University of Toronto Scarborough

Deliverable 5

CSCC01H3 - Summer 2020

Team Members of Team 8:

Alac Wong

Calvin Cheng

Jin Rong (Tony) Cao

Robert Nichita

Winnie Lam

Team TA:

Sally Moon

Table Of Contents

1 - Release Plan

We will continue with our model of having weekly sprints as we want to ensure for consistent sprints throughout this project. This means the sprints begin on Sunday at 5 pm EST and end the following Sunday at 4:59 pm EST for a sprint length of one week. The reason we choose to continue with our one week sprint model is that people, in general, have been known to work better under deadlines, hence the shorter sprints. This also encourages the team to break down stories or features into smaller chunks which makes planning easier and more accurate. With a sprint length of one week, impediments and slowdowns are highlighted more quickly since the team is expected to get feature(s) done weekly. This will allow for greater visibility and understanding of the team's progress within a sprint and our overall project progress.

2 - Product Backlog

2.1 New User Stories Added since Deliverable 4

Since deliverable 4 there has been no new user stories added to the product backlog. This is a result of consulting with the clients consistently throughout the project to ensure that we were building the right product and did not miss any requirements. However, in the product backlog user stories were repointed to account for the fact that team members are now more familiar with the relevant technologies needed leading points to be either over or under pointed initially.

2.2 Existing Product Backlog from Deliverable 4

Here is the backlog that we had from deliverable 4 with certain user stories marked off as completed based on our sprints.

Feature: *Login & Signup*

As a restaurant owner, I want to log in so that I can update my restaurant information. (COMPLETED - SPRINT 2)

- Points: 7
- Priority: High

[Deleted] As Al Capone (system admin), I want to allow customers and restaurant owners to reset their password.

- This user story was deleted due to the fact we used Auth0 to handle login and sign up of users
- Auth0 also handles the process of resetting a password and there's no action required on our ends for the process of login, user sign up and resetting the password

As a customer, I want to be able to use SMS verification for my account.

- Points: 5
- Priority: Low

Feature: *Connecting with Customers*

As Hames Jarden (a restaurant owner), I want to upload videos from files so that I can update my page with videos that I do not want on other platforms.

- Points: 5
- Priority: High

As Hames Jarden (a restaurant owner), I want to embed videos from YouTube so that I can easily link videos on different platforms without uploading twice.

- Points: 4
- Priority: High

As Hames Jarden (a restaurant owner), I want to delete comments from timeline posts so I can remove the comments I do not want. (COMPLETED - SPRINT 6)

- Points: 3
- Priority: Medium

As Hames Jarden (a restaurant owner), I want to delete timeline posts so I can get rid of irrelevant content. (COMPLETED - SPRINT 6)

- Points: 3
- Priority: Medium

As Hames Jarden (a restaurant owner), I want to list and link my social media platforms so that I can connect with customers on different platforms. (COMPLETED - SPRINT 4)

- Points: 1
- Priority: Low

Feature: *Restaurant Profiles (i.e. Address, Menu Items, Reviews, etc...)*

As Robert Downey (a restaurant manager), I want to change items on the menu so that I can adapt to my seasonal menu. (COMPLETED - SPRINT 5)

- Points: 5
- Priority: High

As Steve Hobbs (a small business owner), I would like to have access to restaurants' contact information so that I can call them to modify my orders. (COMPLETED - SPRINT 2)

- Points: 1
- Priority: High

As Bob Lee (a cuisine exploring user), I want to be able to see pictures and reviews of dishes served at a restaurant so I can decide if it suits my siblings' appetites.
(COMPLETED - SPRINT 2)

- Points: 6
- Priority: High

As Rachel Lin (a food ordering user), I want the restaurants to indicate what kind of food they serve so that I can select which different type of food to try.
(COMPLETED - SPRINT 7)

- Points: 1
- Priority: High

As Robert Downey (a restaurant manager), I want to change restaurant information and bio so that I can keep customers informed.
(COMPLETED - SPRINT 6)

- Points: 4
- Priority: High

As Robert Downey (a restaurant manager), I want to have the option to mark a dish as out of stock/sold out on the web page so that I would not have to cancel on a customer.

- Points: 2
- Priority: Medium

As Rachel Lin (a food ordering user), I want to see user ratings of the restaurant so that I can decide to choose to eat there or not. (COMPLETED - SPRINT 2)

- Points: 2
- Priority: Medium

As Robert Downey (a restaurant manager), I want to change the colours/themes of my web page so that it suits my restaurant aesthetics.

- Points: 2
- Priority: Low

As Karen D'Souza (a restrictive food ordering user), I want to have symbols next to restaurant dishes to denote meals that contain common allergies so I am certain that I am not consuming something dangerous.

- Points: 3
- Priority: Low

As Al Capone (system admin), I want to be able to edit the home page content on the site.

- Points: 7
- Priority: Medium

Feature: *Restaurant Setup*

As Robert Downey (a restaurant manager), I want to be able to add items to my menu on set up. (COMPLETED - SPRINT 5)

- Points: 5
- Priority: High

As Robert Downey (a restaurant manager), I want to be able to set up my restaurant page details. (COMPLETED - SPRINT 5)

- Points: 5
- Priority: High

Feature: *Customer Placing Orders*

As Steve Hobbs (a small business owner), I want it to be simple to place orders for multiple dishes or quantities of a dish so that I can order for an entire group of people.
(COMPLETED - SPRINT 7)

- Points: 2
- Priority: High

As Steve Hobbs (a small business owner), I would like to be able to cancel my orders so that I do not pay for incorrect orders.

- Points: 3
- Priority: High

As Bob Lee (a cuisine exploring user), I would like to be able to order from UberEats/DoorDash so that I do not need to leave my house for food.
(COMPLETED - SPRINT 7)

- Points: 2
- Priority: High

As Steve Hobbs (a small business owner), I would like to see the full transaction history on my business' account for tax reasons.

- Points: 4
- Priority: Medium

As Alice Wong (a food ordering user), I want the option to tip so that I can reward good service.

- Points: 2
- Priority: Low

As a customer, I want to be able to track my orders.

- Points: 4
- Priority: Low

Feature: *Restaurant Accepting Orders (Restaurant Dashboard)*

As a restaurant manager, I want to be able to view all orders placed.
(COMPLETED - SPRINT 7)

- Points: 9
- Priority: High

As a restaurant manager, I want to be able to accept orders. (COMPLETED - SPRINT 7)

- Points: 7
- Priority: High

As a restaurant manager, I want to be able to cancel orders. (COMPLETED - SPRINT 7)

- Points: 7
- Priority: High

As a restaurant manager, I want to be able to mark orders as completed.
(COMPLETED - SPRINT 7)

- Points: 3
- Priority: High

Feature: *Search Engines/Restaurant Filtering*

As Bob Lee (a cuisine exploring user), I want to view restaurants that serve a specific cuisine of a dish so that I can explore different cuisines. (COMPLETED - SPRINT 7)

- Points: 1
- Priority: High

As Rachel Lin (a food ordering user), I want to see a list of recommended restaurants and/or dishes so I can order from restaurants that others like.

- Points: 6
- Priority: High

As Alice Wong (a food ordering user), I want to search for food within a certain price range so that I can stay within my budget. (COMPLETED - SPRINT 7)

- Points: 3
- Priority: High

As Alice Wong (a food order user), I want to search for new cuisines, so that I can try new types of food.

- Points: 7
- Priority: High

As Alice Wong (a food order user), I want to search by proximity, so that the food comes quickly to my apartment. (COMPLETED - SPRINT 7)

- Points: 4
- Priority: Medium

Feature: *Mapping Restaurants*

As a customer, I want to be able to view nearby restaurants on a map.
(COMPLETED - SPRINT 7)

- Points: 2
- Priority: Medium

As a customer, I want to be able to view my location on a map.
(COMPLETED - SPRINT 7)

- Points: 2
- Priority: High

Feature: *Specials and Discounts*

As Robert Downey (a restaurant manager), I would like to be able to push special deals further up the search engine, so that they are more visible to customers.

- Points: 2
- Priority: Medium

As Robert Downey (a restaurant manager), I would like to indicate which of the dishes have specials so that it attracts the consumers' attention.

- Points: 2
- Priority: Medium

As Alice Wong (a food ordering user), I want to be able to see discounted foods so that I can stay within my budget for expenses.

- Points: 3
- Priority: Low

As Karen D'Souza (a restrictive food ordering user), I want to view restaurants that cater to a specific dietary restriction such as Vegetarian, Halal or Organic foods so that I can suit my family's dietary needs.

- Points: 3
- Priority: Low

Feature: *Restaurant Recipes*

As Robert Downey (a restaurant manager), I want to upload my family recipes, so that I can pass on the generations of good food to others.

- Points: 4
- Priority: Low

As Bob Lee (a cuisine exploring user), I want to be able to view the recipe of a dish so that I can attempt to recreate it at home.

- Points: 1
- Priority: Low

Feature: *Restaurant Reviews and Favourite Restaurants*

As Alice Wong (a food ordering user), I want to save/favourite my favourite dishes so that I can continue to order it and support the business.

- Points: 4
- Priority: Low

As Janet Jackson (a healthcare worker), I want to rate and leave reviews for restaurants, so that I can voice my opinion on the food I eat. (COMPLETED - SPRINT 7)

- Points: 3
- Priority: Low

Feature: *Platform Analytics*

As Al Capone (system admin), I would like to be able to view statistics and analytics of user data so that I can gain insights into customer usage trends and modify our services accordingly. -

- Points: 7
- Priority: Low

3 - Sprint Plans

3.1 Sprint 5: July 19th to 26th

3.1.1 Sprint Backlog

Feature: *Community Timeline*

[BACK] Create endpoint to add post (to timeline)

- Points: 4
- Priority: Medium
- Acceptance Criteria: None as not a user story

[BACK] Create endpoint to add comments to timeline posts

- Points: 2
- Priority: Medium
- Acceptance Criteria: None as not a user story

[BACK] Create endpoint to delete a comment from a post

- Points: 2
- Priority: Medium
- Acceptance Criteria: None as not a user story

[BACK] Create endpoint to delete posts and its related comments

- Points: 3
- Priority: Medium
- Acceptance Criteria: None as not a user story

Feature: *Restaurant Signup and Setup*

[BACK] Error handling for duplicate restaurants

- Points: 1
- Priority: Medium
- Acceptance Criteria: None as not a user story

[BACK] Fix Twitter and Instagram URL to not be required

- Points: 1
- Priority: Medium
- Acceptance Criteria: None as not a user story

[BACK] Create endpoint for owner information

- Points: 2
- Priority: Medium
- Acceptance Criteria: None as not a user story

[FRONT] Error check for duplicate restaurants on initial set up

- Points: 2
- Priority: Medium
- Acceptance Criteria:

    - Given that a basic user wants to upgrade to a restaurant owner account, when they click upgrade and enter a restaurant email that is already used then they should be provided with a popup message saying the email is already in use and be redirected to the homepage.

Feature: *Image Upload*

[BACK] Fix merge error for cloud storage and documentation
- Points: 2
- Priority: Medium
- Acceptance Criteria: None as not a user story

[BACK] Create endpoint for image upload (NOT COMPLETED - MOVE TO SPRINT 6)
- Points: 5
- Priority: Medium
- Acceptance Criteria: None as not a user story

[FRONT] Guards for restaurant set up process
- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that a non-logged in user or a basic user is logged in, when they attempt to open up the owner setup, menu setup and menu edit page URL then they should be told a message that they do not have the right role and be redirected to the home page.
    - Given that a restaurant owner logs in and has a valid restaurant attached, when they attempt to open up the owner setup, menu setup and menu edit page URL then they should be allowed access to fill out these pages.

[FRONT] Fix owner story set up using the endpoint
- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that a basic user upgrades to a restaurant owner, when they finish filling out all the relevant signup forms then they’re information should be stored in the database.

Feature: *Web Page Design*

[FRONT] Change some text on main page to be relevant
- Points: 1
- Priority: Medium

- Acceptance Criteria:
    - N/A as UI which is not good style to include acceptance criteria for UI

Feature: *Login and Signup*

[BACK] Create endpoint for editing user info

- Points: 3
- Priority: Medium
- Acceptance Criteria: None as not a user story

Feature: *Menu Edits*

[BACK] Create endpoint to edit dishes

- Points: 2
- Priority: Medium
- Acceptance Criteria: None as not a user story

[BACK] Create endpoint to delete dishes

- Points: 1
- Priority: Medium
- Acceptance Criteria: None as not a user story

As Robert Downey (a restaurant manager), I want to change items in the menu so that I can adapt to my seasonal menu.

- Points: 5
- Priority: High
- Acceptance Criteria:
    - Given that Robert Downey navigates to his restaurant page, when he presses the “Edit Menu“ button, then he will be taken to a menu edit page.
    - Given that Robert Downey reaches the menu edit page, when he presses the “Add New Dish“ button, then he should be shown a modal to add a new dish.
    - Given that Robert Downey reaches the menu edit page, when he presses the X button on the dish card, then the dish should be deleted from the menu.
    - Given that Robert Downey reaches the menu edit page, when he presses the Edit button on the dish card, then the edit dish modal should appear and be editable.
    - Given that Robert Downey finishes editing his dish, when he presses the “Save Dish“ button, then the dish should be updated on the menu.
    - Given that Robert Downey finishes editing his menu, when he presses the “Save & Next“ button, then the menu is updated on his restaurant page.

[FRONT] Edit menu set up dishes to allow edits

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - Given that Robert Downey reaches the menu edit page, when he presses the Edit button on the dish card, then the edit dish modal should appear and be editable.

[FRONT] Edit menu set up dishes to allow deletes

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - Given that Robert Downey reaches the menu edit page, when he presses the X button on the dish card, then the dish should be deleted from the menu.

