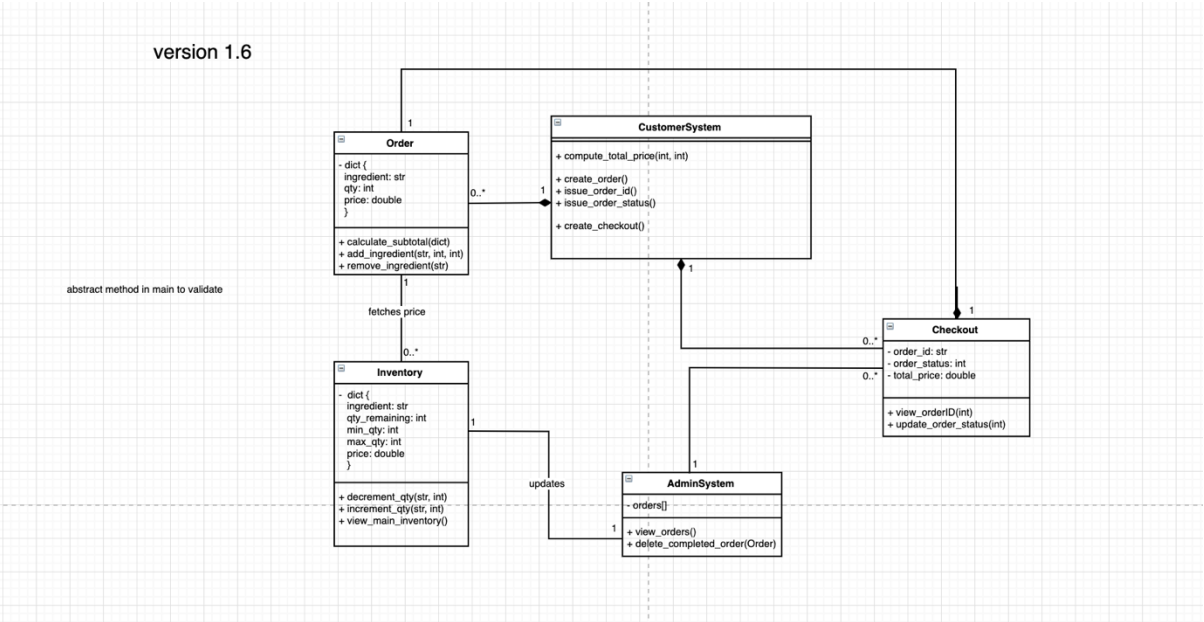
OBSTACLES

- We found that having so many iterations of a single UML diagram became an obstacle. Soon, our whole team was getting confused about what needed to be included.
    - In the future, we discussed that everyone working individually on a diagram and then bringing them together to compare would have worked better, since more perspectives could have been discussed.

Responsibilities going forwards:

- Meet up after the next lab on 02/04/19 for UML feedback so we can begin coding

MILESTONE – this is the final version of our UML diagram that we submitted to GitHub



TUESDAY, 2nd of APRIL

We received feedback on our submitted UML class diagram during this lab. The feedback points raised were:

- You can merge customer and admin system into one system that should contain a bunch of methods that calls a bunch of other methods
- Order object should be to compute its own price
- Each class should have methods to control their own attributes.
- Combine checkout and order into order
- Definitely have a separate class for main.
    - Think of how cars are validated differently in the lab exercises. The different types of mains should be validated similarly as well.

While our first draft of the UML was too complex, the one we submitted was ultimately missing a lot of considerations.

Responsibilities going forwards:

- Chris to revise the UML diagram
- Set a meeting for 03/04/19 to start coding

WEDNESDAY, 3rd of APRIL

This was our first meeting to discuss the coding portion of the assignment.

PURPOSE:

- To assign responsibilities and expectations for the next milestone with regards to the coding portion

