MAJOR PROJECT REPORT – COM559 – VEGGIE QUEST

Kelly, Liam

INTERACTIVE MULTIMEDIA DESIGN B00716443

Acknowledgements

I would to take this opportunity to thank everyone who supported me throughout my university education. I want to thank the Interactive Multimedia Design course team, the School of Computing clerical staff, and the staff in the CEBE Faculty Operations Office, where I took my industrial placement 2018-2019.

In particular, I would like to thank my COM559 Major Project mentor, Dr Peter Nicholl, whose advice, help and guidance has been invaluable to me over the last few months.

I would also like to thank family and friends, in particular, my brother Marc and my girlfriend Jasmine.

Table of Contents

Introduction

Veggie Quest is an online web application catered towards Vegetarians and Vegans that allows users to search for a particular menu item (e.g. cheeseburger, hot dog, vegan pizza, chips, etc.) and then see a returned list of menu items from local restaurants. The web application will also feature a login feature that allows restaurant owners (or a representative) to access the site and add, edit or remove menu items from their restaurant's menu; a rating system for menu items; a 'tagging system' for menu items; a filter function (that allows search results to be filtered by if a menu item is vegetarian, vegan, gluten-free, lactose-free, nut-free, etc.) and a responsive mobile version as well as desktop.

Aim

My aim is to create a web application that allows users to search for and view menu items from local restaurants. The restaurants themselves will be able to log in and add, edit or remove menu items from their restaurant's menu.

Objectives

In order to create the web application that I have described in the Introduction and Aim sections, there's a number of objectives I must achieve. These include:

- Creating a search function that:
    - returns results based on the title of the menu item (e.g. 'cheeseburger')
    - returns results based on alternative spellings or names (e.g. 'cheeze' is a common 'Veganized' spelling of 'cheese'. A search for 'cheeseburger' must also return results for 'cheezeburger'). I think I can achieve this by allowing restaurants to enter 'tags' for each item. The search should also return based on tags that the restaurant sets (e.g. a curry has jackfruit in it, the restaurant owner should be able to attach a tag of 'jackfruit' to that item so if a user searches 'jackfruit', that curry is returned).
    - Allows users to filter menu items based on if the item is vegetarian, vegan, lactose-free, nut-free, etc.
- Creating a backend restaurant login feature and dashboard that:
    - allows restaurant owners to log in to view and edit their menu. This means being able to add, edit or remove menu items.
    - A 'menu item creation' page that allows restaurant owners to add menu items and requires them to fill out a set list of criteria. The criteria will include fields such as title, description, tags, picture (allows them to upload it), dietary information (can indicate if a menu item is vegetarian, vegan, dairy-free, etc.) and price, etc.
- Create a function that allows new restaurants to sign up to Veggie Quest. This wouldn't be automatic and would require some sort of admin verification.
- Creating a rating system for menu items. This feature would be easy to implement; all I would do is have a function that lets users rate an item out of 5, continually add this to a data value ('totalRatingValue'), then divide the it by the amount of times a rating has been given ('numOfRatings'). I could limit each IP to one 'review' per item as a way to stop potential abuse of this feature.

- I will need to get a few local restaurants on board with the concept. To do this I will get in contact with local restaurant owners, explain my concept idea, and ask them if they would like to be involved. This will also allow me to discover if there's any issues that I haven't considered.
- Create a fully functional web application on desktop as well has having a fully responsive version for mobile screens. This is essential because many of the users will be using the application on their mobile phones.
- I know I could create the web application using JavaScript/jQuery, PHP and MySQL. Based on what we are currently learning in other modules, I could also use JavaScript/jQuery, AXAJ and JSON. I also have had the PHP framework, Laravel, recommended to me. I need to further think about what I will use, but I'm confident in my abilities to use any of the aforementioned technologies, languages or frameworks.
  I also need to look at the JustEat API, which is publicly available and may be relevant to my project.

Scope

Time:

- 1st of May 2020; this gives me 5 months to get The Vegan Food Finder designed and developed to completion. I think this is ample time to fully develop the web application.
- There are a couple of hard and soft deadlines that give a better idea of some milestones I'll need to reach and when. The initial UX design needs to be completed for the 8th of November 2019, and I'll need to have a functional prototype created for the 6th of January 2020. These deadlines have helped me create the time plan featured later in the report.

Cost:

- Currently, I can't identify any costs I will incur as part of the project.

1. All of the technologies, languages and frameworks I identified in the Objectives section of this report are free to use, and the Laravel PHP framework is free and also open source.
2. The web application will be hosted on the university's servers, so there won't be any fees related to the domain name or hosting.

Quality:

- In terms of the quality of the web application that I can deliver, I'm confident that I'll be able to develop an application that will be of a high quality and have eager users on both the customer and restaurant side.
- I want to my project to be created completely with open-source software and be as up to date as possible. To be open source I can use MySQL.

This project will not include a feature to allow customers to preorder/order food from the restaurant. Although this is a feature that could be looked at down the line, it won't be in the final product that's handed in in May.

Work Completed

In its final state, Veggie Quest offers the following features on the 'user' side:

- The ability to create an account.
- The ability to log into said account, as well as the ability to change their password.
- The ability to add foods to a favourite food list, and then view that list on their user profile page
- The ability to log out

The 'restaurant' side of Veggie Quest has the following features:

- The ability to create an account, allowing them to upload a picture of their restaurant, specify what cuisine it serves, and specify if they are a fully vegan or vegetarian kitchen, or if they serve meat as well
- The ability to log into said account, the ability to change some details about their restaurant, and to update their restaurant picture
- The ability to review their menu, and the choice of:
    - Adding new foods to their menu
    - Edit existing foods
    - Deleting existing foods
- When adding and editing foods, restaurants are able to specify if the food is vegan, gluten free, or nut free, as well as being able to include a photo of the food
- The ability to log out

Veggie Quest also:

- Contains a homepage that displays all restaurants on it, accompanied by information about them. It also plots the restaurants on a map, so that users can view where the restaurants are in relation to themselves
- Displays all foods on the home page (on a button click). The intention was for users to be able to search and filter through the foods
- Has a 'food page' for each specific food, displaying information about that food, that allows users to add the food to their favourites, as well as being able to 'share' the food on Facebook (although this does not work on local host)
- Has a 'restaurant page' for each restaurant, displaying information about that restaurant, and includes a menu that you can scroll through, that lets users click on through to a food page

On the home page, all restaurants are displayed in a neat grid, and are also displayed on a map of Belfast, depending on their coordinates. The coordinates are calculated for each restaurant by using the Node-Geocode module and the OpenStreetMap API. The home page also has a button to only view foods instead.

The front end of Veggie Quest has been created using HTML5, CSS3, the Bootstrap CSS framework, JavaScript and jQuery. I also made use of the Leaflet JavaScript library for the maps. The backend has been created using Node.js (using AJAX), a number of different Node.js modules, and APIs.
All of the data is stored within a local MySQL database, set up using MAMP and administered using phpMyAdmin.

Overview of Report

This report will cover the concept definition and testing of the project, as well as going into detail on the methodology technique used for the project development. I will discuss the design stage of the Veggie Quest website, including the designing of the database and site map. I will also talk about the technology that has been implemented in the Veggie Quest website.

Concept Definition & Testing

Idea Generation

My idea for the Veggie Quest came from the difficulty in finding restaurants that cater to vegans and vegetarian. I myself am a vegetarian, so have first-hand experience in this. A lot of restaurants (especially outside of major cities), won't have their menus online, and when they do, it can be hard to find and separated from the rest of their menu, on other social media platforms, etc.