3.1.2 Assignment of Tasks

Below is a rough plan of what each team member during the fifth sprint:

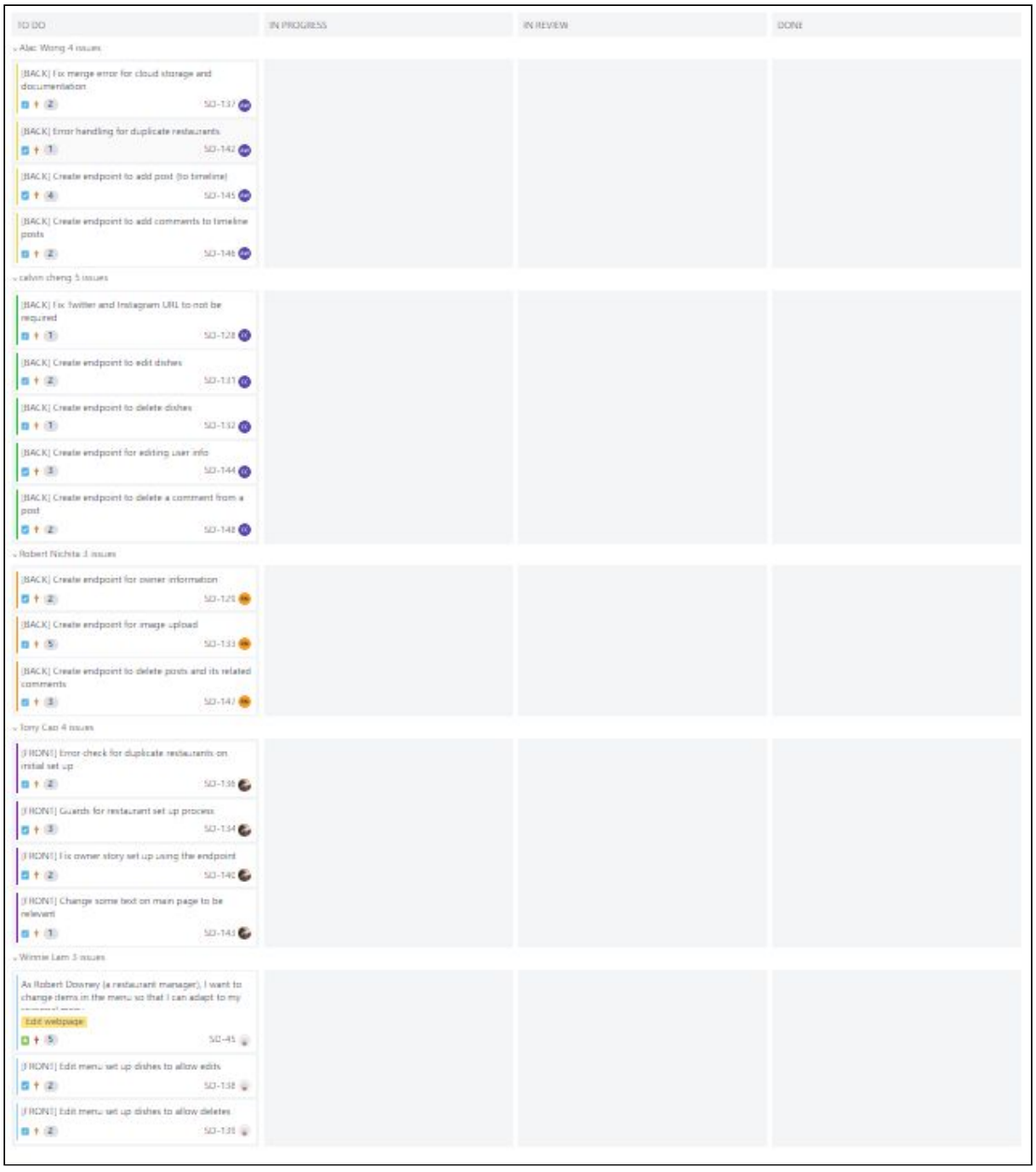
- Winnie Lam: Worked on menu setup such as adding, editing and deleting dishes
- Tony Cao: Stored owner information upon setup, redesigned parts of the home page as requested from the client and worked on ensuring pages are accessible by certain peoples only
- Calvin Cheng: Created endpoints to edit and delete dishes, edit user info and delete comments from the timeline posts
- Robert Nichita: Created endpoints to store owner information from signup and work on image upload
- Alac Wong: Worked on creating endpoints to add posts and comments to the timeline and fixed various errors such as duplicate email for restaurant signup

The order/priority of the tasks for the sprint will be as such (keeping in mind most tasks will happen concurrently):

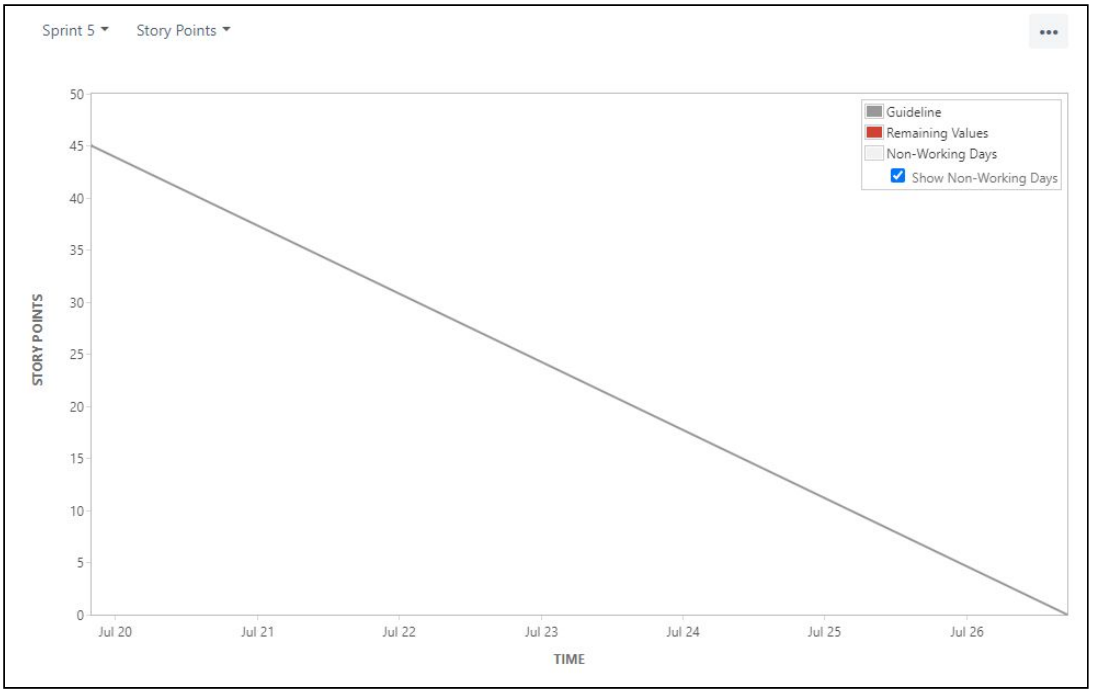
1. Adding, editing and deleting dishes for a restaurant
2. Saving owner and user information
3. Adding and removing comments and posts from timeline
4. Redesigning the webpage to put heavier emphasis on Scarborough owners

Legend
AW = Alac Wong
CC = Calvin Cheng
RN = Robert Nichita
TC = Tony Cao
WL = Winnie Lam

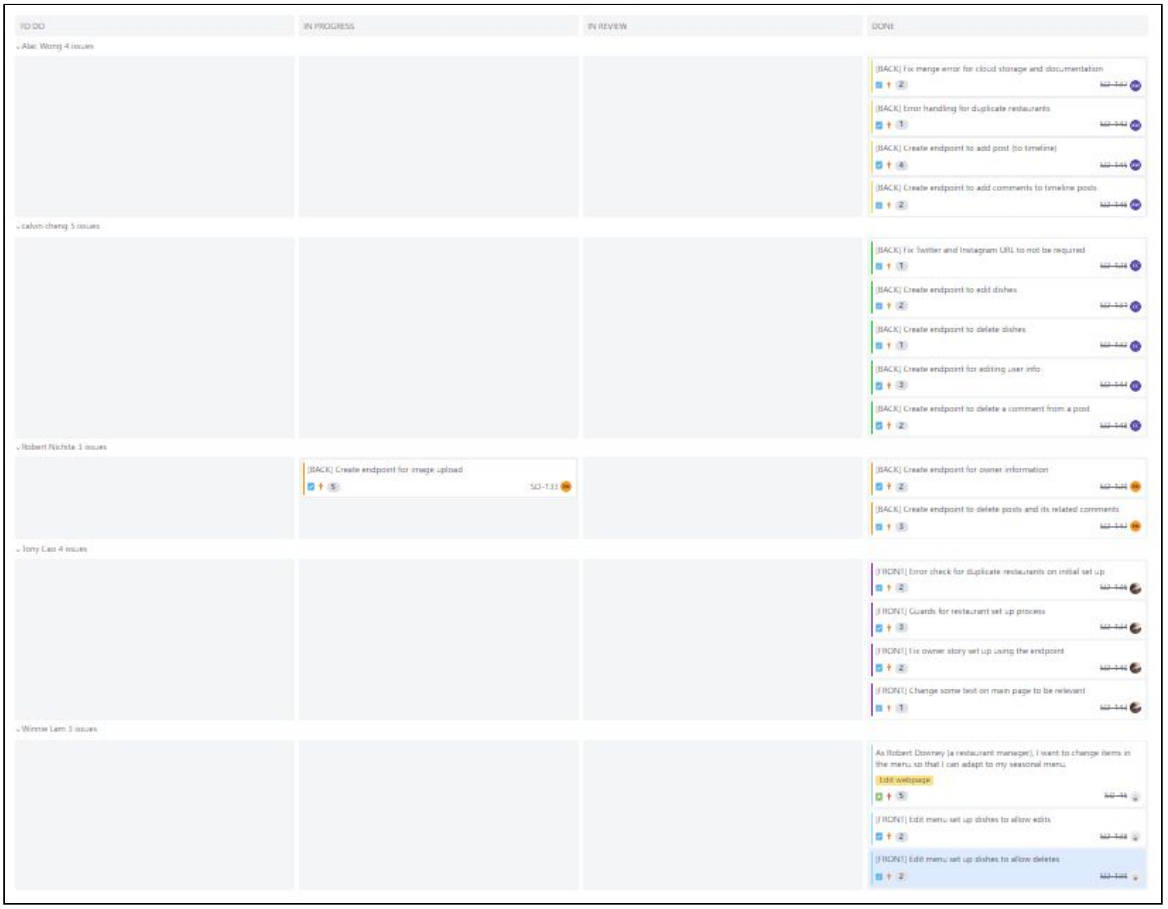
3.1.3 Initial Task Board



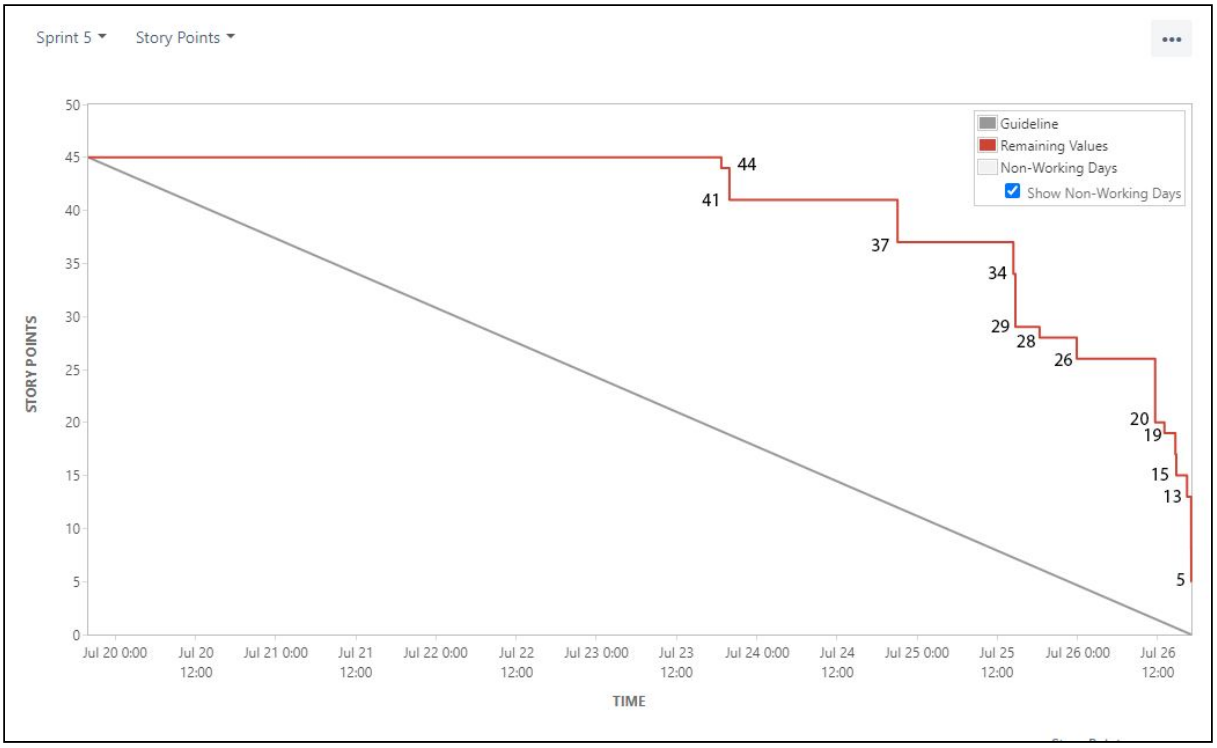
3.1.4 Initial Burn-Down Chart



3.1.5 Final Task Board



3.1.6 Final Burn-Down Chart



3.2 Sprint 6: July 26th to August 2nd

3.2.1 Sprint Backlog

Feature: *Restaurant Details*

[BACK] Create endpoint to get details of a restaurant

- Points: 2
- Priority: High
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

Feature: *Ordering System*

[BACK] Create endpoint to remove from cart

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create order status endpoint

- Points: 4

- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create endpoint to create an order
- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create endpoint to add to cart
- Points: 4
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create endpoint to edit entry amounts
- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[FRONT] Create order page UI
- Points: 4
- Priority: Medium
- Acceptance Criteria:
    - N/A as just a UI that will most likely change

Feature: *Webpage Loading*

[FRONT] Create loading status for HTTP request calls
- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that a user is on the website, when they click the “Browse“ tab, then they should be able to see a loading signal while the page calls the database so that they are not looking at an empty list of restaurants.