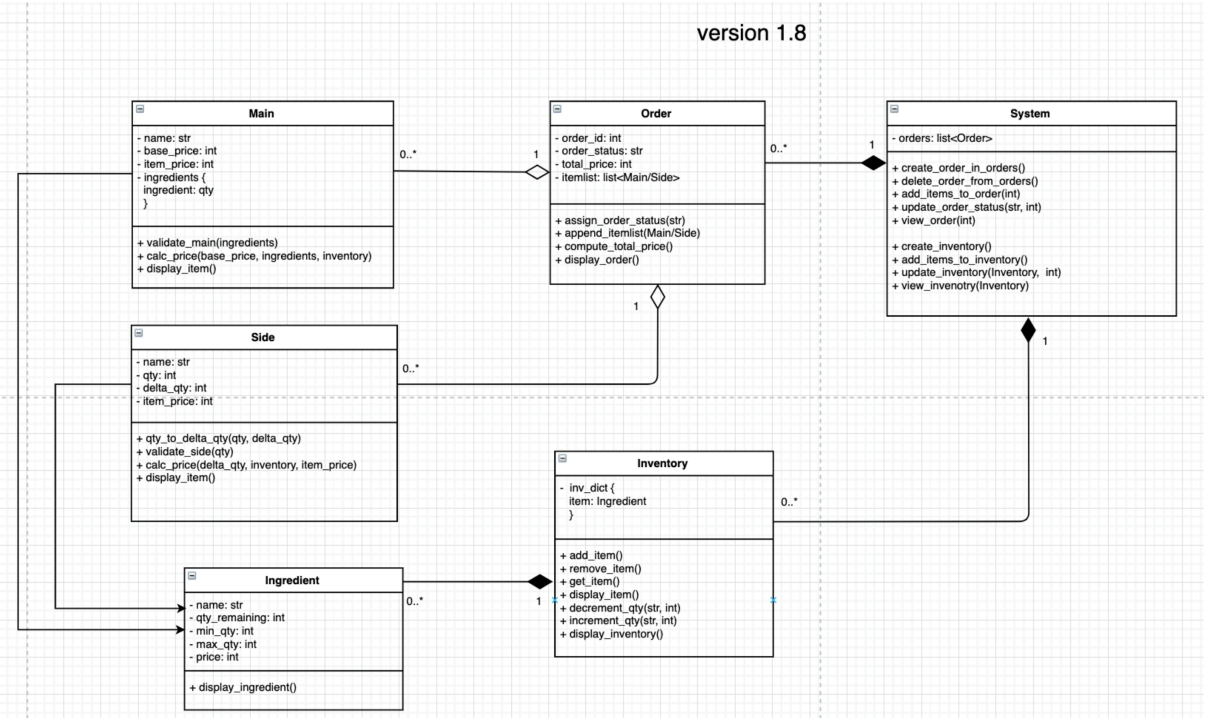
During this meeting, everyone revised their understanding of Git and we set up our branches. We decided that each member would have a branch named after them, which would be merged into master when iterations of code were completed.

Chris set up empty boilerplate files based on revised UML classes on his branch and merged them into master.

Responsibilities going forwards:

- Chris in charge of main and sides
- Mel in charge of order
- Will in charge of inventory
- Set a meeting for 04/04/19 to work together on the code

MILESTONE – this was the final UML diagram after feedback. This is the one we ultimately used to begin coding:



THURSDAY, 4th of APRIL

This was a meeting for everyone to work together while implementing their respective classes.

PURPOSE:

- To have a collaborative meeting where members can discuss and work on their technical implementation together

STAND-UP:

- Chris had started setting up the main.py file
- Mel had started on the order.py file
- Will had nearly finished implementing inventory.py

Will had made some key decisions related to the implementation of the inventory component of our project.

- He decided to introduce an Ingredient class in the same file
    - The purpose of this class was to instantiate an ingredient before placing it into inventory
- He created a setter for quantity remaining
- He also added in validations for inventory, including min and max quantities.

Mel began to write some tests to attempt to follow test driven development.

OBSTACLES

- We found that getting used to python as a language was hindering our performance slightly. We needed to take more time to research its behaviours and inbuilt functions
    - Although more time was taken, this was ultimately a good learning experience.

Responsibilities going forwards

- Set up an in-person meeting for 06/04/19 to finalise the assignment together

SATURDAY 5th of APRIL

This was a meeting to finalise all files that our system would have to import, and to begin writing tests.

STAND-UP:

- Mel had finished the first version of the order components
- Will had finished the first version of the inventory components
- Chris was nearly finished with the main and side components

FINAL CODE DESIGN RATIONALE:

Chris took some inspiration from the in-tutorial car booking project in terms of using arrays to keep track of objects and combined it with our current strategies in using dictionaries to keep track of ingredients.