This makes it really hard for figuring out where to eat, especially when you're already out, and need to find somewhere quick.

One solution is checking if the restaurant is on online delivery services such as JustEat or Deliveroo, but this process is inconvenient because it requires the user to input a postcode, etc. before the can view the menu.

My solution to this became the idea for the Vegan Quest; having an easy to use website that allows users to search for vegan and vegetarian food in the area around them. It also allows them to search by menu item (e.g. pizza, burger), which is a feature that JustEat doesn't have at all and is only something that Deliveroo implemented recently (it is still rudimentary).

I want the user to be able to search for a menu item (e.g. pizza, pasta) and view restaurants that have that item, as well as being able to search through an individual restaurant's menu.

Contextual Research

One of the fastest growing trends at the minute is the vegan and vegetarian diet:

- By 2025 it's estimated that vegetarians and vegans will make up 25% of the British population, and that flexitarians will make up just under 50%. (Sainsbury's, 2019)
- Orders of vegan meals grew by 338% from 2018 to 2018 and 'vegan' is now the fastest growing takeaway choice. (Booth, R., 2019)

As I stated above, there's nothing really like the Vegan Food Finder at the minute. There are solutions to the problem, yes, but they're cumbersome and don't really work that well.

There is a service called HappyCow (mobile app and web page), that is a place to discover vegan and vegetarian restaurants, but it usually doesn't feature a restaurant's menu, and if it does it isn't searchable at all.

HappyCow is also limited in that it allows users to suggest websites to add (the restaurant is not involved). On the contrary, I want the restaurant to be involved, so that they can add their own menus and keep them up to date. I think this is important because it gets buy in from the restaurant and makes sure the data is well maintained and kept updated.

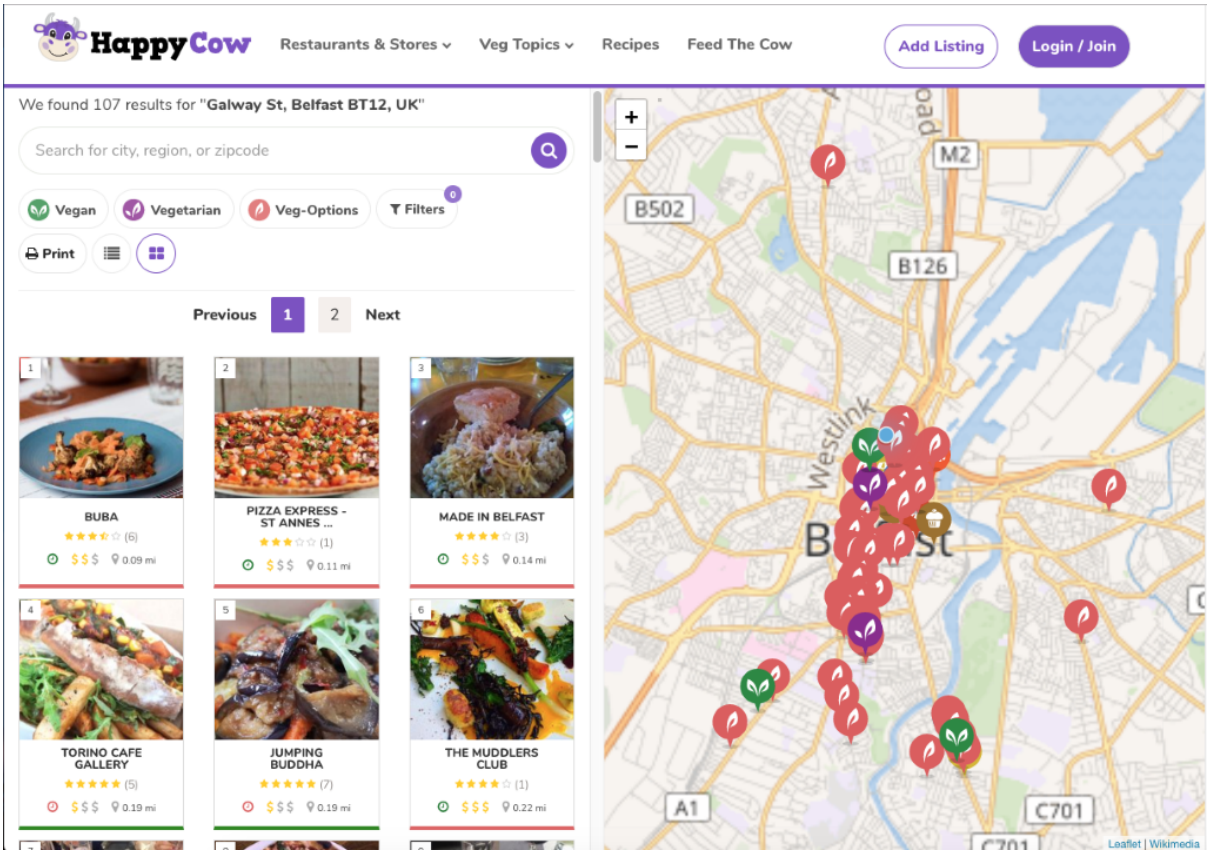


Figure 1 The Happy Cow desktop homepage.

Requirement Specifications

It was also important that I fully outlined the requirements on the project before setting into the design and creation of it. I used the 'User Stories' technique for assessing the project requirements. A user story is basically listing all of the process that a user will need to carry out for the website to have full functionality.

My user stories can be found at the bottom of this report at **APPENDIX A**.

Specification (6-UPs & 1-UPs)

After creating my user stories, I had a clearer idea of how the website would need to look, which led on to the design process. Because I envision that a lot of the website's users will be accessing the site from a mobile device, I decided to use the 'mobile first' design principle, which basically means that the site is designed for mobile users in mind. In 2016, the number of users accessing the internet on a mobile device surpassed those who access from a desktop computer, meaning a mobile first design is essential now. (Xia, V., 2017)

For designing the site, I created '6-UPs'. This is where I take each page, and create 6 different layout designs, pick the one I think is best suited, and then flesh it out more in 1 '1-UP'. I created 3 6-UPs in total:

1. Home Page
2. Food Page
3. Restaurant Dashboard Page (for adding menu items, etc.

Below are my 6-UPs and 1-UPs, highlighting the design process:

