INFO30005 Web Information Technologies
Deliverable 4 – Report
Google 2.0
Tutorial: Tuesday 11am , Steven , Group 3
Damian Wang (995317), Natasha Ireland (999132), Leonard Wang (993548), Erica Caelli (949209), Kumi Sotomatsu (940966)

Introduction

In this project, our team was tasked with designing and developing two web apps for Snacks in a Van, one for vendors and one for customers. In the vendor app, vendors are able to mark their current location on the map, mark themselves as open or closed for business as well as view the current and previous orders. Meanwhile, in the customer app, customers are able to view the vans that are open for business that are located closest to them, view the menu and submit an order. Once an order is submitted customers can modify or cancel their order within the allocated 10 minute time period.

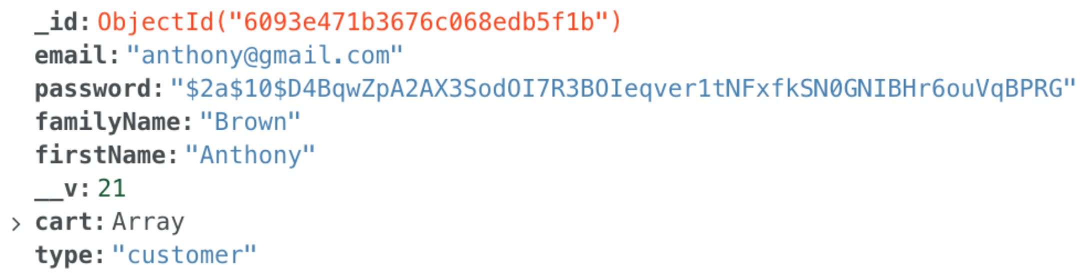
Development Process

For our website, we have used the Node JS Express framework to create the backend for both the server-side and client-side. The customer and vendor apps were designed with several key features in mind: authentication, location services (for locating vendors and setting vendor location), session management, accessibility and responsiveness.

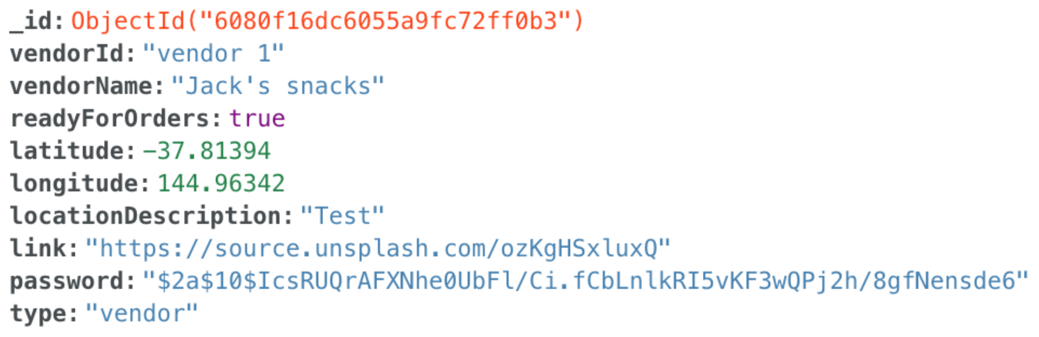
Authentication system

For the authentication system, we have used the passport.js middleware to implement user authentication for our app on both vendor and customer side. This enables our app to have a token-based authentication system which provides users authorization only while in the current session. Each time a user registers and verifies themselves on the website, a token is created which enables different and additional URL access to navigate the site. The token is generated using a module called JsonWebToken which is implemented in passport.js. This authentication feature will protect our app against attackers who are attempting to access the account of another user or vendor through brute force attacks.

The app authenticates users based on their username and password. Upon signup, the user's password will be hashed via the module BCrypt before being stored in the database. Each time an existing user enters a password to log in, the password that they entered is hashed and this hashed value is compared with the hashed value stored in the database, and if they are the same, the user will be successful in logging in. This feature allows us to protect against attackers trying to hack into our database and also prevents the passwords of the users of our app from being exposed to the employees working at Snacks in a van who are maintaining the database. Following is an example of how passwords for both vendors and customers are stored in our database:



```
  _id: ObjectId("6093e471b3676c068edb5f1b")
  email: "anthony@gmail.com"
  password: "$2a$10$D4BqwZpA2AX3SodOI7R3BOIeqver1tNFxfkSN0GNIBHr6ouVqBPRG"
  familyName: "Brown"
  firstName: "Anthony"
  __v: 21
> cart: Array
  type: "customer"
```



```
_id: ObjectId("6080f16dc6055a9fc72ff0b3")
vendorId: "vendor 1"
vendorName: "Jack's snacks"
readyForOrders: true
latitude: -37.81394
longitude: 144.96342
locationDescription: "Test"
link: "https://source.unsplash.com/ozKgHSxluxQ"
password: "$2a$10$IcsRUQrAFXNhe0UbFl/Ci.fCbLnlkRI5vKF3wQPj2h/8gfNensde6"
type: "vendor"
```