Feature: *Media Upload*

[BACK] Create interface for uploading media
- Points: 8

- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create endpoint for image upload
- Points: 5
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

Feature: *Login and Signup*

[BACK] Add new sign up data to the database
- Points: 1
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[FRONT] Add sign up modal after Auth0 signup
- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that a user creates an account when an account is created successfully then the user should see a modal that allows the user to edit and add their name, address, phone number and birthday.

[FRONT] Create guard for user details
- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that the user doesn't fill in the sign up modal upon sign up, when they try to access the order page, then they should be blocked and be told to complete the necessary information done in the profile page.

Feature: *Restaurant Profile*

[FRONT] Edit owner information
- Points: 2
- Priority: Medium
- Acceptance Criteria:

    - Given that an owner wants to edit their owner name and/or story, when they select the edit owner information button, then they should be redirected to a page that shows their current owner information and let them edit it.
    - Given that an owner edits their owner name and/or story, when they click the update button, then they should be redirected back to their restaurant page with their owner information updated.

[FRONT] Edit restaurant information

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - Given that a restaurant personnel wants to edit their restaurant information (name, address, phone number, bio and social media links), when they select the edit restaurant information button, then they should be redirected to a page that shows their current restaurant information and let them edit it.
    - Given that a restaurant personnel edits their restaurant information (name, address, phone number, bio and social media links), when they click the update button, then they should be redirected back to their restaurant page with their restaurant information updated.

[FRONT] Add tab to restaurant page for About Us

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that when a user goes to a restaurant page, when they arrive at that page, then they should be displayed with an “About Us” tab that shows the owner's picture, name and story.

Feature: *Community Timeline*

[BACK] Create endpoint to get all posts and comments

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create endpoint to get all posts and comments for certain restaurant

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create timeline UI

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as just a UI which is bad to include in acceptance criteria

[BACK] Create timeline post card component

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - N/A as just a UI which is bad to include in acceptance criteria

As Hames Jarden (a restaurant owner), I want to post updates to the timeline so I can give updates to my customers.

- Points: 5
- Priority: High
- Acceptance Criteria:
    - Given that Hames signs in, when they click their timeline and click new post to create a new post, then a modal should show up for them to input their post content.
    - Given that Hames finishes creating his content, when he clicks post, then he should see his new post in his updated timeline.

As Hames Jarden (a restaurant owner), I want to add comments to timeline posts so that I can interact with my customers.

- Points: 5
- Priority: High
- Acceptance Criteria:
    - Given that Hames signs in, when they click their timeline and go to any post, then they should be able to enter a comment to the post and see their new comment posted on enter.

As Hames Jarden (a restaurant owner), I want to delete comments from timeline posts so I can remove the comments I do not want.

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that Hames signs in, when they click their timeline and click the trash can icon on their post, then a modal should show up for them to confirm the deletion, and upon confirmation, the post should no longer be in the updated timeline.

As Hames Jarden (a restaurant owner), I want to delete timeline posts so I can get rid of irrelevant content.

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that Hames signs in, when they click their timeline and click the trash can icon any comment under their posts, then a modal should show up for them to confirm the deletion, and upon confirmation, the comment should no longer be in the post in the updated timeline.

3.2.2 Assignment of Tasks

Below is a rough plan of what each team member during the sixth sprint:

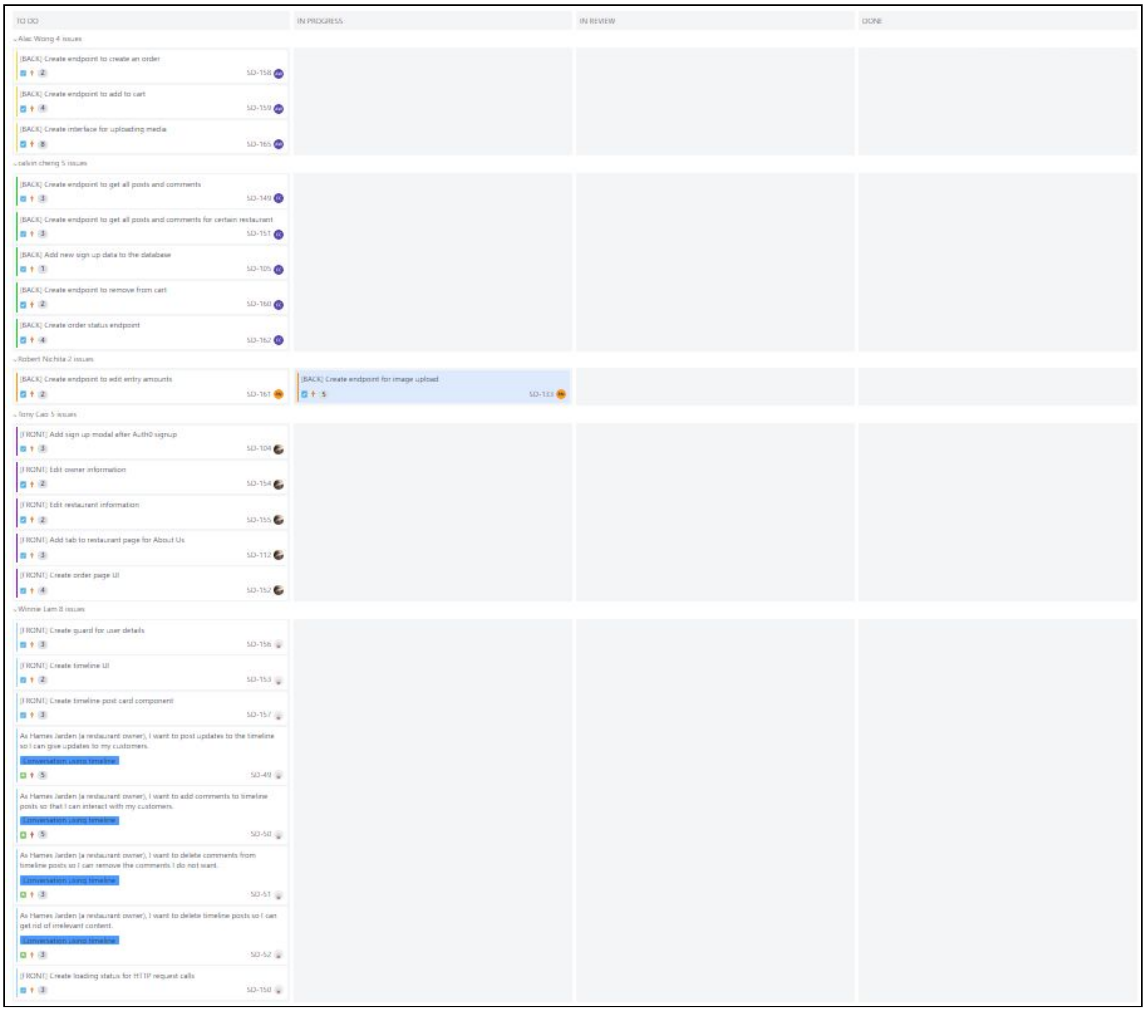
- Winnie Lam: Create the entire timeline process (the UIs, posting/deleting posts and adding/deleting comments)
- Tony Cao: Create a tab for restaurant owner information on the restaurant page, create a UI for the checkout page and work on all relevant edit restaurant info (owner information and restaurant information) which includes the UIs and sending the updated information
- Calvin Cheng: Create endpoints related to timelines (get posts and comments for all restaurants or for a particular restaurant) and endpoints to create orders and add items to the cart
- Robert Nichita: Create endpoint for image upload and endpoint for edit entry amounts in a cart
- Alac Wong: Create endpoints to get details of a restaurant, to delete items from a cart and get order status; Create interface for uploading media

The order/priority of the tasks for the sprint will be as such (keeping in mind most tasks will happen concurrently):

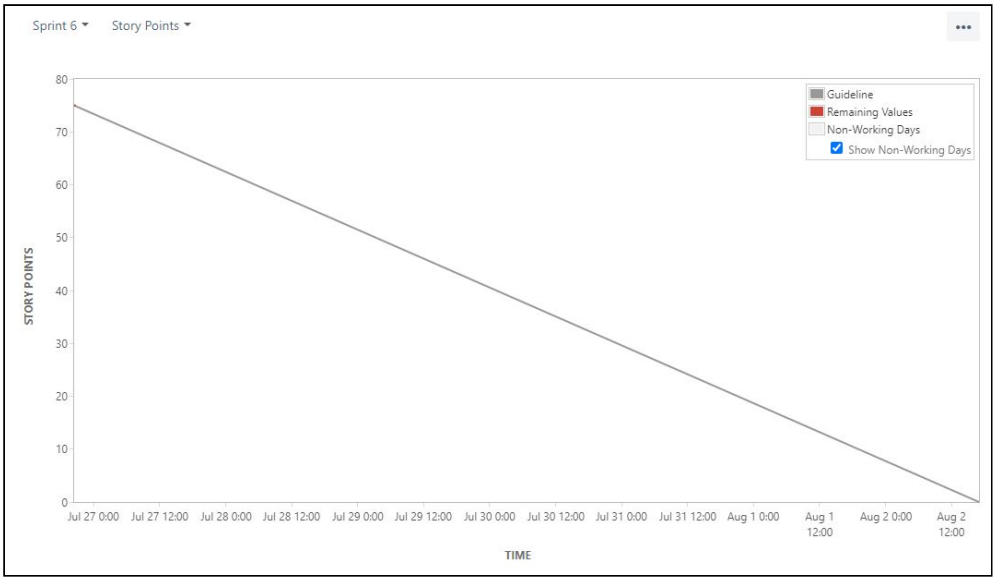
1. Creation of the community timeline platform on the webpage
2. Editing user and restaurant owner information
3. Frameworks for creating orders in the ordering system

Legend
AW = Alac Wong
CC = Calvin Cheng
RN = Robert Nichita
TC = Tony Cao
WL = Winnie Lam