Home Page

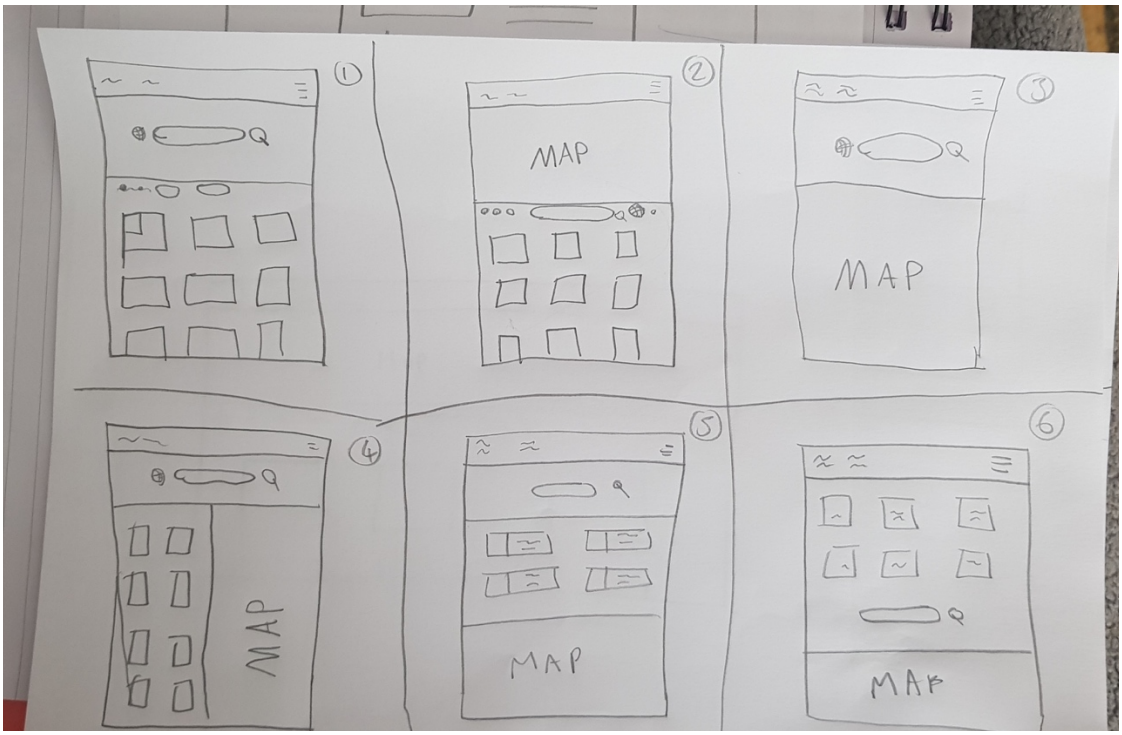


Figure 2 6-UP for Veggie Quest home page.

Home Page 1

I liked this 6-UP but one thing I thought it was missing a map feature, especially as having a visual geographic component of the site was important.

Home Page 2

Of all the 6-ups, this was the one that I liked the most. It included a map as well as displaying menu items in a tile grid display. I thought the layout would be the best suited for mobile first users and worked well with the search and function filter. This is the 6-UP I decided on.

Home Page 3

I don't really like this 6-up, although it would display results on the map, not having a grid display of all the menu items isn't the best.

Home Page 4

This idea of a 6-up came from the popular website – Airbnb's search result display. I think it could be good but ultimately because I want to display a high number of search results it isn't the best.

Home Page 5

I thought the 6-up had too many tiles displayed on the screen at one time and wouldn't lend itself to a mobile-first design.

Home Page 6

I wondered what it would look like having the search bar in the middle of the screen but ultimately I decided it wouldn't be a good layout for the website.

Home Page 1-UP

This is my 1-UP for the home page:

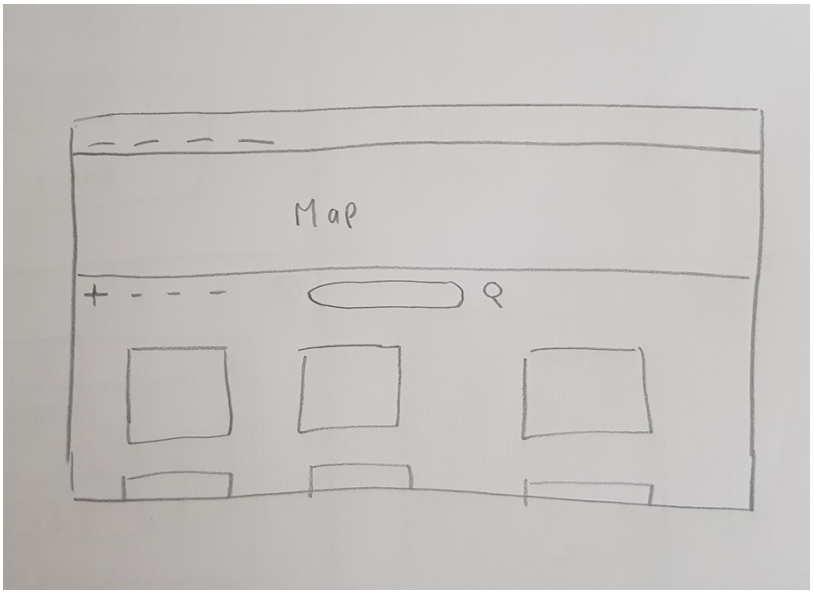
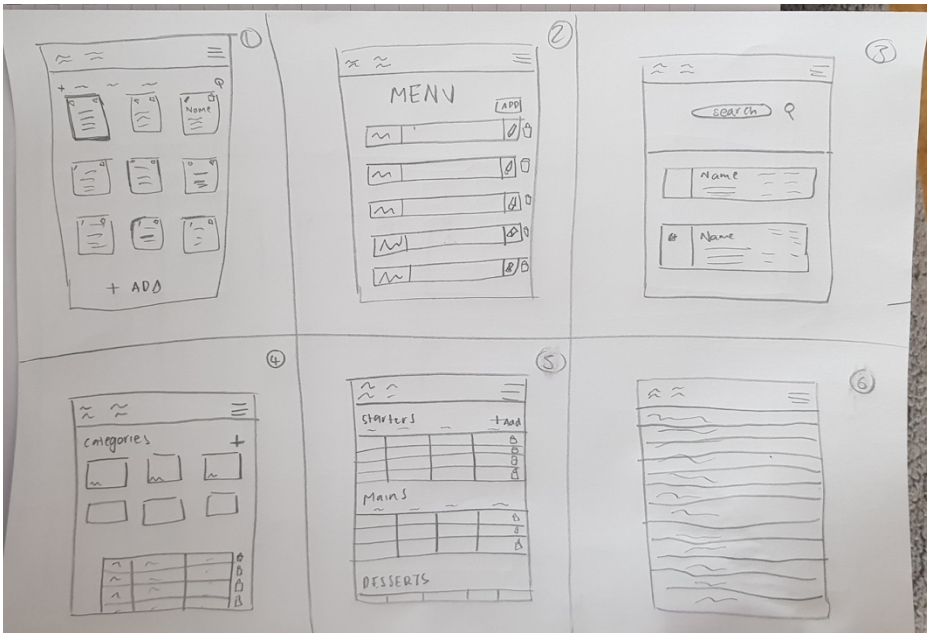


Figure 3 Final 1-UP for Veggie Quest home page

Restaurant Dashboard



Restaurant Dashboard 1

This is the 6-UP that I chose to turn into my 1-UP. I thought it was the most intuitive page to allow restaurants to add items to the menu, edit existing items and delete items + edit their public details. It is the best display of the data.

Restaurant Dashboard 2

I also liked this 6-UP, but I didn't think the menu items were displayed well.

Restaurant Dashboard 3

I didn't like 6-UP.

Restaurant Dashboard 4

I thought this 6-UP had too many sections.

Restaurant Dashboard 5

I also thought this 6-UP had too many sections.

Restaurant Dashboard 6

I thought this 6-UP had was too information heavy and wouldn't be a good design choice, especially for a mobile first design.