The app also uses the password validator package to ensure that the passwords chosen by our customers follow secure password protocols which include: having at least one alphabetic character, at least one numeric digit and being at least 8 characters in length. This will ensure that the passwords entered by our customers are more secure and less easily guessed through brute force attacks.

Sessions

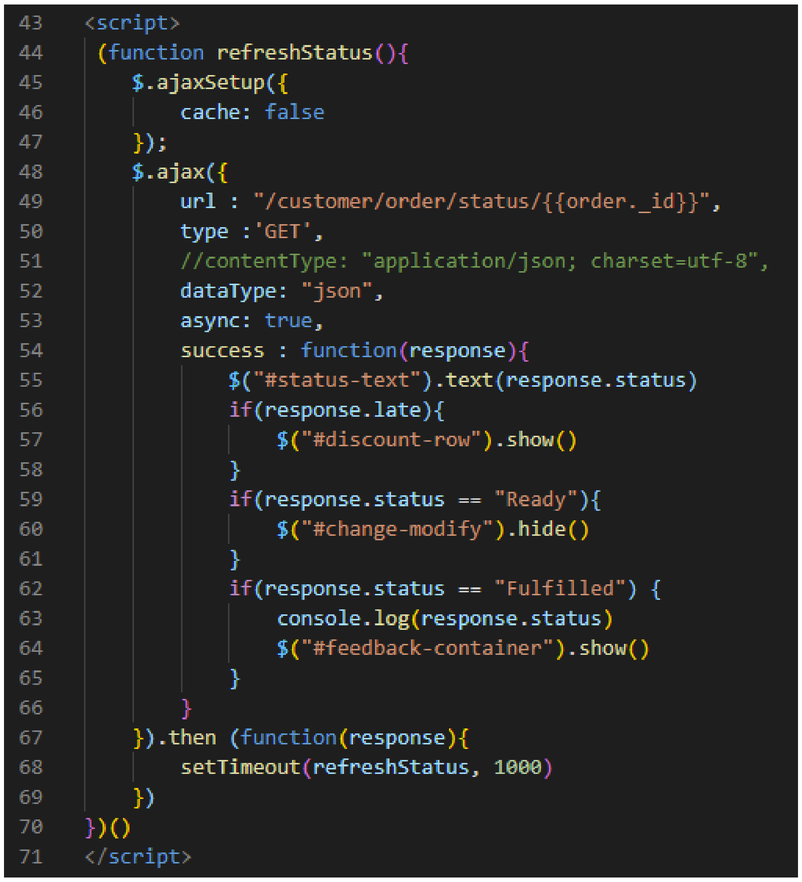
We have used the express-sessions package to enable sessions with our express apps to store the cart information for our customers. In addition, we have stored the user's ID to enable authenticated users to continue accessing restricted pages in the app.

Location map

To obtain the user's location, we have used the geolocation API. The map displaying the user's location and the vendors nearby was created using leaflet API, as it also supports usage on mobile devices and is suitable for our app which is used on both mobile, desktop and tablet devices. Location services must be enabled to make use of this feature.