- There are three main levels of classes for this iteration – Main/Sides, Orders, and the System.
- The system keeps track of all the orders that have been created using a list. It is a higher-level interface to the Orders, Mains and Sides. Orders will be managed via a First-In First-Out basis – with restaurant only being able to process one order, and able to move on only once that order has been completed. This implementation is not efficient but eliminates the need for parallelism-esque algorithms and concurrency management.
- Orders keeps track of all the mains/sides that have been assigned under it through a list.
- Main contains (initialised with an) ingredient dictionary containing all the ingredients in itself. The main class calculates its own price and validates that the dictionary of ingredients is valid.
- Side converts user specified details IE – 1x Six Pack of Nuggets, into a delta that can be adjusted onto inventory levels. The reason for all the subclasses is because of the different rules for conversion from user spec.

OBSTACLES

- After reflecting on the progress of our project, we realised that having vastly different timetables was a bit of an obstacle. It was definitely difficult to find a time where everyone could meet in person, and phone calls lasting into early hours of the morning were becoming more frequent.

MILESTONE – All files are working individually without the system class

SUNDAY 6th of APRIL

This was our last in-person meeting before submitting the coding portion of the assignment.

STAND-UP:

- Mel to finish her tests, finish the log book, and upload the updated UML and user stories
- Will to finish his tests
- Chris to implement system and finish tests

Everyone worked together to finalise pytest files and error files for exception handling.

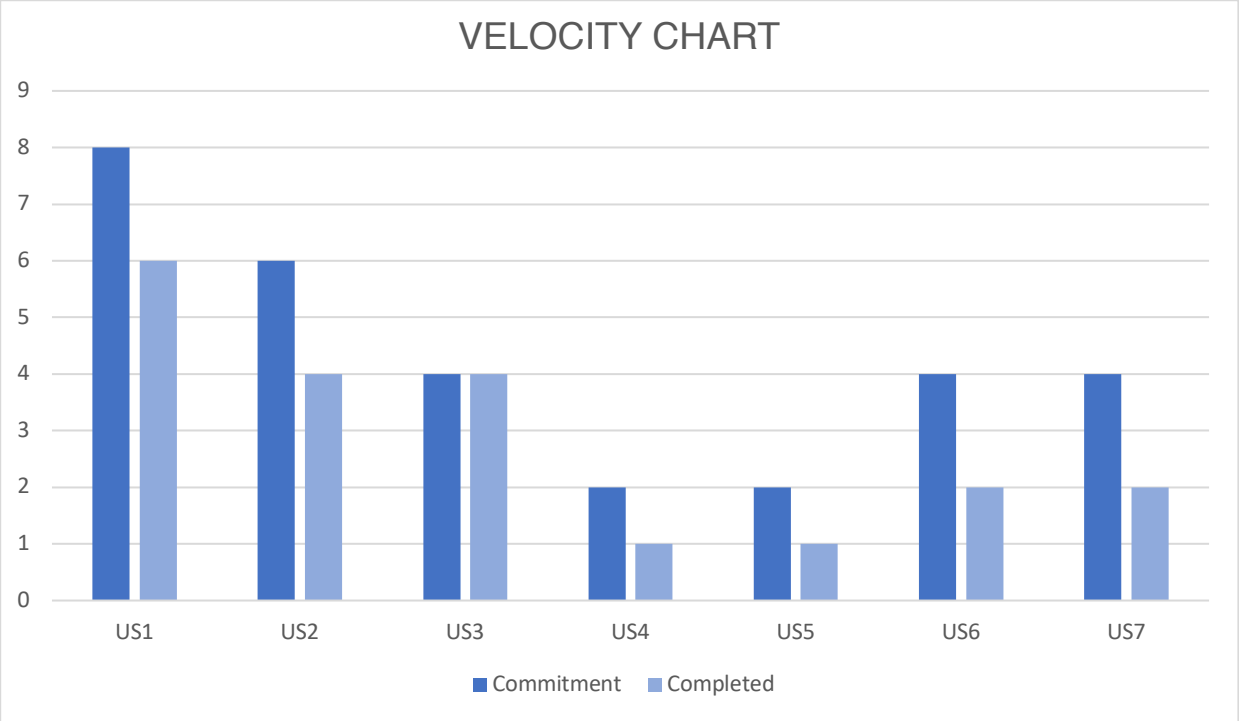
Final edge cases considered and catered for:

- Orders with no buns or patties
- Empty orders
- Two different types of buns or patties
- Inputs that may include negatives or symbols

Responsibilities:

- Everyone to make sure their tests are working for the files they wrote
- Chris to merge a stable master into release
- Mel to update the docs

MILESTONE – Backend system working with tests


VELOCITY CHART
9
8
7
6
5
4
3
2
1
0
US1
US2
US3
US4
US5
US6
US7
Commitment
Completed