Restaurant Dashboard 1-UP

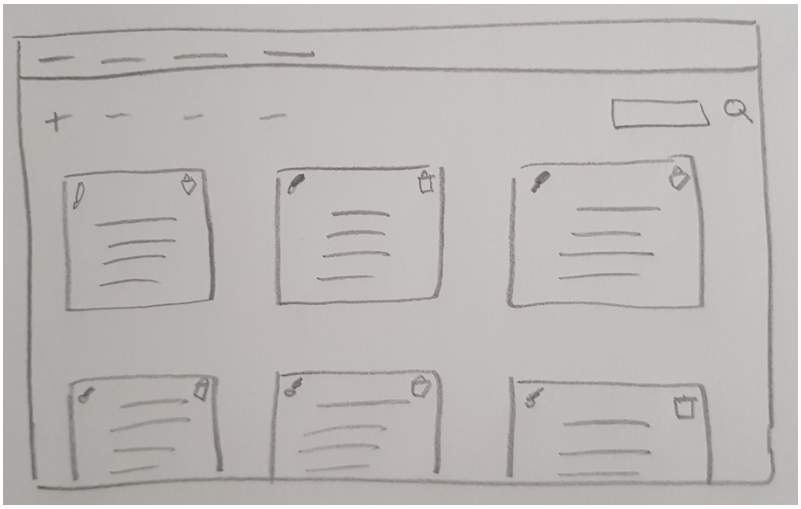


Figure 4 Final 1-UP for Restaurant Dashboard

Food Pop-Up

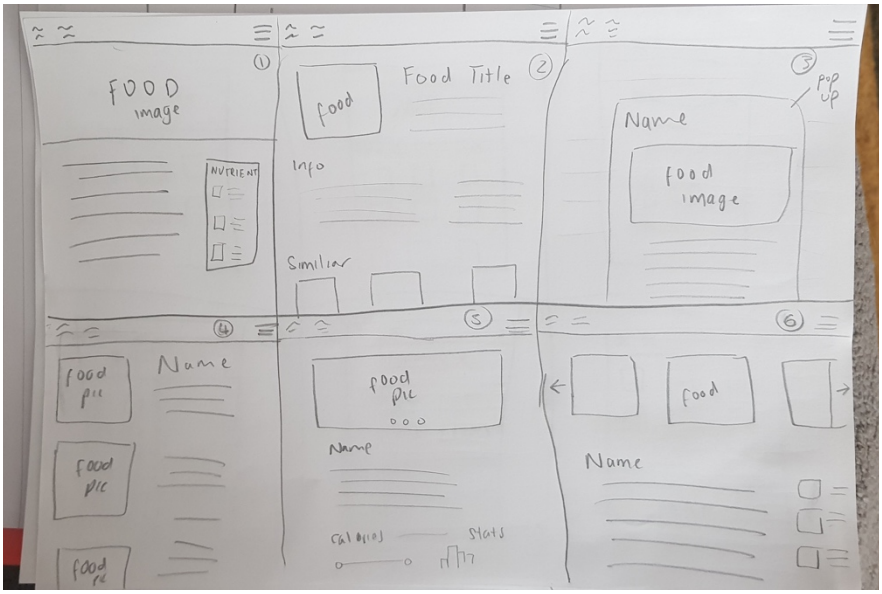


Figure 5 6-UP for Food Pop-Up

Food Pop-Up 1

I thought this 6-UP had too much text detail and wasn't visual enough.

Food Pop-Up 2

I liked this 6-UP because it was a smart way to arrange the data.

Food Pop-Up 3

I liked this 6-UP because it's very visual and displays the food well.

Food Pop-Up 4

This 6-UP was more text heavy but could display related food.

Food Pop-Up 5

This 6-UP is also more visual while retaining a good bit of text data.

Food Pop-Up 6

This 6-UP allowed related menu items to be displayed which I liked.

Food Pop-Up 1-UP

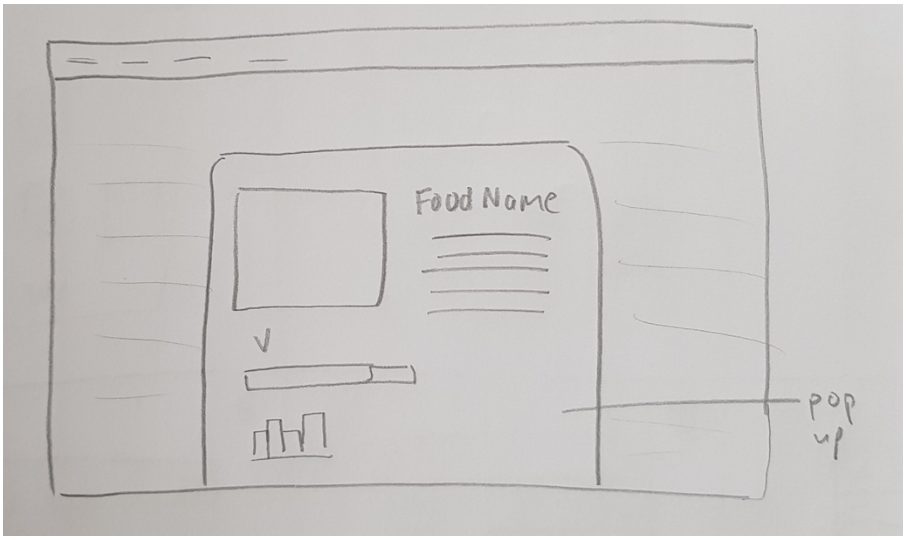


Figure 6 Final 1-UP for Food Pop-Up

Feasibility Testing

Part of developing a functional prototype was so that I could test how feasible the requirements were, etc. and how that plays into my risk assessment. It would allow me to assess how feasible the project would be as a whole. These are the following features that I wanted to have functional for the protype:

3. Accessing the site.
4. Being able to view food items loaded in from a database.
5. Viewing an interactive map with restaurants marked on it, loaded in from a database.
6. Adding food to the database and having it displayed on the site.

In terms of biggest risks, the ability to load restaurants from the database and display on a map was the main risk I had identified.

Each of the features are now fully functional, except for viewing restaurants on a map (which have been loaded from the database. There are restaurants on it, but they are hardcoded in. The reason I wasn't able to make this feature fully functional was because the map API I am using – Leaflet – only allows for mapping based on geographical coordinates, as opposed to street addresses. As I used PHP, I found a PHP library called Geocoder that would allow me to geocode street address into geographic coordinates, however I'm not skilled enough at PHP to successfully get the library working.

However, I found a NodeJS module called Node-geocoder that does the same thing, just using Node. As I am more skilled with Node, this is why I have decided to switch the main development language from PHP to NodeJS (as I mentioned previously in the report).

This means that I failed in bringing one of my biggest risks to functionality, but through extra research, have discovered the way in which I will solve this problem for the final product.

Risk Assessment

An essential part of the project was identifying and assessing risks, and planning what I would do if any of those risk jeopardised the project. These might include my knowledge level of various technologies or coding languages, time constraints, personal issues, etc.

My risk assessment is located at the bottom of this report at **APPENDIX B**.

Methodology Selection

I considered 3 types of development methodologies for this project; Waterfall, Prototyping and Agile.

Waterfall

The Waterfall methodology would probably be the most straightforward to use. It keeps things simple and splits up the development lifecycle into well-defined sections or timeframes.
A drawback of using Waterfall would be that not every aspect of the project would be at the same stage at the same time. For example, I may have already coded and integrated the search function but the menu editing function may only be at the beginning of the code stage.

I came to the conclusion that using the Waterfall methodology would be good for keeping track of deadlines and data over the entire project development cycle (5+ months).