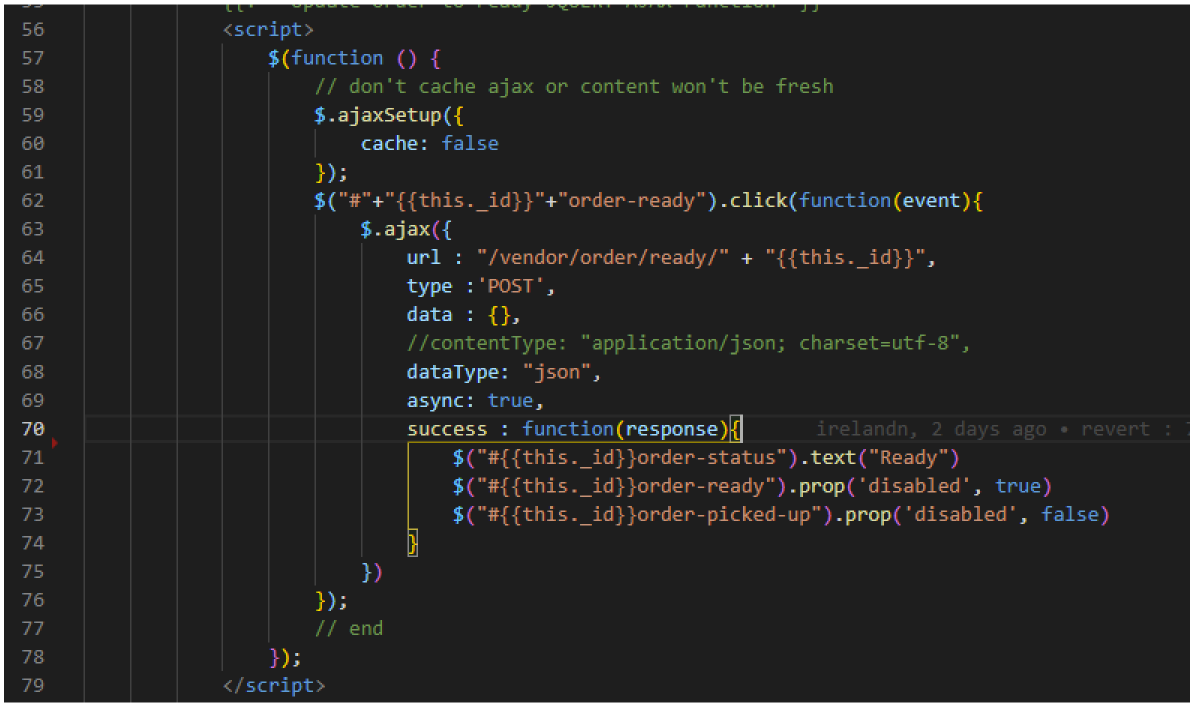
jQuery and AJAX

On the front-end, jQuery is used to allow changes to happen dynamically to the web page after loading without the need to refresh. This means pages are more usable as users do not need to refresh to see new or changed content. An example of this is changing the order status of a current order being viewed by the customer. Below is a snippet from *oneOrder.hbs* that queries the database for the status of an order every 1000 milliseconds to allow the customer to see live updates from the server. Line 55, updates the status on the page before hiding and slowing relevant elements depending on the current status.



```
43  <script>
44   (function refreshStatus(){
45      $.ajaxSetup({
46          cache: false
47      });
48      $.ajax({
49          url : "/customer/order/status/{{order._id}}",
50          type :'GET',
51          //contentType: "application/json; charset=utf-8",
52          dataType: "json",
53          async: true,
54          success : function(response){
55              $("#status-text").text(response.status)
56              if(response.late){
57                  $("#discount-row").show()
58              }
59              if(response.status == "Ready"){
60                  $("#change-modify").hide()
61              }
62              if(response.status == "Fulfilled") {
63                  console.log(response.status)
64                  $("#feedback-container").show()
65              }
66          }
67      }).then (function(response){
68          setTimeout(refreshStatus, 1000)
69      })
70  })()
71  </script>
```

jQuery is also used to post when buttons are pressed to update dynamically as shown below in a snippet from *vendorOneOrder.hbs*. When a ready button is pressed for an order, then a post request is sent to the server to update that order to ready before changing the status and disabling the ready button and enabling the picked up button.