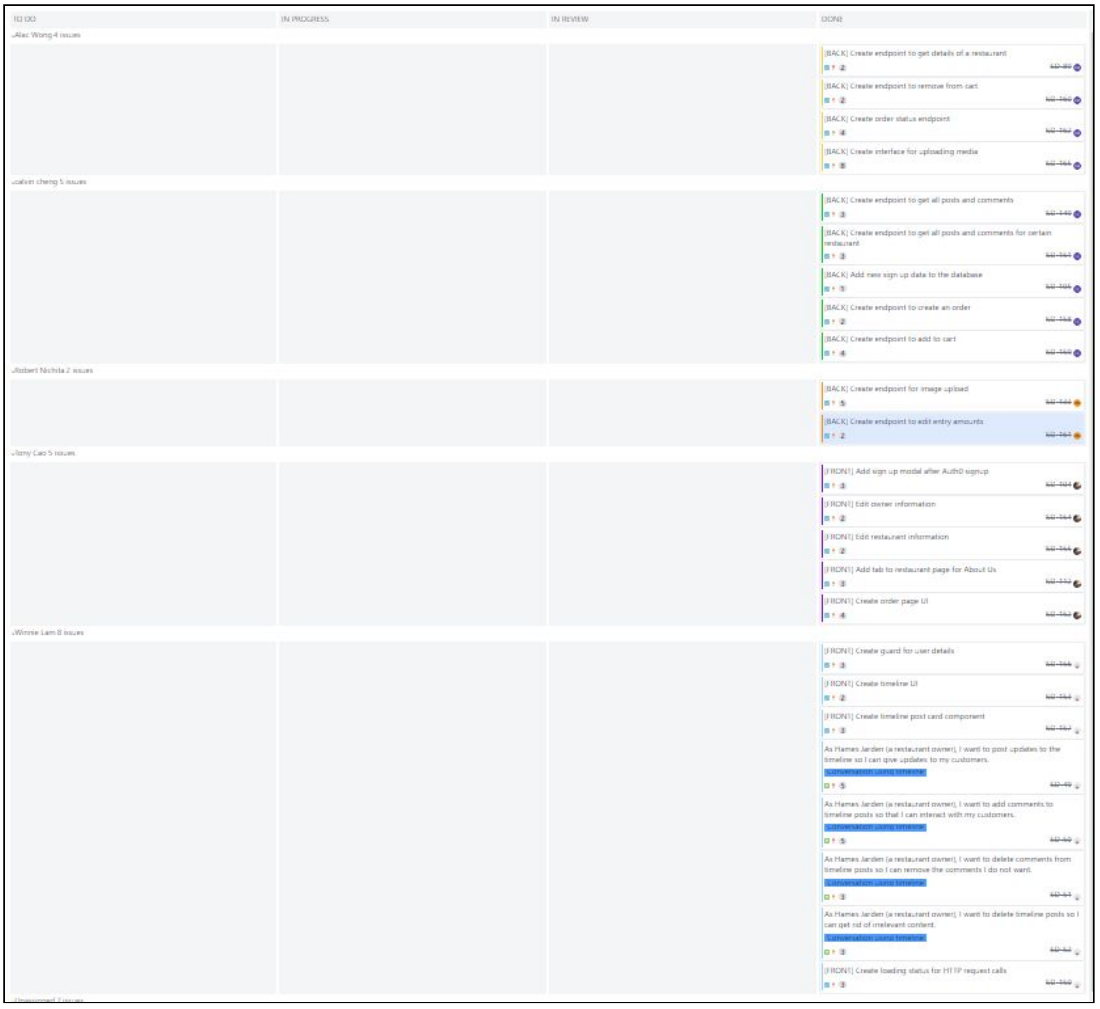
3.2.3 Initial Task Board



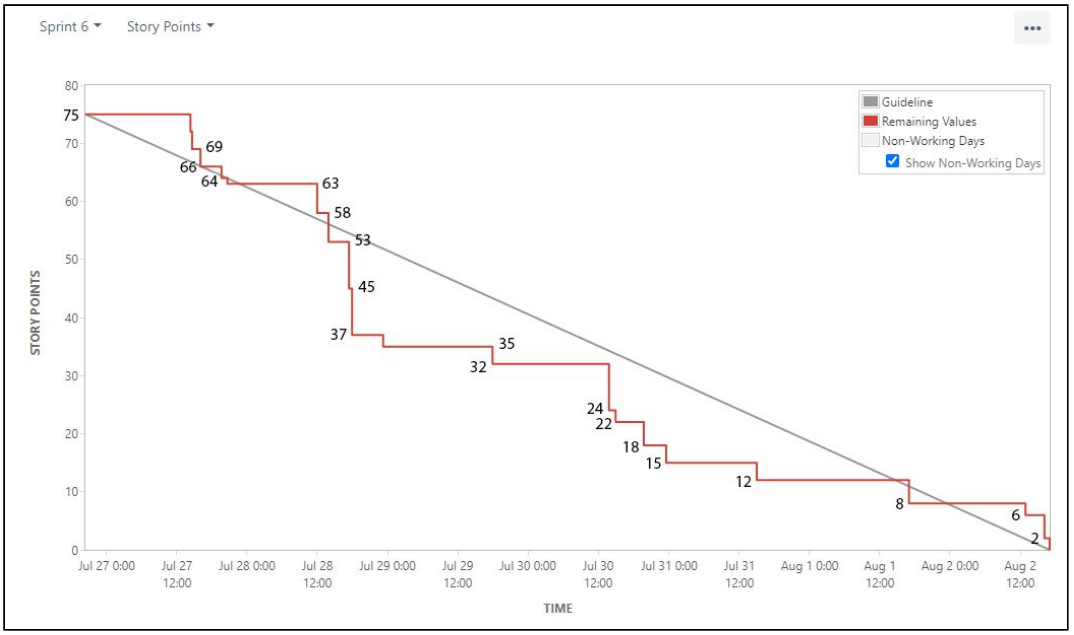
3.2.4 Initial Burn-Down Chart



3.2.5 Final Task Board



3.2.6 Final Burn-Down Chart



3.3 Sprint 7: August 2nd to 9th

3.3.1 Sprint Backlog

Feature: *Community Timeline*

[BACK] Fix create post endpoint to return timestamp

- Points: 1
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Sort post list by descending time stamps

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Clean up time stamp appearance

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

Feature: *Login and Signup*

[FRONT] Show users field validation error

- Points: 6
- Priority: Medium
- Acceptance Criteria:
    - Given that one of the login, signup or restaurant setup forms has invalid data entered, when they submit the form then there should be an indication below the field that the data is invalid and possibly why.

[FRONT] Display user address in navigation bar

- Points: 1
- Priority: Medium
- Acceptance Criteria:
    - Given that a basic user logs in and has an address provided, when they get redirected to the homepage, then they should see their address within the navigation bar.

    - Given that a basic user logs in and does not have an address provided, when they get redirected to the homepage, then they see “no address provided” in the navigation bar.

Feature: *Restaurants by Proximity*

[BACK] Create class to use geocoding API

- Points: 4
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Add geocoding for user and restaurant address

- Points: 4
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

As Alice Wong (a food order user), I want to search by proximity, so that the food comes quickly to my apartment.

- Points: 4
- Priority: Medium
- Acceptance Criteria:
    - Given that a user has allowed the browser to use their location data, when they go to the browse page then the list of restaurants should be sorted by ascending distance from the user.

Feature: *Restaurants Information*

[BACK] Clean the database

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

As Bob Lee (a cuisine exploring user), I want to view restaurants that serve a specific cuisine of a dish so that I can explore different cuisines.

- Points: 1
- Priority: High
- Acceptance Criteria:

    - Given that a user wants to search for a specific cuisine, when they enter a cuisine in the search bar, then they should be shown only restaurants that serve only those cuisines.

As Rachel Lin (a food ordering user), I want the restaurants to indicate what kind of food they serve so that I can select which different type of food to try.

- Points: 1
- Priority: High
- Acceptance Criteria:
    - Given that Rachel wants to browse restaurants, when she selects a restaurant, then the type of food served by the restaurant should be shown.

As Bob Lee (a cuisine exploring user), I would like to be able to order from UberEats/DoorDash so that I do not need to leave my house for food.

- Points: 2
- Priority: High
- Acceptance Criteria:
    - Given that a restaurant owner has a link to an external food platform (like UberEats, DoorDash, etc…), when they fill out the form with the link to the external food platform, then an icon should appear on the restaurant page.
    - Given that a Bob Lee choses a restaurant with an external food platform link, when they click on the icon for the link, then they should be redirected to the respective external food platform link.

[FRONT] Create an all owners page

- Points: 4
- Priority: High
- Acceptance Criteria:
    - Given that a non-logged/basic user wants to view all restaurant owners on Scarborough Dining, when they click the chefs tab, then they should be displayed a list of cards of owner's names, stories and their picture with the ability to navigate to that restaurant page.

[FRONT] Home page fix to populate with database info

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - Given that a user visits the website, when the page loads and they scroll down, then they should see stories and dishes that are generated from the database.

Feature: *Restaurant Menus*

[BACK] Add menu category to dishes

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[FRONT] Fix dish stuff so that it adds category

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - Given that a restaurant owner adds dishes to the menu, when they input a category and save the menu, then the restaurant page should display menu tabs for each specific category for dishes.
    - Given that a restaurant owner adds a dish to the menu, when they input a category and save the menu, then the restaurant page should display the dish under the correct category tab in the menu.

Feature: *Favourite Restaurants*

[FRONT] Create favourites page UI

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as UI which is not good style to include acceptance criteria for UI

Feature: *Image Upload*

[FRONT] Upload and save images to cloud

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - Given that a user wants to upload a photo, when they select the photo upload button, then they should be directed to their local file explorer to search for a photo for upload.

Feature: *Searching Restaurants*

As Alice Wong (a food ordering user), I want to search for food within a certain price range so that I can stay within my budget.

- Points: 3
- Priority: High
- Acceptance Criteria:

    - Given that Alice wants to search for foods of a certain price, when she selects the price range, then only restaurants/dishes of that price range should show up.

As a customer, I want to be able to view nearby restaurants on a map.

- Points: 2
- Priority: High
- Acceptance Criteria:
    - Given that a customer allows the browser to access their location, when they go to the browse page, then they should be displayed with a map containing nearby restaurants.

As a customer, I want to be able to view my location on a map.

- Points: 2
- Priority: High
- Acceptance Criteria:
    - Given that a customer allows the browser to access their location, when they go to the browse page, then they should be displayed with a map containing their location.

Feature: *Ordering System*

[BACK] Create a cancel orders endpoint

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Create a decline orders endpoint

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

As a customer, I want to be able to track my orders.

- Points: 4
- Priority: Low
- Acceptance Criteria:
    - Given that a user has some in progress/ completed orders, when they go to the checkout page then they should be able to see the status of said orders(sent, accepted, declined, completed).

As Steve Hobbs (a small business owner), I want it to be simple to place orders for multiple dishes or quantities of a dish so that I can order for an entire group of people.

- Points: 4
- Priority: High
- Acceptance Criteria:
    - Given that Steve finds a dish that he wants to order, when he tries to place the order, then he should be able to click a counter to increase amounts or manually input in a number and place an order of that amount.

As Steve Hobbs (a small business owner), I would like to be able to cancel my orders so that I do not pay for incorrect orders.

- Points: 2
- Priority: High
- Acceptance Criteria:
    - Given that Steve placed an order that he did not want by accident, when he tries to cancel the order after confirming, then he should have a 30-second window to be able to cancel the order before it gets processed.

Feature: *Restaurant Accepting Orders (Restaurant Dashboard)*

As a restaurant manager, I want to be able to accept orders.

- Points: 3
- Priority: High
- Acceptance Criteria:
    - Given that an order appears on the dashboard under the new order tab, when a restaurant manager presses the accept button, then the order should be updated to reflect its status of in-progress instead of new.

As a restaurant manager, I want to be able to cancel orders.

- Points: 3
- Priority: High
- Acceptance Criteria:
    - Given that an order appears on the dashboard under the new tab, when a restaurant manager presses the cancel button, then the order should be completely removed from the restaurant dashboard.

As a restaurant manager, I want to be able to mark orders as completed.

- Points: 3
- Priority: High
- Acceptance Criteria:

    - Given that an order appears on the dashboard under the in-progress tab, when a restaurant manager presses the complete button, then the order should be updated to reflect its status of completed instead of in-progress.

As a restaurant manager, I want to be able to view all orders placed.

- Points: 3
- Priority: High
- Acceptance Criteria:
    - Given that a restaurant manager wants to view all orders for their restaurant, when they click view all orders, then they should be presented with all orders for their restaurants along with their current status.

Feature: *Restaurant Reviews*

[BACK] Create endpoint to add a review

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[BACK] Average ratings in the back based on reviews

- Points: 3
- Priority: Medium
- Acceptance Criteria:
    - N/A as was task in the back end that did not involve user

[FRONT] Load reviews for each restaurant

- Points: 2
- Priority: Medium
- Acceptance Criteria:
    - Given that a restaurant contains reviews, when a user clicks on the review tabs of the restaurant, then they should be presented with all reviews of the restaurant.
    - Given that a restaurant contains no reviews, when a user clicks on the review tabs of the restaurant, then they should be presented with a message saying no reviews found for the restaurant.

[FRONT] Allow user to add a review for the restaurant

- Points: 2
- Priority: Medium
- Acceptance Criteria:

    - Given that a logged in customer wants to add a review to a restaurant, when they select add review, then they should be displayed with a form to fill out relevant review information (title, review and a rating out of 5).
    - Given that a logged in customer wants to add a review to a restaurant, when they submit their review, then they should be able to see their review for the restaurant.

3.3.2 Assignment of Tasks

Below is a rough plan of what each team member during our seventh sprint:

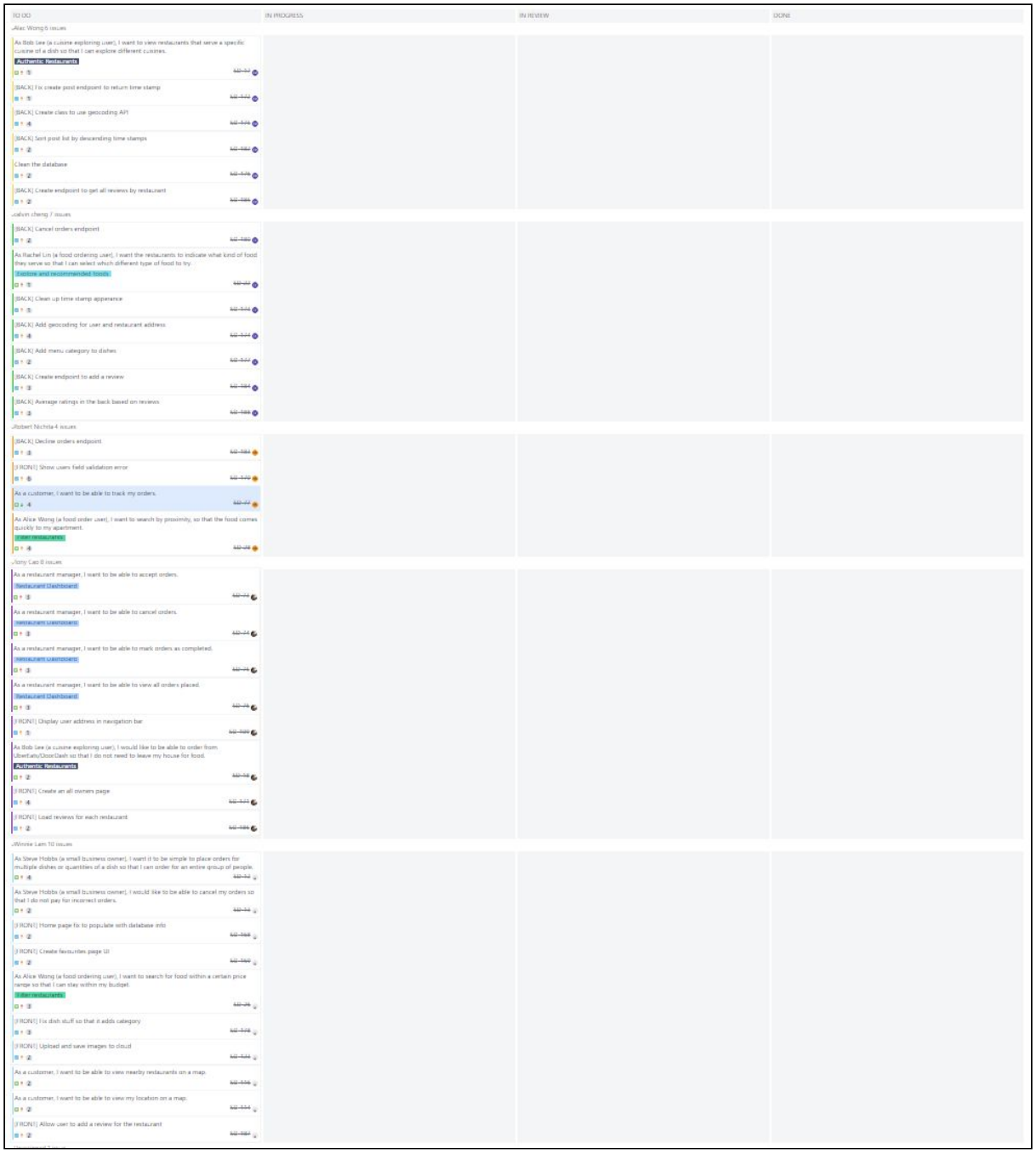
- Winnie Lam: Create filters for browsing restaurants, process image upload in the front, view user location on map and work on the placing order process from user perspective
- Tony Cao: Work on restaurant dashboard (accepting, completing and canceling orders), design an all owners page showing all owners at Scarborough Dining, load reviews for restaurants and edit forms to allow for external delivery links for restaurants.
- Calvin Cheng: Create endpoints to store restaurant reviews, menu category for dishes and create geocoding for user and restaurant address
- Robert Nichita: Create endpoints to decline orders, do validation of inputs on forms, show order track for customers and sort restaurants by proximity.
- Alac Wong: Create endpoints to get reviews by restaurants, class to use geocoding API and work on the cleaning of the database