Prototyping

With Prototyping, I liked that it included cycle stages for developing and reviewing prototypes. This is really important for a project like mine that will need that kind of prototyping and enhancement. The Prototype model takes into consideration customer satisfaction, which may be more suited to a project that has a commercial client, whereas my web application is being created so that BOTH the ‘eaters’ and restaurants are users, and doesn’t have a specific client.
One specific feature I liked about the Prototyping model was that it took maintaining the project into consideration because I want to continue developing the Vegan Food Finder after the hand in date at the end of semester two.

Agile

I liked Agile the best and think is most suited to my project is the Agile model, specifically the SCRUM model. I've used SCRUM before for projects in previous jobs and I think that it's the best suited to me personally, as well as my project. In my experience it allows you to plan for working software and actual objectives more, and focuses less on tools, processes or specific, ridged aims. It also allows you to respond to change more than following a ridged, set plan, which is important for a project such as this. I'm going to be doing user testing and working with restaurant throughout the entire development lifecycle, so I need to be able to take users/customers into account and possible change or add new objectives depending on newly identified issues, etc.
I think the SCRUM model will allow me to focus more on the important factors.

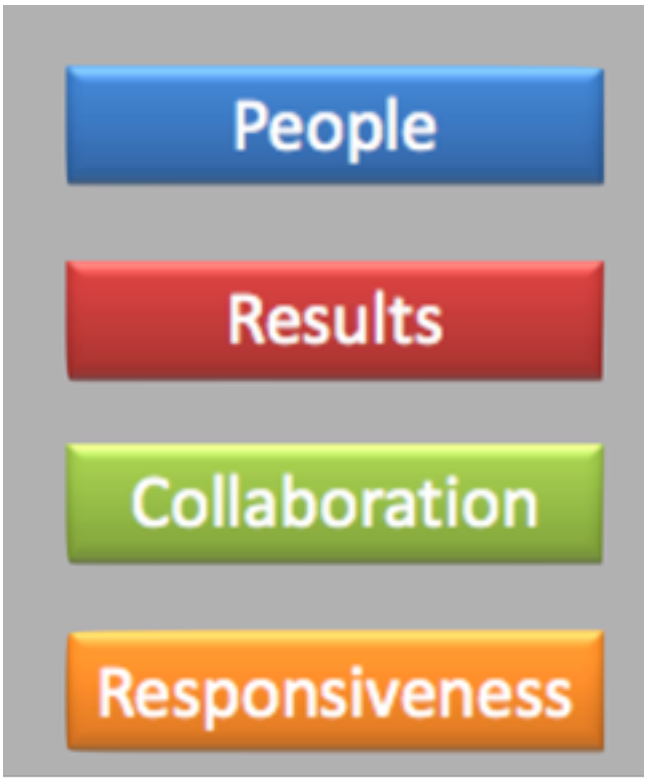


Figure 7 Diagram of Agile methodology

Conclusion

I decided to use the Agile methodology going forward. As I've stated I have used it in pervious projects and think it is best suited for my development cycle. Additionally, I also created a Gantt chart that can be used to keep track of project deadlines over the course of the 1st and 2nd semesters. It can be found at the bottom of this report at **APPENDIX C**.

Design

Branding

Even though design is not my strong point, I recognise the need for strong branding and a consistent design.

The main colours featured throughout the website are green (for form backgrounds) and white / black for text. This keeps the website looking clean and uncomplicated, and the green elicits feelings of health and the environment.

I also created a logo for the website, which features a leaf and a magnifying glass. The leaf represents a plant-based diet, while the magnifying glass represents searching for something. The logo is shown at Fig.8.

Figure 8

Mock-ups

Mock-ups were created using the layouts decided upon in the 6-UPs and 1-UPs. They're more in-depth ideas of how the Vegan Food Finder will look when it's created. I created mock-ups using a wire framing tool called Sketch. The mock-ups for my project can be found at **Appendix D**.

System Design

Site Map

I also created a site map that would help me in designing each of the pages. Creating the site map also helped me to ensure that the user flow of the website remained easily used and navigable. Although I had designed a very rough site map at the very start of planning, I had to create another, more updated site map, as there were issues that came up that I hadn't previously considered.
I eventually created a final site map, once I had a clearer image of how the site hierarchy would look in the finished project:

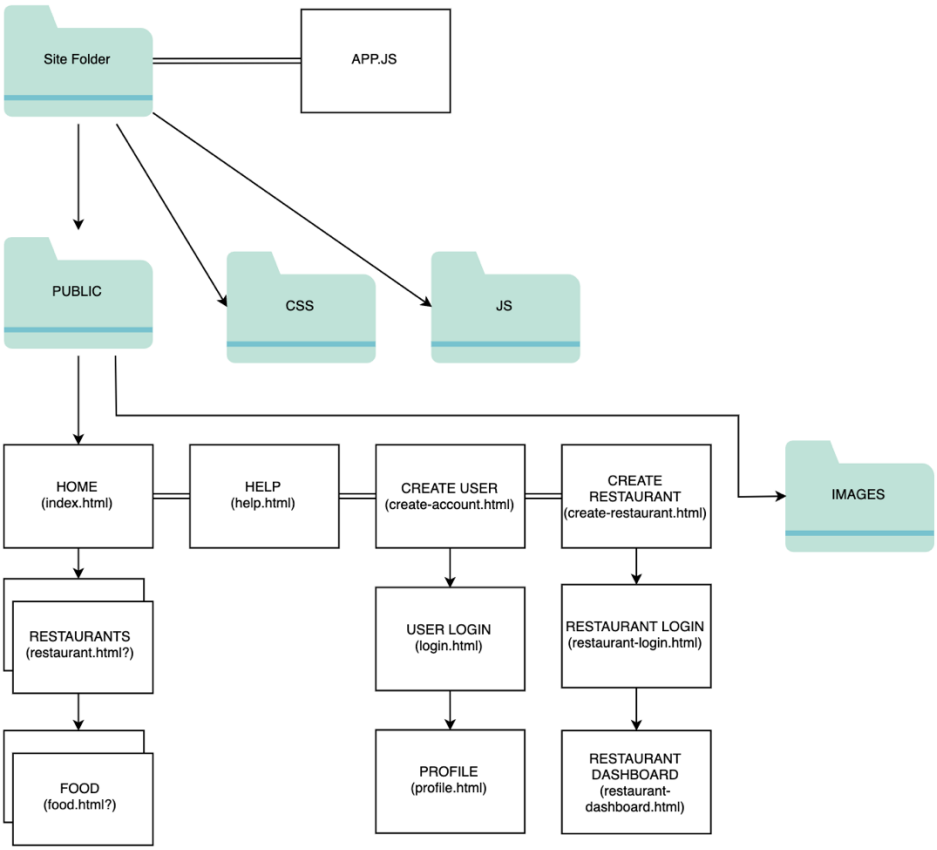


Figure 9 Final site map for Veggie Quest

A previous site map is located at **APPENDIX E**.

Client Server Model

The client server model highlights how the server-side and client sides of the website work and interact. In order to see where different technologies will be needed to complete the project to specification, we must create a client-server model:

Depending on if I use PHP or Node.js, that code will be activated on the Server side of the model, whereas HTML, CSS, JavaScript and jQuery will be executed on the Client side. Using Ajax allows you to let the two sides of the model interact.

In Fig. 10, you can see how the different technologies interact with each other based on the Client-Server model.