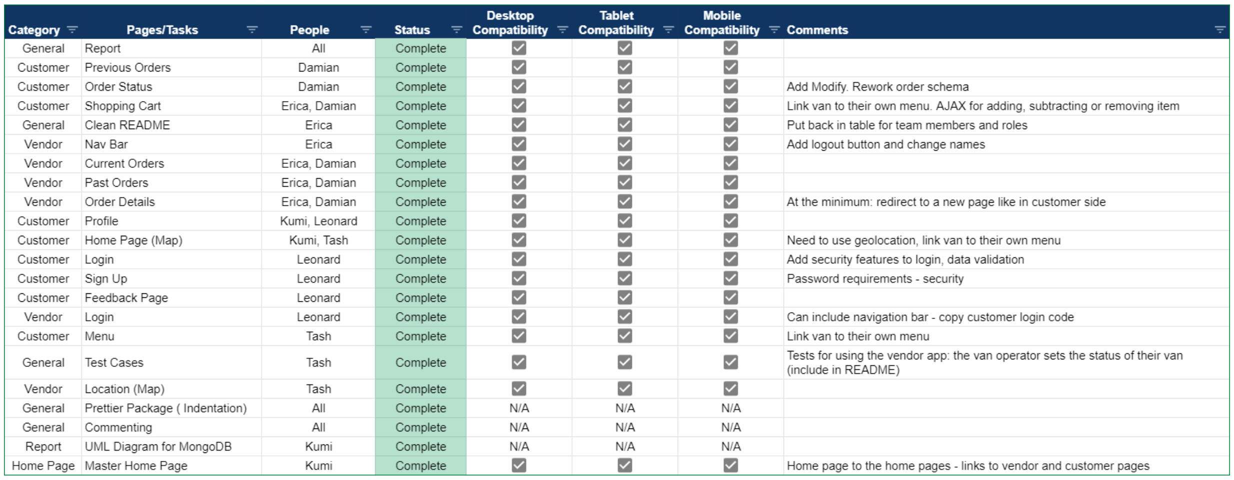
```
<script>
    $(function () {
        // don't cache ajax or content won't be fresh
        $.ajaxSetup({
            cache: false
        });
        $("#"+"{{this._id}}"+"order-ready").click(function(event){
            $.ajax({
                url : "/vendor/order/ready/" + "{{this._id}}",
                type :'POST',
                data : {},
                //contentType: "application/json; charset=utf-8",
                dataType: "json",
                async: true,
                success : function(response){        irelandn, 2 days ago • revert :
                    $("#{{this._id}}order-status").text("Ready")
                    $("#{{this._id}}order-ready").prop('disabled', true)
                    $("#{{this._id}}order-picked-up").prop('disabled', false)
                }
            })
        });
        // end
    });
</script>
```

jQuery and AJAX was used instead of handlebars logic operators as hbs executes when the html is sent to the client. Thus, it would not be possible to use handlebars to dynamically change the webpage without reloading.

Team roles

Our repository was created and handled by one team member. Team members created a new branch each time a new functionality was to be added, and merged it into the main branch.

Our group held weekly group meetings discussing our progress and working out issues together. Meetings were held virtually on discord and the meeting agenda was recorded on Google Docs. An example of how we divided the tasks is shown below:



Category	Pages/Tasks	People	Status	Desktop Compatibility	Tablet Compatibility	Mobile Compatibility	Comments
General	Report	All	Complete	✓	✓	✓	
Customer	Previous Orders	Damian	Complete	✓	✓	✓	
Customer	Order Status	Damian	Complete	✓	✓	✓	Add Modify. Rework order schema
Customer	Shopping Cart	Erica, Damian	Complete	✓	✓	✓	Link van to their own menu. AJAX for adding, subtracting or removing item
General	Clean README	Erica	Complete	✓	✓	✓	Put back in table for team members and roles
Vendor	Nav Bar	Erica	Complete	✓	✓	✓	Add logout button and change names
Vendor	Current Orders	Erica, Damian	Complete	✓	✓	✓	
Vendor	Past Orders	Erica, Damian	Complete	✓	✓	✓	
Vendor	Order Details	Erica, Damian	Complete	✓	✓	✓	At the minimum: redirect to a new page like in customer side
Customer	Profile	Kumi, Leonard	Complete	✓	✓	✓	
Customer	Home Page (Map)	Kumi, Tash	Complete	✓	✓	✓	Need to use geolocation, link van to their own menu
Customer	Login	Leonard	Complete	✓	✓	✓	Add security features to login, data validation
Customer	Sign Up	Leonard	Complete	✓	✓	✓	Password requirements - security
Customer	Feedback Page	Leonard	Complete	✓	✓	✓	
Vendor	Login	Leonard	Complete	✓	✓	✓	Can include navigation bar - copy customer login code
Customer	Menu	Tash	Complete	✓	✓	✓	Link van to their own menu
General	Test Cases	Tash	Complete	✓	✓	✓	Tests for using the vendor app: the van operator sets the status of their van (include in README)
Vendor	Location (Map)	Tash	Complete	✓	✓	✓	
General	Prettier Package (Indentation)	All	Complete	N/A	N/A	N/A	
General	Commenting	All	Complete	N/A	N/A	N/A	
Report	UML Diagram for MongoDB	Kumi	Complete	N/A	N/A	N/A	
Home Page	Master Home Page	Kumi	Complete	✓	✓	✓	Home page to the home pages - links to vendor and customer pages

Google Spreadsheet was utilised in addition to Google Docs to maintain a running checklist of pages/tasks that were needed to be completed throughout the duration of the project. The table provided information on each member's roles, the status of each task, as well as any additional comments.