The order/priority of the tasks for the sprint will be as such (keeping in mind most tasks will happen concurrently):

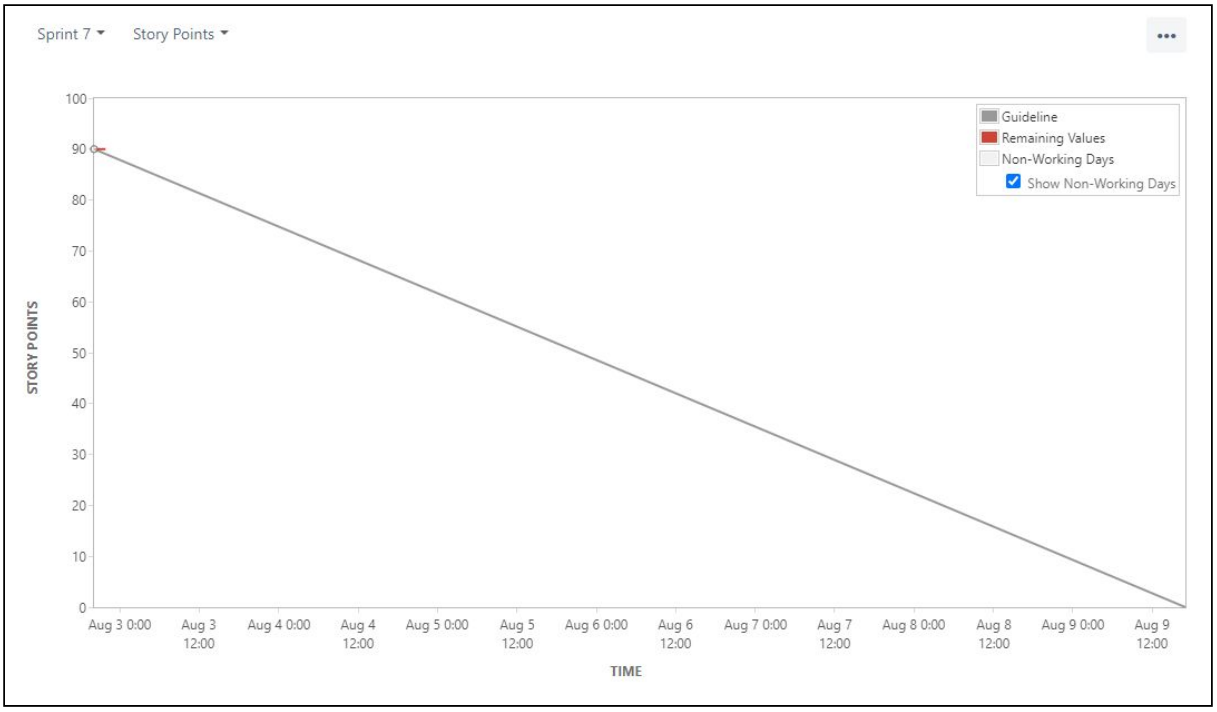
1. Image upload
2. Ordering system
3. Geolocation of user and restaurants
4. Restaurant reviews

Legend
AW = Alac Wong
CC = Calvin Cheng
RN = Robert Nichita
TC = Tony Cao
WL = Winnie Lam

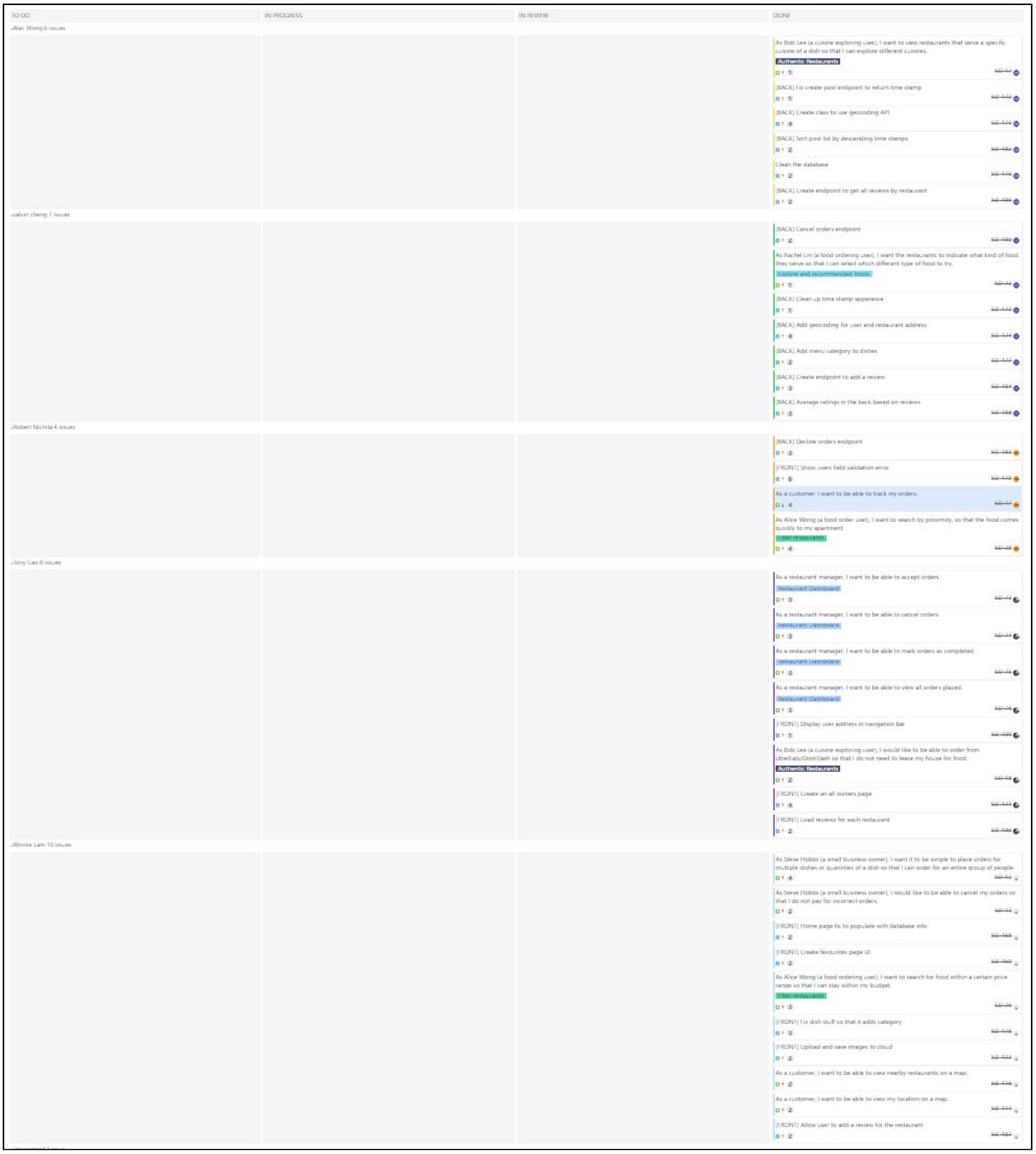
3.3.3 Initial Task Board



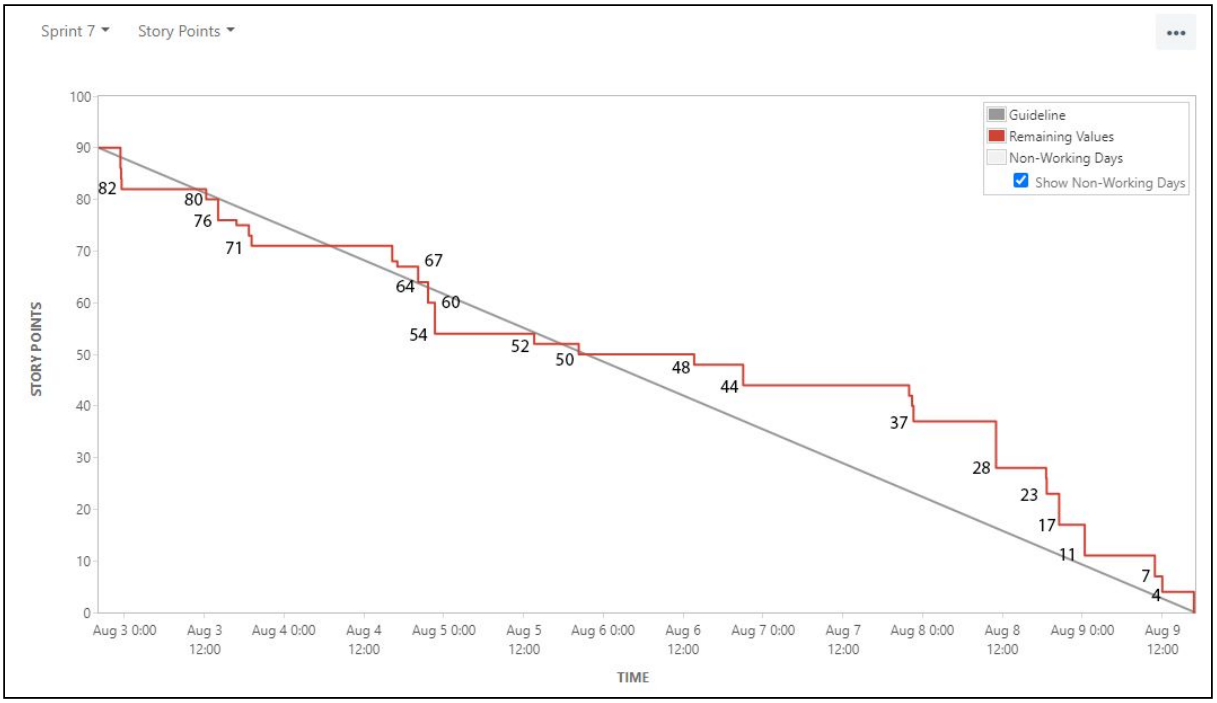
3.3.4 Initial Burn-Down Chart



3.3.5 Final Task Board



3.3.6 Final Burn-Down Chart



4 - System Components

4.1 Change in Components

In terms of changes in the system components, quite a few changes were incurred , and thus our previous system component diagrams failed to accurately capture our current software architecture. We have updated our component diagram to better reflect our actual design below. For our backend, our design consists of a model view controller paradigm. For components Restaurant, Reviews, User, we have a model component that connects to our database in the mongodb and views which manipulates such data and exposes itself to the corresponding frontend page component who displays the data. Our backend also includes several modules such as Geolocation and Media Upload component which communicates with its respective apis (file storage and maps) and is used by other backend components for tasks such as updating cover images and geocoding. Our frontend consists of different page components (Restaurant page, User page, Timeline page, and Order page which includes a service module that connects to a backend view for data. Each page also includes a browsing component and an editing component for displaying and modifying user data. A more thorough description of changes is found below in section 4.3.

4.2 Updated UML Diagrams

See attached file in our deliverable 5 folder called UML_Use_Case_Updated.pdf for our UML use case diagram for the Scarborough Dining system. This UML use case diagram is the same one submitted in deliverable 4 as there has not been any new personas or major functionality changes since deliverable 4.

4.3 Updated Component Diagrams

See attached file in our deliverable 5 folder called SCDining_Component_Diagram_Updated.pdf for our components diagram for the Scarborough Dining system.

Paragraphs for changes to the component design can be found below:

Geolocation component: There has been a request to add a Geolocation service to the application to allow users to see restaurants nearby on the map. This required us to implement a map API on the front (for map display and latitude/longitude parsing) and a geocoding API in the backend. In the frontend, this map API was hidden behind a UI map component which can be embedded anywhere on the website. The map also uses a javascript browser API in order to get the user's current location so it can be displayed on the map. In the backend, the API was implemented into a geo_controller class which handles all of the geocoding and validation, it can be used generally for all geocoding requests.

Media Upload component: This component handles our application's media upload. This component acts as an interface for uploading media to the cloud. This component includes a view component to receive requests from the Frontend service module and calls the Media factory module to create the correct type of Media component to complete the request such as updating a file for a database entry.

Media Factory Component: This component handles importing different the correct media component given a corresponding request.

Media Component: This component is an implementation to the Media Upload Component where it uploads files to the cloud and receives a link to that file. It then updates a database entry from its respective database model with a link to that media file. Different data models have different locations for files such as restaurant logos for restaurants and pictures for users, and thus the interface handles the differences when updating the database entry.