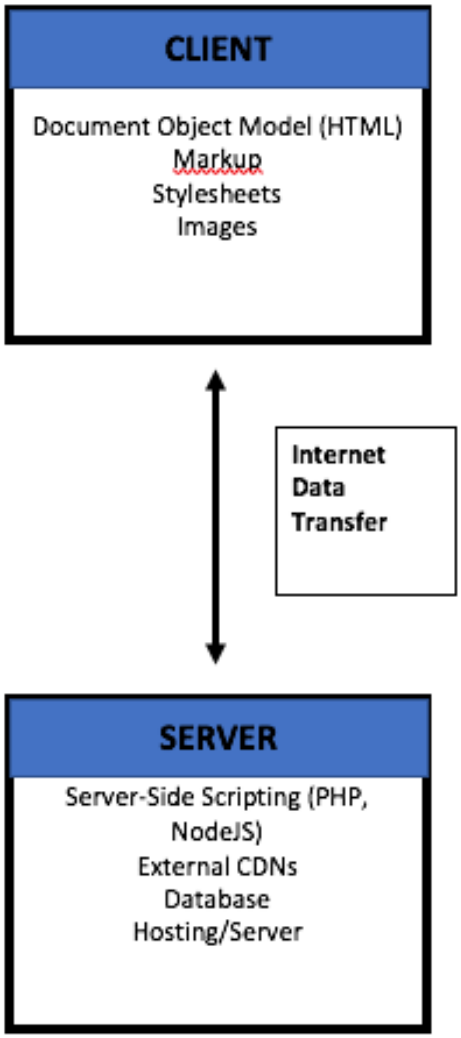


Figure 10 Client-Server model of Veggie Quest

Database Design

Although the database may be updated or changed throughout the development of the project, it's important to properly plan out and design a database system as early as possible. It means that when I'm developing the website, I can be fully prepared for unforeseen changes I need to make. Below is my current database design in the form of an Entity Relationship Diagram:

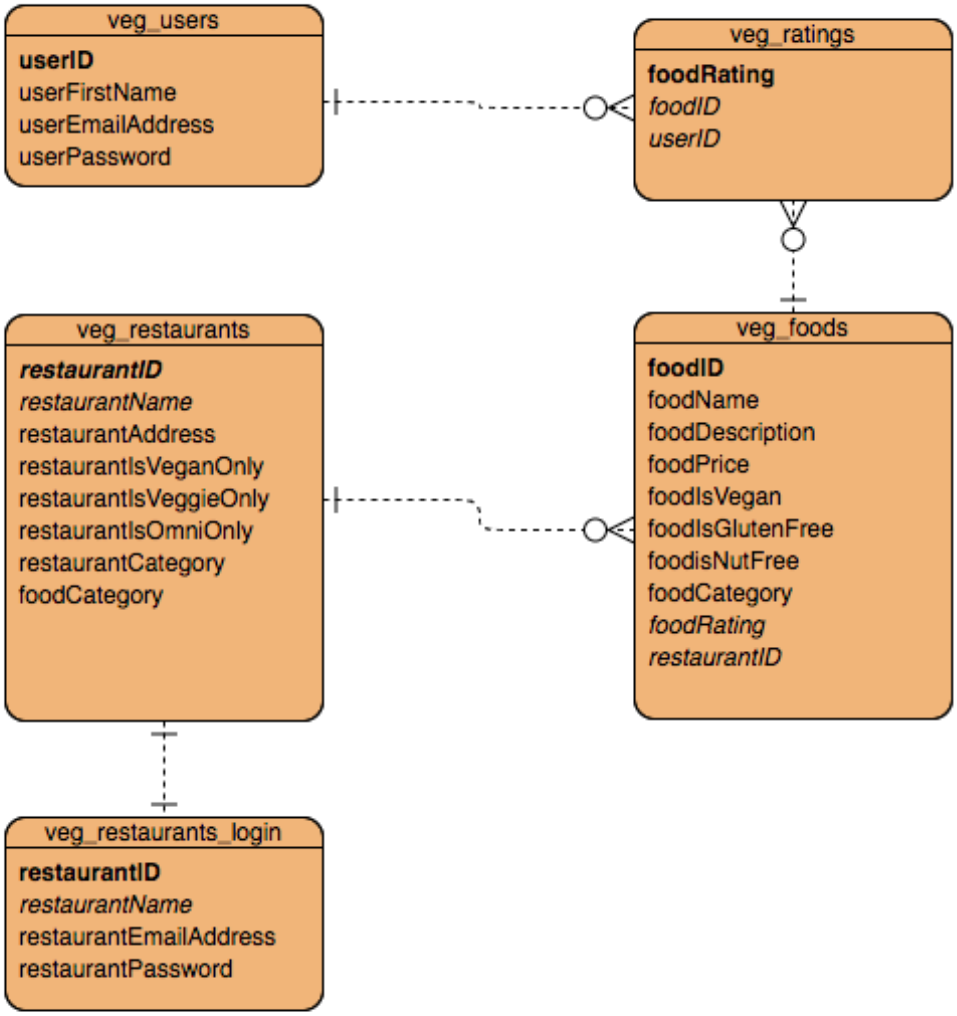


Figure 11 First working Database Design

Implementation

The next stage is to review various different technologies, frameworks, languages, etc. and weigh up their flaws and benefits in order to see what I should use. There are many different functions and features that the Vegan Food Finder will have upon completion, and different features or sections might have to use different technologies than the rest, which is why it's important to review different technologies and languages now, to be informed and prepared for the development.

Technology Review

Server-Side Technologies

PHP

PHP is a server-side coding language. It is run on over 244 million websites, making it the most common programming language for web development. (Kakkar, M., n.d.)PHP can be embedded within the DOM itself (within the HTML code), or it can be used within frameworks such as Laravel, or individual PHP files be can called when needed. PHP is used by data intensive sites such as Facebook, because it's good at communicating data with databases. (Pingdom Royal., 2019)

Because PHP is so widely used, there's a plethora of online resources that can be accessed for help tutorials, etc. PHP is also an open source language, which is something that is important to this project; I want the site to use 100% open source technologies.

While I've pointed out PHP's benefits, it also has its flaws. For example, PHP is slower than a lot of other programming languages because its variable types are checked at run-time and not when compiled. PHP is also known for its security flaws and poor error handling, which can increase development time and constrain time due to attempting to fix bugs and errors. (Chedygov, S., 2014)

NodeJS

NodeJS is a 'framework' of sorts that allows JavaScript to be ran/executed outside of the browser (i.e. on the server side). It allows developers to install 'modules' into their server environment that act as JavaScript libraries and offer a massive range of functionality. I think NodeJS is a great language and the ability to download these modules really helps to develop interactive and dynamic sites.

There are a lot of advantages to using NodeJS, such as the fact that it is easy to scale (important as I have plans for the project beyond the hand in date), and that it uses the Google V8 JavaScript engine, meaning it has a faster execution than other languages would have. (Quora.com., 2019)

The other massive advantage of using NodeJS is that, because it's a runtime environment for JavaScript, the fact that I am already fairly proficient in JavaScript will make the development of the Vegan Food Finder far easier. Using JavaScript via NodeJS means that you remove the need for another server-side language, and can use JavaScript for both client-side and server-side.

Additionally, NodeJS is open source, and you can download many open source NodeJS modules (such as Node-geocoder, which I will use to develop the map feature in my project). As I mentioned before, this is hugely important for me as I want my project to be developed using open source technologies.