Team Member	**Key Contributions**
Leonard Wang	Authentication API services Styling and functionality of login, sign up, profile and feedback page
Damian Wang	Created back-end Order and Cart functionality Assisted in implementation of jQuery for dynamic pages Troubleshooting technical issues
Natasha Ireland	Menu, menu item, vendor map, customer map, one order: route, control and view. Map API usage. Testing
Erica Caelli	Styling: CSS, HTML Responsive and accessible design
Kumi Sotomatsu	Status Controller Entry Page Report

Each team member was responsible for creating the user interface mockup.

System architecture

The entry point to our app enables the user to navigate to either the vendor or customer web application. Once navigated: the user cannot navigate to the opposing side without returning to this entry point.

Customer

If the user is a customer, they will be redirected to a page with a map where they can find the nearest vans located to them depending on their current location. Choosing a vendor will direct them to the menu page of that specific vendor which is stored in the menu database. Customers are able to view the menu without logging in but will be required to log in or create a new account when adding items to their cart. After the customer has entered the menu items into the cart, this information is stored into the database. This will be visible to both the customer, who would be able to modify or cancel their order within 10 minutes of placing it, and to the vendors who will receive this information as an incoming order. Once an order has been fulfilled by the vendor, the customer will also be given the opportunity to provide feedback.

<u>Vendor</u>

While logged in, vendors would be able to change between tabs to view their current outstanding orders, past orders and also change their location if they decide to relocate their van. Information about the current orders and previous orders will be stored in the orders database.

If the user is a vendor, they will be first directed to the login page where they will be authenticated based on their vendor name and password. Having done so, the vendor is then directed to the setLocation page where they can check their current location on the map, set their location description-which. This will update the status of the vendor on the database (the geo-location of the vendor, the short text address of the vendor and whether the vendor is open or closed for business) which is also accessed by the customer app when displaying the vendoors closest to them. Additionally, the vendor can change the status of their business from open to closed. Opening for business is built into setting their location.

Vendors are then able to navigate between current orders, past orders, location status page and logout-which redirects them back to the login page. Current orders can be updated once completed and display a countdown so vendor's know how long it has been pending and when the over time discount will be applied. The orders are accessed and updated in the database and Jquery enables these updates to occur in real time with no page refreshes. There are also back end functions that keep orders recent and refresh so that new orders are constantly shown.

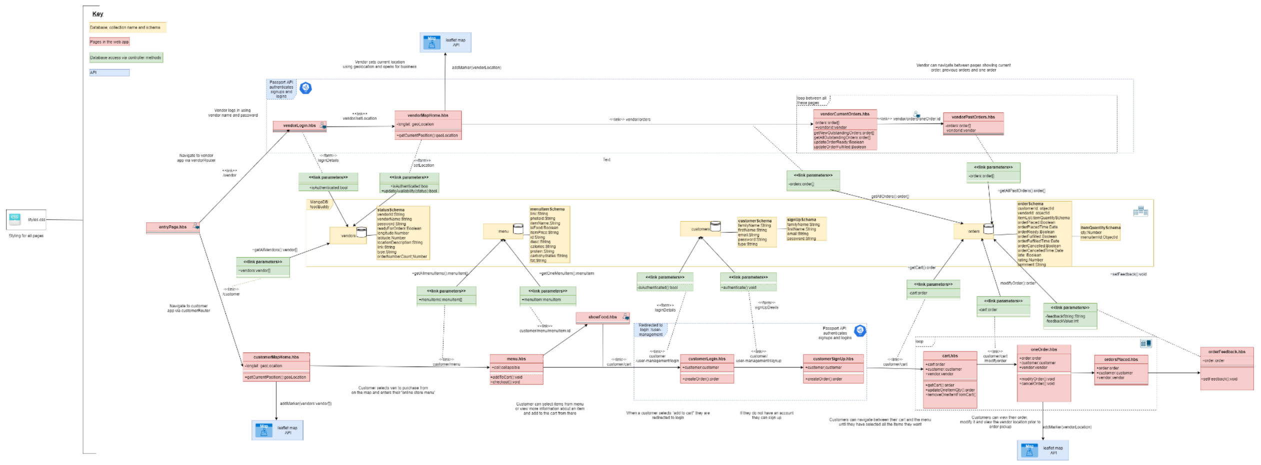


Figure 1: UML diagram illustrating the database design

Links

A link to view the published diagram:
https://viewer.diagrams.net/?highlight=0000ff&edit=_blank&layers=1&nav=1#G1wAg4wyOcVqtckJaRiNnooKz5qSJtmGI3

Link to our repository: https://github.com/INFO30005-2021-SM1/project-t08-google2-0

Snacks in a van website: https://snacksinavan-google2.herokuapp.com/