Database Models: This component is an abstraction which is a part of django. Database Models allow us to specify pre-programmed or custom-made database types. Django then

performs *database migrations* automatically based on these types to create new collections, modify fields, and change field restrictions present in the database. The models also allow us to perform queries using a standard set of functions provided by django's *Object Relational Model* (ORM) which will translate these python functions into queries, creating a standard way to interact with the database regardless of which model we use. Inside model classes is where we implement most of our business logic which interacts with the database, so that we can use the model to provide pre-written queries for responding to requests. Database Models also allow several convenient command-line options such as dumping data from the database, which was a feature that our client requested.

View Models: This component is an abstraction which is a part of django. View Models allow us to respond to incoming HTTP requests in a standard format. It sends in a ***Request*** object with all of the request data and metadata parsed onto it, we then read this request object to handle the request, and return a ***Response*** object which has several subclasses for different response methods i.e. (HTTPResponse, JSONResponse, etc.). This makes it easy to maintain a request-response architecture. The views are usually where we serialize data which is acquired from a Database Model.

URLS: This component is an abstraction which is a part of django that we did not include in our component diagram to keep it readable. URLS allow django to decide which View Model will handle a certain request URL, i.e. it abstracts our ***Routing*** functionalities out so that we can give it an arbitrary url string or even pattern matching queries(we did not use this feature) to ensure the URL format is correct. It then sends the request to the correct View Model instance to be handled.

Ordering System: This was separated from the payment component and actually implemented(payments were mocked). The ordering system allows users to place orders through the checkout UI, and also allows restaurant owners to view and dictate the progress of their sent orders through the restaurant dashboard. Each user only has one active cart at a time, and they can:

- Clear their carts (deletes cart from database)
- Send their carts to the restaurant (sent status, payment is on hold)

There is a 30 second timer which allows the user to stop sending their order if they wish.
The Owner can:

- Decline the cart(cancelled status)
- Accept the cart(in progress status, payment would be confirmed here, cannot decline)
- After which they Complete the cart(completed status, payment is received by restaurant here)

Frontend Component-Service model: This describes the general way in which the frontend interacts with the backend, angular has an ***HTTPClient*** abstraction which allows the frontend to make HTTP requests using simple typescript objects. We contained all of our uses of this

abstraction inside of various *services* which are then called from various related UI components in order to make requests to the backend

Below are the paragraphs for the remainder of the components once again:

Authentication component: This component is responsible for securely authenticating the user and then retrieving/storing user data to all other components except the database component. This component calls the Database component to retrieve data of a logged-in user or store data of a new user. This component will interact with both customers and restaurant personnel as they will be using the same login and sign up page but given access to only pages that their specific profile can view.

Database Component: This component is responsible for storing user data, restaurant data and maintaining a collection between restaurant pages and subsequent tags. All other components except the restaurant dashboard call this component to either store, retrieve, or update data. This component will interact with both customers and restaurant personnel as they will be viewing all the items on each page based on the database population.
This design was more of a high level idea, it does not represent our code structure accurately. For a more accurate representation of the database interactions see "Database Models" above.

Page editing component: This component is responsible for allowing a restaurant owner to successfully update their website. After updating their page, this component is then responsible for updating tags associated with the modified website. This component calls the Database component's restaurant table to update website data, and then update corresponding tags for updated websites. This component will only interact with restaurant owners as they will be the ones who will be able to make changes such as menu changes, limited-time specials or profile changes such as phone number, address or operating hours.

Homepage Component: This component is responsible for providing the user with an attractive interface that features different restaurants and dishes to the user. This component calls the database component to get different restaurants and search for specials from the restaurant table. This component will interact with both customers and restaurant personnel since it will be the first page they land on when going onto the site.

Page Component: This component is responsible for providing the user with an interface to view a restaurant's page as well as providing interactions such as favoriting and reviewing that page. This component interacts with the database component by loading the page data as well as updating its user reviews from the restaurant table and updating a user's favorites from the user table. This component will interact with mainly customers only as this page will be providing useful information that a customer may want to know before placing orders

such as what other users thought of the restaurants or which restaurants they have enjoyed in the past.

Filter Component: This component is responsible for providing the user with an interface for searching for different restaurants or dishes. This component calls the tag collection table from the database component to find relevant restaurants to display. This component will interact with customers only as this provides customers a way to narrow down a long list of restaurants to restaurants that either cater to their taste, budget or is within proximity.

Payment Component: This component is responsible for providing the user with an interface to pay for orders, track transactions and send data to the restaurant dashboard. This component calls the database component's transaction table to record transactions and sends payment order requests to the restaurant dashboard. This component will interact with both customers, restaurant personnel and financial institutions. When customers pay for an order, a transaction will be sent to the financial institution to process payment. Then upon successful payments, the financial institution will send the information back to the restaurant who can then accept the order and process it.

Restaurant Dashboard: This component is responsible for providing the restaurant owner with an interface to accept/decline orders. This component receives data from a web socket from payment components and sends data back to the payment component from its socket. This component will interact with the restaurant personnel since it allows restaurants to interact with orders placed by customers.
This design was included in orders, we were not able to add websockets due to their async nature being difficult to facilitate with our request-response model. For a more accurate representation of the order interactions see "Ordering System" above.

5 - Verification

5.1 Testing Strategies

5.1.1 Backend

When writing code, manually testing the happy path is done on the spot using Postman or Insomnia (depending on the dev) to determine if the code works at all. After passing the initially manual testing we move on to writing more in depth test cases by writing automated unit tests. The automated test cases use mock returns for URL validation (Checking if a URL is valid). Moving on to the pull request, we list out how to manually test the endpoint including the fields needed and the url it should be sent to. We make sure to manually test all the blankable fields and the test cases. Also a table is kept for test cases for backend points in the documentation.

5.1.2 Frontend

Due to lack of time to research and implement frontend to end testing, we decided to stick with manual testing of the frontend component and user interface. For each pull request, we list out the test cases and instructions for the reviewer on how to use the code to see if the website performs the expected behaviour. We kept a table for test cases for front end pointents in the documentation.

We also perform cross-browser and cross-system tests, by testing on Google Chrome and Firefox, and on Windows and Mac. By testing on different browsers, we realize what other specific code we need to add in order for it to work on different platforms.

5.2 Backend Unit Tests

Documented test cases can be found in server/{app}/test**.py where app is the corresponding app to test case.

Specific apps, test suites, or even individual test cases can be run using the format: "python manage.py test App.File_Name.Test_Suite.Test_Case" depending on how deep you want to go.

5.2.1 Testing Table Legend

Column	Column Description
Test Case Name	Name of Test Case
App	Django app test case is testing
Test Suite	Test Suite for test case
Evaluation Criteria	Criteria function must pass to pass the test case
(Possible Risk) Description	A description of possible risks associated with test case failure
Magnitude	Measurement of how dangerous possible risks associated with test case failure
Probability	Measurement of how likely possible risks may occur associated with test case failure
Priority	Priority of importance for a function to pass test case, priority is influenced by probability, magnitude and the function's priority (found in backlog) it is testing

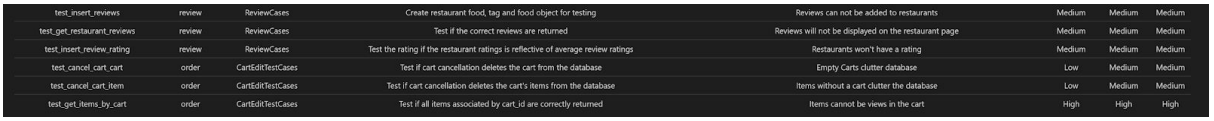
5.2.2 Master Testing Table

Our nicely formatted master testing table in a Markdown format can be found in our repository in the following section: team_08-project/sd-site/docs/tests/back-tests.md

However, below is an export of that table (in the format of screenshots due to the number of columns in the table:

Master Testing Table

Test Case Name	App	Test Suite	Evaluation Criteria	(Possible Risks) Description	Magnitude	Probability	Priority
test_signup	user	SDUserTestCases	Checks to see if The User has been inserted into the database	No new Users can be made	High	High	High
test_signup_invalid_role	user	SDUserTestCases	Checks to see the proper Validation Error is thrown	Anything can be used as a role	High	Low	Medium
test_reassign_RO_to_BU	user	SDUserTestCases	Checks if the role change to BU is reflected in the database	Users won't be able to be 'demoted'	Low	Low	Low
test_reassign_BU_to_RO_User	user	SDUserTestCases	Checks if the role change to RO and new restaurant id is reflected in the database (user side)	New ROs won't be able to be created	High	Low	Medium
test_reassign_BU_to_RO_Restaurant	user	SDUserTestCases	Checks if the role change to RO and new restaurant id is reflected in the database (restaurant side)	New Restaurants won't be able to be created	High	Low	Medium
test_data	user	SDUserTestCases	Checks the user data returned is the matching the queried user	Display incorrect data for users	High	High	High
test_exists_true	user	SDUserTestCases	Checks if an existing user exists	Users will be unable to upgrade	High	High	High
test_exists_false	user	SDUserTestCases	Checks if a non-existing user exists	Unable to sign up users	High	High	High
test_edit_user_valid	user	SDUserTestCases	Given new user data, user document is updated to represent new data	Users will be unable to customize their profile	Low	Low	Low
test_edit_user_invalid	user	SDUserTestCases	Test if invalid fields are returned	Incorrect invalid fields will be highlighted	Low	Medium	Low
test_add_randomizer	utils	SeederTestCases	Adds a new {"key": randomization_function()} pair into the seeder dictionary	Incorrectly add new randomization functions to seeders	Low	Low	Low
test_gen_rand_dict	utils	SeederTestCases	Generates random data based on given seeding dictionary	Inability to randomly generate data for seeding	Low	Low	Low
test_clean_dict	utils	SeederTestCases	Cleans invalid randomization functions (Non JSON-Encodable outputs) from a given seeding dictionary	Potentially broken seeding scripts with functions that produce non JSON-encodable outputs	Low	Low	Low
test_clear_tags	restaurant	TagClearCases	Tag ids are correctly purged from a Food document	Tags remain in database which may result in referencing non-existent tag documents	Medium	Medium	Medium
test_clear_foods	restaurant	TagClearCases	Food ids are correctly purged from Tag document	Foods remain tagged which may result in search engine displaying non-existent/incorrect documents	Medium	Medium	Medium
test_food_ids	restaurant	AddTagCases	Tag ids are correctly updated from tagging an existing Tag document	Food will not be associated with subsequent tag which may cause incorrect search engine results	Medium	Low	Medium
test_tag_ids	restaurant	AddTagCases	Tag ids are correctly updated from tagging an existing Tag document	Restaurant owners will be unable to tag their dishes resulting in search engines skipping their dishes	Medium	High	Medium
test_tag_creation	restaurant	AddTagCases	Tag document is correctly generated upon tagging with a "new" tag word	New Tag documents will not be generated, resulting in limited search engine results	Medium	Low	Medium
test_foods_already_tagged	restaurant	AddTagCases	Food ids are not duplicated upon tagging an already tagged (Food, Tag) couple	Duplicate food ids take up extra space in the database and slow down querying	Low	Low	Low
test_tags_already_tagged	restaurant	AddTagCases	Tag ids are not duplicated upon tagging an already tagged (Food, Tag) couple	Duplicate tag ids take up extra space in the database and slow down querying	Low	Low	Low
test_auto	restaurant	AutoTagCases	Correct Tag document correctly automatically generated based on Food's description	Search engine results become slowly reliant on user input and cannot provide robust results to the user	Medium	High	Medium
test_insert_food_valid	restaurant	FoodTestCases	Given food data, restaurant document is inserted into database representing said data	Restaurants will not be able to upload dishes onto their menu	High	High	High
test_insert_food_invalid	restaurant	FoodTestCases	Test if invalid fields are returned	Incorrect invalid fields will be highlighted	High	High	High
test_get_all_foods	restaurant	FoodTestCases	All food documents within the database are correctly retrieved	Frontend will be unable feature dishes on the homepage	Medium	High	Medium
test_get_by_restaurant	restaurant	FoodTestCases	All food documents for a restaurant within the database are correctly retrieved	Frontend will be unable show each restaurant's page/menu	High	High	High
test_delete_food	restaurant	FoodTestCases	The Food object is correctly wiped from the database	Restaurant Pages will have previously deleted dishes	Medium	Medium	Medium
test_edit_dish_valid	restaurant	FoodTestCases	Given new food data, food document is updated to represent new data	Restaurant Owners will be unable to edit the information of their dishes	Medium	Medium	Medium
test_edit_dish_invalid	restaurant	FoodTestCases	Test if invalid fields are returned	Incorrect invalid fields will be highlighted	Medium	Medium	Medium
test_find_restaurant	restaurant	RestaurantTestCases	Correct restaurant document is retrieved given primary key 'id'	Frontend will be unable to documents associated with that specific restaurant such as dishes and users	High	High	High
test_find_all_restaurant	restaurant	RestaurantTestCases	All restaurant documents are retrieved from database	Frontend will is unable to display restaurant data	High	High	High
test_insert_restaurant_valid	restaurant	RestaurantTestCases	Given restaurant data, restaurant document is inserted into database representing said data	New restaurants cannot be added to the database	High	High	High
test_insert_restaurant_invalid	restaurant	RestaurantTestCases	Test if invalid fields are returned	Incorrect invalid fields will be highlighted	High	High	High
test_edit_restaurant_valid	restaurant	RestaurantTestCases	Given new restaurant data, restaurant document is updated to represent new data	Restaurant data becomes static and cannot be changed by restaurant owner	Medium	Medium	Medium
test_edit_restaurant_invalid	restaurant	RestaurantTestCases	Test if invalid fields are returned	Incorrect invalid fields will be highlighted	Medium	Medium	Medium
test_add_menu_category_restaurant	restaurant	RestaurantTestCases	Test if the menu category is added to the restaurant object when a dish is added	Will not be able to display the menu properly under each menu category	High	High	High
test_empty_menu_category_restaurant	restaurant	RestaurantTestCases	Test if there the menu is deleted from the restaurant object if there are no more food items with the category	Will include empty menu categories on menu page	Low	High	Medium
test_upload	timeline	PostSuite	Given post data, Post document is generated in the database	No Post can be created	Medium	High	Medium
test_get_post_by_restaurant	timeline	PostSuite	All post documents for a restaurant within the database are correctly retrieved and sorted	Restaurant Timelines will not be populated properly	Medium	High	Medium
test_delete	timeline	PostSuite	Given post id, post and its related comments are deleted from the database	No Post can be deleted	Medium	Medium	Medium
test_get_all_post	timeline	PostSuite	All post documents within the database are correctly retrieved and sorted	Story tab will not be populated properly	Medium	High	High
test_upload_comment	timeline	CommentSuite	Given Comment data, Comment document is generated in the database	No Comments can be created	Medium	High	Medium
test_upload_post	timeline	CommentSuite	Given Comment data, Comment document id is added to original post's comments	No Comments can be viewed	Medium	High	Medium
test_comment_delete_comment	timeline	CommentSuite	Given the id of the comment, comment is deleted on the comment side	No Comments can be deleted	Low	Medium	Medium
test_comment_delete_post	timeline	CommentSuite	Given the id of the comment, comment is deleted on the post side	Posts will include deleted comments	Low	Medium	Medium
test_get_comment	timeline	CommentSuite	Correct comment document is retrieved given primary key 'id'	Comments can not be viewed	Medium	Medium	Medium
test_upload	cloud_storage	CloudStorageTestCases	File is uploaded to cloud, and correct path pointing to file is returned	Images media cannot be changed	High	High	High
test_delete	cloud_storage	CloudStorageTestCases	File is removed from the cloud	Images remain clogging the storage	Medium	Medium	Medium
test_delete_default	cloud_storage	CloudStorageTestCases	Files in default-buckets are not deleted	Default images are deleted, affecting many users unwantingly	High	High	High
test_restaurant_logo	cloud_storage	CloudEndPoints	Url is saved to correct restaurant document's logo_url	Restaurant logo cannot be updated	Medium	High	Medium
test_restaurant_cover_photo	cloud_storage	CloudEndPoints	Url is saved to correct restaurant document's cover_photo_url	Restaurant cover photo cannot be updated	Medium	High	Medium
test_restaurant_owner_photo_url	cloud_storage	CloudEndPoints	Url is saved to correct restaurant document's owner picture	Restaurant owner picture cannot be updated	Medium	High	Medium
test_food_picture	cloud_storage	CloudEndPoints	Url is saved to correct food document's picture	Food picture cannot be updated	Medium	High	Medium
test_owner_picture	cloud_storage	CloudEndPoints	Url is saved to correct SDUser document's picture	SDUser picture logo cannot be updated	Medium	High	Medium
test_insert_cart	order	CartTestCases	Test if cart document is inserted into the database	No customer can create a cart and therefore can't order food	High	High	High
test_get_users_cart_unsent	order	CartTestCases	Test if the active cart document is retrievable by user_email	No customer can view their cart and therefore can't order food	High	High	High
test_get_users_cart_sent	order	CartTestCases	Test if sent cart documents are retrievable by user_email	No customer can view theirsent carts	High	Low	Medium
test_get_restaurant_carts	order	CartTestCases	Test if sent cart documents are retrievable by restaurant_id	No restaurant can view their orders and therefore can't fill orders	High	High	High
test_get_closed_cart	order	CartTestCases	Test if a user with only closed carts is notified that there is no open cart	May cause loading closed carts and an inability to tell if a new cart should be created	High	High	High
test_get_no_cart	order	CartTestCases	Test if a user with no previous carts is notified that there is no open cart	Causes an inability to tell if a new cart should be created	High	High	High
test_insert_item_item	order	CartTestCases	Test if item document is inserted into the database	Carts will not be populated	High	High	High
test_insert_item_cart	order	CartTestCases	Test if cart price is correct after item document is inserted into the database	Total Price will not be recorded	High	High	High
test_send	order	CartStatusCases	Test if send timestamp status is updated	Orders cannot be sent	High	High	High
test_accept	order	CartStatusCases	Test if accept status is updated	Orders cannot be accepted	High	High	High
test_complete	order	CartStatusCases	Test if completion status is updated	Orders cannot be completed	High	High	High
test_decline	order	CartStatusCases	Test if declining an endpoint is completed successfully	Orders cannot be declined by the RO after they are sent	High	Medium	Medium
test_decline_fail	order	CartStatusCases	Test if declining an endpoint fails correctly	Orders may be incorrectly declined	Low	High	Medium
test_order_fail	order	CartStatusCases	Test if order status cannot be updated and an appropriate httpbadresponse error message is sent	Orders are illegally updated	High	High	High
test_invalid_form	order	CartStatusCases	Test if invalid request httpbadresponse is sent	Invalid requests may infect the database	High	High	High
test_remove_price	order	CartRemoveTestCases	Test if item has been successfully removed from the database	Illegal item resides in database	High	High	High
test_remove_item_price	order	CartRemoveTestCases	Test if cart price has been correctly decremented from the database	User may have to pay for items he has nor ordered	High	High	High
test_remove_cart	order	CartRemoveTestCases	Test if cart has been removed from database given it is last item	Illegal cart resides in database	High	High	High
test_item_count_change	order	CartEditTestCases	Test if item count has been correctly edited	Incorrectly edit item amounts	High	High	High
test_count_change_cart_price	order	CartEditTestCases	Test if item count change properly recalculates prices on its cart	User will pay incorrect prices	High	High	High
test_zerocount_deletion	order	CartEditTestCases	Test if item has been removed database given its amount is changed to 0	Illegal item resides in database	Medium	Medium	Medium