Client-Side Technologies

HTML5 & CSS3

It's really important that my website is developed using the most up to data practices; this means using HTML5 and CSS3. HTML5 is cleaner and more readable and allows for creation of better forms (as in for data input) which is a huge plus for this website as there will be a lot of data input required from websites. One other advantage of HTML5 is that HTML5 offers an offline application cache feature – meaning that a loaded page can be viewed even if the user is offline – which I think is something I should include given that I expect a lot of the Vegan Food Finder's users to be accessing the site from mobile devices, a case where mobile data might not always be perfect. (TechArk Solutions., 2014)

CSS3 offers many new features from its predecessors. It allows for faster loading times, sleek animation and transitions, media queries (essential for a mobile-first design) and allows for smaller file sizes. (Creativeweblogix.com. (2012).)

Using both HTML5 and CSS3 is essential to making sure my website is using the most current practices, for making sure my website looks good and is easy for users to use, and to ensure I can make my website is fully mobile-first in its design.

SCSS

SCSS, short for 'Sassy Cascading Style Sheets', is a CSS pre-processor which gives you more features that aren't part of the native CSS functionality, while being fully compatible with native CSS.
However, I have never used it before. I am not that great with CSS, anyway, let alone trying to learn a new styling language.

jQuery & JavaScript

JavaScript is a programming language that is executed on the client-side (i.e. on the browser). It allows you to manipulate how HTML and CSS look, as well as handling data and functions.
jQuery is a JavaScript library (sort of like a framework) that makes it easier to use JavaScript on a website. It can be used to easily change the data being displayed on the page, and can also be used to process Ajax requests, making it easier to communicate with the server-side code (either in Node.js, PHP, or another language).

Frameworks

Bootstrap

Bootstrap is a framework for CSS that allows developers to create mobile-first websites that are fully responsive. I have a lot of experience using Bootstrap and would consider myself fairly proficient in it, which is why I am choosing to use it to develop my website. Because it

has a lot of built-in styling, it means that developers can spend less time on styling their site and more time on actual coding and development; a huge plus for me as I am not that great at web design (as opposed to web development).

One downside is that sometimes, a website designed using Bootstrap for the front end doesn't really look unique and can look like a lot of other websites.

ExpressJS

ExpressJS is a framework for Node.js that allows developers to set up local hosts and routing (handling GET and POST requests) in order to create websites. In other university modules that covered Node.js, we always used Express so it makes sense that I will also use it in my final project (if I use Node.js).

Databases

MySQL

MySQL is an open source relational database management system. I've used it previously, so it was the first database management system that came to mind for this project. It is easy to use with both of the languages I named above (PHP & NodeJS), so naturally it is a good fit for my project. MySQL is able to deal with large amounts of data, so a lot of websites like Facebook. (Pingdom Royal, 2019) However, unlike NodeJS or other database systems, development isn't community driven (i.e. only the Oracle publishes updates and patches, the community cannot work on its development)., which means that MySQL has stagnated in its development, having only one major update in the past few years. It can also have problems with scaling, which might be an issue further down the line when I continue to develop the Vegan Food Finder. (Mack, J. and twitter., F., 2014)
Once again, the fact that MySQL is open source is a huge plus for me.

MongoDB

MongoDB is another database management system, that differs from MySQL in that it is not relational. Instead of storing data in multiple tables, MongoDB stores data in 'JSON-like documents'. Instead of using SQL to query the database, it uses its own query language (MQL). There is an open-source version of MongoDB, as well as one that is proprietary and requires a license. Although I was hesitant to use MongoDB, as I haven't used it before, I would like to learn how to use it and use it in future projects. (Tutorialspoint.com., n.d)

Other Technologies

Leaflet & Node-Geocoder

Leaflet is a JavaScript library that allows web developers to have mobile friendly apps within their web applications. It's completely open source, enjoys community-contributed source code, has a lot of documentation for its API and has lots of plugins to really give your map data-driven functionality. It's extremely lightweight too, with a JavaScript file size of 38KB.

It is possible to display markers on a map based on variables, so I could use Leaflet to plot the restaurants and foods on the home page map of Veggie Quest. To do this, it requires the map coordinates of a location (as opposed to the street address). I can get this information using Node-Geocoder, a Node.js module that allows for geocoding by getting map data from

an API). (Tomlin, R., 2020) There are a number of APIs I could use for getting the map data, so I will have to investigate which one works best. It is also open source.

Technology Selection

After reviewing the various different languages, technologies, frameworks and APIs that I can use, I had to come to a final selection on what would be used in the creation of the Veggie Quest website.

Server-Side

On the server side of the project, I decided on using Node.js. Although I had used PHP before in a number of different university projects (especially in Year 2), I thought that Node.js would be a better choice.
I picked it because:

- We have been using it in recent Final Year modules, so it was fresh in my memory.
- There is an abundance of Node.js modules (similar to a JavaScript library) available to developers. Some are built in, but many are available for download. There a number of Node.js modules I used in my project, which I will go over later.
- I hadn't used it before, and I was drawn in by the opportunity to learn a new technology.

Client Side

HTML5 & Styling

I decided that I would use HTML5 in my project. HTML5 is the most up-to-date version of HTML and contains the most up to date features. HTML5 has better functionality for forms, and as my website would depend on a number of different form inputs, it made sense to use.

With styling, CSS3 is, again, the most up to date version of CSS, and has new features like animations, etc., which I could make use of in my website (if I choose, but I probably won't). I decided to use CSS3 because I have a few years' experience with it, so it will be far easier to use as opposed to trying to learn something like SCSS while also working on the backend of my project.

JavaScript & jQuery

Having used jQuery before, it was an easy selection to make. jQuery allows for lot of customisation of the HTML DOM. Because I will need to load database information from the server-side, jQuery is useful because it can be used to output the data onto the HTML page. jQuery would also be useful in the use of cookies.

Frameworks

Bootstrap

I decided on using the Bootstrap CSS framework for my project. I have used Bootstrap in many of my past university assignments and personal projects. I am not great with styling (or design in general), so using Bootstrap makes a lot of sense. I was able to create things such as navigation bars and forms far easier than I would have been able to with just plain HTML and CSS. It saves a lot of time which can be better spent working on the back end of my website.

ExpressJS

As expected, I used the Node.js framework – ExpressJS – for setting up my localhost and routing URLs.

Databases

MySQL

For storing restaurant, food and user data, the only real choice for me was MySQL. I have used it in a couple of other university projects (mostly in Year 2), so there wasn't anything extra to learn. I can run a local instance of a MySQL server on my laptop by using the MAMP software, which lets me use PHPMyAdmin UI to interact with it.

It is also a relational database system, and this seemed the best option due to the nature of the data I would need for the project (foods been connected to restaurants as well as users, etc.).
I had trouble in designing my database, in that I found it hard to get the relationships between tables set up and ensuring there wouldn't be foreign key constraints when I tried to delete certain rows. This led to some trouble, which I will talk about later in the report.

Looking back, while I still think MySQL was the best choice, I think it might have been useful to learn MongoDB for us in this project, as it would have been interesting to see how it compared to MySQL.

Other Technologies

As well as the above listed technologies, there were a few others that I used in the development of my project. This includes things such as Node.js modules, APIs, libraries, etc.

I am also using the Visual Studio Code editor for working on my project

Technology Use

Now that I have reviewed and selected my technologies, the following section covers how they were used and implemented.

Client Side

I used HTML5, CSS3, JavaScript and jQuery for the client side of the website. I have used each of technologies before, so using them was fairly straightforward. I also used the CSS framework – Bootstrap – to style the web pages.
I used HTML5 syntax where I could, making sure it was up to date with the latest conventions. This includes HTML elements such as '<nav>', '<section>' and '<header>', among others.