test_insert_reviews	review	ReviewCases	Create restaurant food, tag and food object for testing	Reviews can not be added to restaurants	Medium	Medium	Medium
test_get_restaurant_reviews	review	ReviewCases	Test if the correct reviews are returned	Reviews will not be displayed on the restaurant page	Medium	Medium	Medium
test_insert_review_rating	review	ReviewCases	Test the rating if the restaurant ratings is reflective of average review ratings	Restaurants won't have a rating	Medium	Medium	Medium
test_cancel_cart_cart	order	CartEditTestCases	Test if cart cancellation deletes the cart from the database	Empty Carts clutter database	Low	Medium	Medium
test_cancel_cart_item	order	CartEditTestCases	Test if cart cancellation deletes the cart's items from the database	Items without a cart clutter the database	Low	Medium	Medium
test_get_items_by_cart	order	CartEditTestCases	Test if all items associated by cart_id are correctly returned	Items cannot be views in the cart	High	High	High

5.3 Frontend Tests

5.3.1 Testing Table Legend

Column	Column Description
Test Name	Name of Test
UI Components	Page/interface where the changes affect
Steps	Reproduction steps
Cases	List of cases for the test
Problems	Problems that future implementation may affect it

5.3.2 Master Testing Table

Our nicely formatted master testing table in a Markdown format can be found in our repository in the following section: team_08-project/sd-site/docs/tests/front-tests.md However, below is an export of that table (in the format of screenshots due to the number of columns in the table:

Test Name	UI Components	Steps	Cases	Problems
RO Menu Edit	Restaurant Page, Menu edit page	Signup/login as RO (make sure they have a restaurant id), go to restaurant page, edit menu (dishes should be here, should be able to add, edit, and delete), save (restaurant menu should be updated)	Menu edit page access, add to menu, delete from menu, edit menu, save menu	Refreshing on Menu Edit page resets the role and restaurant id
RO Form Guards	Owner Setup Page, Menu Setup Page, Menu Edit Page	Do not login and attempt to access Owner Setup Page, Menu Setup Page and Menu Edit page and repeat while logged in as BU (all should be denied access). Now upgrade/login as a RO and attempt to access Owner Setup Page, Menu Setup Page and Menu Edit page (Make sure to have the needed query params - restaurant ID and role) then access should be granted and the forms can be filled out as normal	Owner setup page access, menu setup page access, menu edit page access, fill out owner story, add dishes using menu setup, add/edit/delete dishes using menu edit	Add and delete dishes are not dynamically displayed on FireFox
RO Creation Form	Restaurant Setup Page	1) Sign up for a new account and upgrade the account to RO by selecting upgrade account 2) Fill out all required restaurant sign up forms (make sure the email in the 1st form is *not* used already) 3) Sign out and sign up for another new account 4) Attempt to upgrade this account with a new restaurant that uses the same restaurant email as step 2 - *It should say this email already has a restaurant attached and*	Create new unique restaurant and create duplicate restaurant (in terms of email)	

Add and delete posts	Timeline	1) Log in as RO and access the "Timeline", you should see trash icons for all posts. 2) Click "Add New Post" and enter some content, the timeline should update with the new post. 3) Click the trash icon on any post, a confirmation should appear and if confirmed, the post should update without the post deleted.	Ability for RO to add and delete posts to their timeline	Query parameters are VERY important; also takes a while to load up everything; FireFox doing the dynamic after first one again
Add and delete comments	Timeline	1) Log in as BU, can comment on any post and see the comment section update. 2) Log in as RO and navigate to own timeline, should see delete buttons next to comments. 3) Click the trash can icon and confirm, comment should no longer be in the comment section of the post.	Ability for RO to delete comments, and any logged in user to comment	Query parameters are VERY important
Edit Restaurant Info	Restaurant Page	1. Login/Signup as a RO 2) Go to the "Restaurant Page" page 3) There should be an edit button in the top right corner (under log out button) 4) Click the edit button and you should be redirected to a form that says Edit Restaurants Information with fields that are pre-populated with their restaurant info *(note: email should be greyed out and not editable - i.e. readonly* 5) Edit fields as desired and save 6) You should now be redirected back to the restaurant page with the updated info 7) You can also click edit again and click cancel and no changes should be made	Restaurant Information Edit *- name, address, city, phone number, pricepoint, cuisine, description, and twitter/instagram URL (i.e. everything but email)*	

RO About Us Edit	Restaurant Page	1) Browse all restaurants and select a restaurant (choose one of the latest ones - like Sam's Sushi), the about us tab should be there with a picture (from Figma) and their name and a box with their "story" underneath 2) Login to a RO account or signup + upgrade to a RO account and select "Restaurant Page", the about us tab should appear like part 1 but with an extra button (not configured yet as task SD-154) to edit information	View about us page from customer and restaurant owner account	
About Us Tab	Restaurant Page	1) Login/Signup as a RO 2) Go to the "Restaurant Page" page 3) There should be an edit button that was created from before on the about us tab 4) Click the edit button and you should be redirected to a form that says *Edit Owner Information* with fields that are pre-populated with their existing story 5) Edit the name and/or story and save then you should be redirected back to the restaurant page with the owner name and/or story updated	Owner story edit - *name and story*	
Timeline View	Timeline, Restaurant Page	1) Access "Timeline" without logging in, you should see all restaurant posts and be unable to comment. 2) Access "Timeline" logged in as a BU, should see all restaurant posts and be able to comment. 3) BU account, access an individual restaurant page and click "View Updates", should see individual restaurant posts or empty if it is empty. 4) Sign in as RO and access "Timeline", should see own restaurant posts, be able to comment, and see the new post and trash icons	Ability to view and not view parts of timeline depending on role and login status. 5) Sign in as RO without posts and access "Timeline", should see the empty page and a "Start Conversation" button	Query parameters are VERY important
HTTP Call Loader	Most Pages	1) Navigate to "Browse", should see indicator 2) Naviagte to timeline, should see indicator and be allowed to interact AFTER all posts and comments are loaded. 3) These 2 pages are the main issue, feel free to check anywhere else.	Any page that does a http request will now have a loading indicator to wait for the page to load before the user can interact with it	
Dynamic Search Bar	Browse Page	1) Navigate to "Browse" 2) For restaurants, try typing in names, pricepoints, owner names, and cuisines into the search bar. The listing should update per character. 3) For dishes, try typing in name and prices. The listing should update per character	Restaurants (name, pricepoint, owner name, cuisine), Dishes (name, price)	
User Required Fields Check	Checkout Page	1) Login as a user without address and phone number. 2) Access "Cart", you should be redirected to the profile page to add your information. 3) Login as a user with address and phone number. 4) Access "Cart", you should be able to access the page.	Users with required fields, and users without required fields; access to the cart page to checkout.	Query parameters are VERY important
User Info Modal	Home Page & Profile Page	1) Sign up for a new account 2) A modal should pop up after creating the account successfully 3) Edit just the name, save then log out 4) Login again with the account just created and you should see a modal pop up with all fields blank except the name filled in from the entry from 3 5) Attempt to enter invalid phone number and/or birthday, it should show an alert warning of invalid format 6) Fill out the remaining fields and for birthday you can click on the calendar button to get an interactive calendar 7) The user and the corresponding data should be reflected in the database	Fill in nothing, fill some fields, fill all fields, invalid input	

Checkout UI (Static)	Checkout/Cart	1) Login to a BU account (needed due to guard check from previous) 2) Go to the cart page 3) You should see a UI with 3 cart items and a total (manually calculated in the .ts file) 4) Select the payment button and you'll be redirected to the payment UI 5) Feel free to edit the cart.json with empty list to view how it looks if no orders and the button that is connected to the browse page	Empty list of orders and non-empty list of orders	
Profile UI	Profile Page	1) Login as a BU 2) Go to the profiles tab, you should see the current user info stored in there 3) Feel free to shrink it to as small as possible to see mobile view 4) Click edit user info and edit the user info and click save and the page should refresh with the newly updated user info	Display user info and edit user info	
Image Upload Process (RO Side)	Restaurant page, Restaurant set up	1) Sign up as a RO and try adding the images for logo and owner. Should give no problems. 2) Add items to the menu with an image, the page should update with the menu item and the picture. 3) Save and head to restaurant page, you should see the default burger header, the correct owner image, and items on the menu with the right photos. 4) Try editing images for menu, owner, and header. 4) Log out and go to browse. Look for your created restaurant and the logo should be the image you inputted.	Upload/edit images for owner, logo, header, dishes	
All Owners Page	All Owners Page	1) Launch the website and click on the "CHEFS" tab 2) You should see a listing of all current restaurant owners, owner's bio, owner's pic and a button with a link to the actual restaurant 3) Login as a BU and the chef tab should be there 4) Login as a RO and the chef tab should not be there	View all stories with the 3 different types of role (non-logged in, BU and RO)	
User Address in Navigation Bar	All User Accesible Pages	1) Do not login and there should be no indication of address on the navigation bar *-Also there should not be any dropdown menus* 2) Login as a RO and there should be no indication of address on the navigation bar *-Also there should not be any dropdown menus* 3) Sign up as a new BU (do not fill in the info modal), the navigation bar should say "No Address Provided" with a pin 4) Now go to profile page and edit the info (i.e. add an address) now the navigation bar should display the address recently provided 5) Feel free to also sign up as a new BU and fill out the info modal after signup and you should see the navigation bar displays the user address as well	Try as non-logged in user and RO; A BU user with address and without address	
External Delivery Link	Restaurant Page, Restaurant Setup and Restauant Edit	1) Sign up for a new account and upgrade to RO 2) Fill out all needed forms and ensure to include a delivery link (find one off uber eats or something) 3) Go to my restaurant page and then there should be a shipping logo where if you click you get redirected to the uber eats page provided in *step 2* 4) Try editing the external delivery link (to door dash or what now) and the logo will now reflect the newly updated or even try removing the link and the logo should disappear	Restaurant with no external delivery link and restaurant with delivery link	
Filter boxes	Browse (Restaurant and Dishes)	1) Enter the browse page, check some things in the ratings box, should see page update. 2) Uncheck those boxes, page should reload all restaurants. 3) Repeat with pricepoint. 4) Head to dishes, repeat with price.	It's one box at a time, one search bar at a time. Basically you can't search two things at once.	
BU Ordering Process	Checkout, payments	1) Log in as BU, add some dishes to the cart (make sure its the same restaurant). 2) Head to Cart page, try changing the numbers of the items and save, should save. 3) Place order, it should take you to the payments page. 4) Press pay now, a modal should pop up talking about a 30 second window. 5) Cancel order, should be taken back to cart. 6) Do it again but confirm, cart should be sent and cart page is now empty. 7) Do it again and wait 30 seconds, same thing should happen as 6. 7) Add more dishes to the card and try clearing individual dishes and the whole cart.	Add to cart, clear cart, delete items from cart, send cart, cancel	

All Orders Dashboard	Dashboard	1) Signup/login as a RO account 2) Go to dashboard in the navigation bar 3) Click on the view all orders button 4) The orders should then be displayed one by one with a status of new, in-progress or accepted 5) Use the back button at the bottom to go back to the restaurant overall dashboard	Display new orders, in-progress and completed orders	
Add review	Restaurant Page	1) Enter a restaurant page and go to reviews, should not see an add review button. 2) Login as a BU and add a complete review to the restaurant, it should show up in the database. 3) Do an incompleted review (ie. No title or comment), should get an alert.	Complete and incomplete reviews	
Accept, Complete & Cancel Orders	Restaurant Dashboard	1) Login as a RO 2) Go to the dashboard page 3) There are 3 orders in the new orders, perform the following in this order 4) Cancel the first order - *should have a cancel order confirmation modal, order should be gone from the new orders & all orders page* 5) Mark the second order as accepted - *after reloading, the order should be in the in-progress tab* 6) Go to the in-progress tab and mark that order as complete - *after reloading, the order should be in the complete tab* 7) During steps 5 & 6, feel free to check the all orders page	Accept an order, complete an order and cancel an orer	
Menu categories	Restaurtant page	1) Make a new restaurant, when adding dishes make sure to add category. 2) Head to restaurant page and check if tabs were created and dishes were sorted based on your added categories. 3) Edit a dish and change its category, page should update accordingly. 4) Remove all dishes from a category (either by deleting or changing), tab should no longer be there. 4) Add another dish with a new category, new tab should appear.	Correct category tabs, remove tab, add tab	
Load Restaurant Reviews	Restaurant Page	1) Do not login and go to a restaurant with reviews (Ex: Pho Metro) 2) Go to the reviews tab and you should see one review pop up 3) Login as a BU and go to any restaurant and add a review 4) The page should refresh and now the review should appear for that restaurant	View reviews, add reviews + view reviews	
View all Orders	View all orders UI and checkout UI	1) Go to an account with orders 2) Go to profile > view all orders 3) There should be orders 4) Go to checkout page 5) The view all orders should be gone	View all orders	

6 - Validation

6.1 Meeting Minutes

All members were present when meeting with the client. The main topics discussed during the meetings with the clients were the colour theme of the webpage, the overall presentation of the webpage and designs for future components of the webpage. In terms of colour theme, we wanted to know if the new updated colours reflect well with the new City of Toronto approved logos. Then for presentation of the webpage we were asking if the webpage achieves their expectation of putting less emphasis on the eCommerce aspect of the webpage and more emphasis on highlight Scarborough owners.Then we moved on to showing Figma designs (an export can be found in the Deliverable 5 folder) to ensure that the user interfaces that we are going to be implementing in the feature are what they expected.

6.2 Demos Produced

During our client demos, we went through a walkthrough of the currently deployed web page to show them a sense of how the live webpage currently looks like. We walked through our new webpage colour themes, newly added pages that include all owners' pages. We also walked through our newly updated browse page which now includes a map of user location and the ability to search and filter. Afterwards, we showed our Figma designs to the clients regarding the design of new user interfaces that are to be implemented in our future sprints. Figma designs are attached in a file called “Team Aqua - Scarborough Dining (Customer).pdf” and “Team Aqua - Scarborough Dining (Restaurant Owner).pdf”.

6.3 Questions Prepared for Clients

Below is a list of questions that we prepared for clients to ask about regarding our overall design and plans:

- We have implemented a page that is very similar to the 150 Neighbours page, is this a good reflection of what was expected for the celebrating owners and food aspect of the webpage?
- Does the newly updated colour theme reflect well of the City of Toronto colours and the approved logo?
- Does the list of functionality left to be implemented complete all the requirements initially given out in order to reach minimal value product?

6.4 Feedback From Clients

Below were several suggestions and improvements that the clients have offered us after our demos:

- Change the theme and logo to match Scarborough Dining’s theme and logo
- Add Tooltips to buttons to further explain the function of each one
- Implement Age Validation (18+)
- Create an Admin Page to manage all aspects of the website
- Replace Dish Pictures with Non-Fast Foods and use Real People instead of a placeholder
- The webpage captures the essence of putting emphasis on Scarborough owners as desired

7 - Retrospection

7.1 Overview of the Project Saga

Over the course of the past 2 months, as a team we became more seasoned software engineers as we implemented the Scarborough Dining webpage.. This project taught us valuable things in software development such as version control, usage of an industry technology stack, code

review process, industry level testing and documentation. In addition, we have developed our skills regarding requirement illications from clients and transforming them into personas and user stories. Lastly, this project has allowed for all team members to improve their communication skills either through presenting during deliverable meetings or when interacting with clients during client office hours.

7.2 What was your estimated project velocity? How did it change with each deliverable?

Sprint 5 started with 45 story points but ended with 40 points completed. The 5 points worth of an incomplete task was creating an endpoint to allow image upload, which proved difficult due to networking issues.

Sprint 6 started with 75 story points and ended with all tasks completed. This reason for the increased story points for this sprint was due to the fact several team members had higher availabilities as they had a lighter course load compared to other team members. Thus, with this increased availability they were able to concentrate more of their time working on this project and as a result implementing more needed functionality.

Sprint 7 started with 90 story points and ended with all tasks completed. Sprint 7 with 90 points was the highest number of story points that we assigned for one sprint but this was due to a combination of increased availability of various team members and the fact this would be our last sprint so we need to implement all required functionality by the end of this sprint.

In comparison to the previous deliverables, more story points and tasks were completed for the sprints between deliverables 4 and deliverables 5. This is mainly due to the inability to push off tasks to a later sprint since it is near the end of the project, as well as bug fixes that had to be fixed along the way in order to have the client side function properly.

7.3 Did you follow your plan(s) exactly, or did you have to re-plan? Why and when?

We did not completely follow our exact plans for each sprint due to bug fixes made along the way. Typically the backend endpoints would be created a sprint before it gets implemented in the front so that the front does not get blocked for their tasks. Sometimes the endpoint may not return what exactly the frontend needs when those tasks are being completed. Such issues means that the endpoints need to be fixed right away so that the front can complete their task by the end of the sprint. In addition, some sprints we had left over tasks that could not be finished within that sprint due to the complexity of the task. This led to replanning as we had to include that task in the next sprint backlog and this may result in having less new tasks done in the next sprint.

7.4 How does the work done for deliverable 5 compare to the work on all previous deliverables, both in terms of progress and end result?

A lot more work was completed in deliverable 5 in comparison to all the other deliverables. For the last two sprints, we completed 75 and 90 points respectively, which is almost twice the usual amounts that we completed during the last deliverables. There were more changes based on client needs due to belated answers and requirements. However, comparing the end result of the work done in deliverable 5 to previous deliverable we managed to implement a vast majority of the client requirements. For instance, we had image upload, restaurant setup and edit information and an all owners page done within deliverable 5. This resulted from the fact team members were now more seasoned with using the various technologies needed whether it be the frontend or backend. Lastly, since we had the framework setup during the earlier deliverables we were able to easily swap out static data (in the form of a JSON file) with backend endpoints and having the functionality be fully implemented.

7.5 What would you have done differently if you had to restart the project?

One thing we've realized when doing the project is the lack of manpower on the frontend side. We had two main members working in the frontend (Winnie and Tony) and three working in the backend (Alac, Calvin, and Robert), with minimal exposure to frontend related concepts. Nearing the end of the project, a lot of tasks piled into the front to be completed, which put a lot of stress on the two members that have mainly been working in the front. In the last sprint, Robert helped out by taking on some tasks that involve working with the frontend code.

Another thing we would have done differently is work slower and focus on implementing the features that we are creating with better design principles such as those introduced in class. An example would be to use the adapter pattern to hide the implementation of various APIs which we are using.We would also like to perform refactors during each performance review to reduce the need to do this refactoring work in one large block at the end of the project.

8 - Maintainability

8.1 Backend

Here is a list of things that were performed to the backend aspect of this project to ensure its maintainable and open for extension in the future:

- Broken into apps that represent the different major aspects of the website
- Each app contain its own urls(routing abstraction), views (request handler abstraction), models(database abstraction), tests, and enum files that all relate to a common aspect of the website
- Enum and model structure make it much easier to refactor or extend any code

- Interface for media uploads which allows for quick implementation of media uploads without rewriting the cloud media storage API functionality
- Refactored timeline, restaurant and order app following refactoring techniques shown in class (using helper functions, removing magic numbers, descriptive variables/functions for code clarity)
- Utility module for reusing common function (serializing of django models across multiple apps)
- Abstraction for randomly generating JSON objects for database seeding which can allow for quickly setting up a test database for manual testing or demonstrations

8.2 Frontend

Here is a list of things that were performed to the frontend aspect of this project to ensure its maintainable and open for extension in the future:

- Documentation of each component and page of usability
- Documentation on how to use specific features (for example, image upload and session storage)
- Global styles sheet to easily change themes and styles of multiple components at a time
- Reusable components that can be easily changed in style and called in any location
- Abstract class for form validation which allows to quickly write validation schemes for any user form
- Utility classes for functions that are likely to be reused
- Testing files are generated for each frontend component and service (they were not utilized, but the files are there if tests are created in the future to run)

8.3 Deploy and Development strategy

The deployment strategy was decided at the very beginning of the project with the goal of having a *reproducible build* which we discussed early on in class. There is a software known as **Docker**, this is a containerization software through which you can determine several features of the server's environment such as *Operating System, Installed packages and versions, Networking settings, Build strategy* through a script which will reproduce *exactly* the same steps and environment on our local and deployed machines.

This was incredibly convenient during our development process, because it allowed our frontend developers to run the backend server just by pulling from git and building/running the docker container as opposed to needing to set up all of the python packages. This was also the same for our frontend container, being convenient for our backend developers.

Since our developers were able to run the containers locally, it also made the deployment process ***almost*** completely error-free since the container they were developing on used the same settings as the deployed container. I say "almost" because the development containers

were not always set up to be exactly the same, but this was a human error of not understanding docker as opposed to an error in code or environment setup.

9 - Existing Issues

Here is a list of the existing (known) bugs/ missing features which we didn't have the time to fix/ implement respectively:

Bugs:

- Restaurants with duplicate addresses overwrite each-other's pins on the map
- After logging in and refreshing the page, the navbar elements(browse, etc.) disappear and you are forced to hit log out before you can login or interact with the website. This issue is caused by chrome blocking auth0's cookies because of a "SameSite" setting on the cookies(some security setting), we do not have time to look into a fix.
- Cart items are not updated when a food is deleted/modified from the restaurant's menu, so cart totals will be incorrect and carts will not be able to send correctly
- Endpoints for the ordering and timeline systems are in 1n normal form, as opposed to one endpoint for each page to get all of the required data we have several because of the two tables

Missing Features:(NON-MVP)

- SMS Verification
- Search / Filters are separate, we were not able to combine them.
- Payment form
- Live Orders (you have to refresh or interact with the page from the owner's side to see new orders)
- Favourites Page Backend functionality is missing, everything else is ready for it
- Cuisines at the bottom of the homepage
- System analytics
- Search History/ Page Visit History data
- Search recommendations with tagging