Bootstrap and HTML5 made it easy to create forms, which you can see in Fig. 12. The Bootstrap grid system made it easy to create a website with mobile-first design.

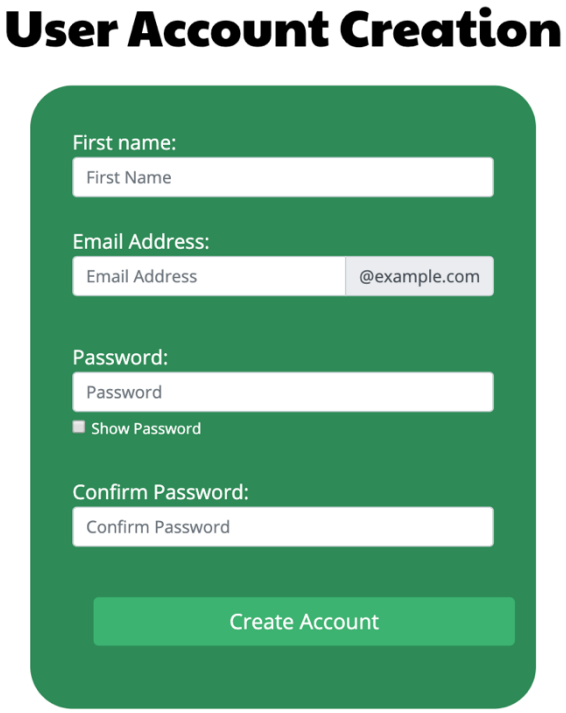


Figure 12

jQuery – the JavaScript library – made it easy to route to Node.js requests and allowed to me to use Ajax connections to pass data back and forth between the client-side and the server-side. This let me get data such as restaurant information (by passing URL parameters to the Node.js route), and it let me perform login checks, which would tell the jQuery if a user was logged in and changed navbar elements depending on what the answer was. I also used native JavaScript for a couple of different functions throughout the website.

Server Side

Node.js was the language that I used for coding the server-side, backend components of the website. I had used it before for some other university modules in Final Year, so there was a learning curve involved, but I was able to code most of the planned components of the project. It was interesting to learn, although I found the asynchronous nature of it hard to get used to at first. I was able to find a lot of tutorials and content online that helped me during the project development.

ExpressJS was really useful in setting up routes for the different requests that made up Veggie Quest, as you can see in Fig.13 and Fig.14. I was able to get the URL query (e.g. 'veggie-quest/food.html?3') on the client-side by using jQuery, pass it to the server-side Node.js route using Ajax, then return a database query (MySQL).

```
$(document).ready(function(){
    // Gets the query from the URL path. I.e. 'food.html?2'
    var queryString = decodeURIComponent(window.location.search);
    var queries = queryString.substring(1);
    var foodID;
    for (var i = 0; i < queries.length; i++){
        foodID = queries[i];
    }

    // AJAX connection to the server to GET the relevent food information.
    $.ajax({
        type: 'get',
        // Passing URL query to apptest.js/getFoodInfo/
        url: '/getFoodInfo/' + foodID,
        success: function (data) {
            // On success, loads relevent food information into the correct HTML DOM objects.
            $.each(data.foodInformation, function(i, data){
                $(document).prop('title', data.foodName);
                $("#foodPic").attr("src",data.foodPic);
                if (data.foodIsVegan = 'No'){
                    $("#foodDiet").text('Vegetarian');
                } else {
                    $("#foodDiet").text('Vegan Ⓥ');
                }
                if (data.foodisGlutenFree = 'No'){
```

Figure 13

```
app.get('/getFoodInfo/:id', function(req, res){ // Gets food information f
    foodID = req.params.id;
    const queryString = "SELECT veg_foods.foodID, veg_foods.foodName, veg_
    getConnection().query(queryString, [foodID], (err, results, fields) =>
        if (err){
            console.log("Could not load food information: " + err);
            res.sendStatus(500);
            return;
        }
        console.log("Loaded food information successfully");
        res.send({"foodInformation": results});
    });
}); // Gets food information for front end view.
```

Figure 14

Express was also really useful for its method for sending HTTP responses, that could be used to send data back to the client side via jQuery (and Ajax), as well as alerting users to the reason why something hasn't worked (through 'res.sendStatus(xxx)' or through 'res.send()' with a custom message).

MySQL Database

Using the MySQL database with Node.js sending the SQL queries was fairly straightforward. I had to download the mysql driver for node.js, but that was all it took to get database quires working without any hassle.

My main problem with the database was the relational nature of MySQL, and ensuring that the foreign keys, etc. were all set up correctly. For example, when deleting rows from the 'veg_foods' table, I ran into errors because it is linked to the 'veg_ratings' table (where user favourites are stored) due to foreign key restraints. I had to do a work around, but it is an example of how I used the MySQL database. Fig.15 shows the current table relation view of the database. As you can see, it doesn't have the setup.

Figure 15

Other Technologies

Node.js Modules, Libraries, APIs

Throughout the development of Veggie Quest, I made use of a number of different Node.js Modules, libraries, plug ins and APIs.

One of the positives about Node.js that I identified in the Technology Review was the ability to use Node.js modules. Modules being used in Veggie Quest include:

- **Node-GeoCoder**: allows me to geocode the restaurant address, meaning I can get the coordinates (longitude and latitude) based on the address that the restaurant owner has entered. I use it in conjunction with the OpenStreetMap API to get the geographic data. Fig.16 is a short snippet showing how we pass variables to Node-Geocode.

```
geoCoder.geocode(newAddress)
.then((res)=> {
    // If we can't locate the restaurant, we delete it from the table,
    if (res === undefined || res.length == 0){…
    } else {
        // If location data is found, input the longitude and latitude
        newLongitude = res[0].longitude;
        newLatitude = res[0].latitude;
        console.log('Lat: '+ newLatitude + "   Long: " +newLongitude);
        // Extra bit of error handling.
```

Figure 16

- **MySQL Driver**: makes the connection to a MySQL database easier. It also makes error handling easier, simplifying the development and debugging process.
- **Formidable**: makes passing form data to the server-side far easier, and aids in parsing form data and simplifies file uploads. I use it in conjunction with the File System module to facilitate file uploads in the form of restaurant and food pictures.
- **File System**: facilitates making changes to the file system on a computer. I have used it with the Formidable module to facilitate image uploads.
- **BodyParser**: middleware that helps to parse data coming from the body (i.e. the HTML page). It is useful for reading in form data in the Node.js file.
- **Cookies**: Node.js library that facilities the setting of cookies. I use cookies for checking if browsers are logged in, allowing me to use jQuery to make cosmetic appearances based on that, and to verify if someone is logged in when they try certain functions (like adding a favourite food, for example).
- **Morgan**: is middleware that helps to get track of what HTTP requests are being logged. The requests show up in the Terminal, and it is useful for keeping track of requests when testing out features and trying to debug errors.
- **Nodemon**: is a tool that automatically restarts the Node.js application. Normally, after every change you make to your Node.js code, you have to manually restart the application in Terminal or in the Command Line. Nodemon removes that step, saving a lot of time and simplifying the development, testing and debugging workflow.

As well as the above Node.js modules, I also used a JavaScript library called LeafletJS. LeafletJS is used for creating interactive maps. I used it to plot restaurants on a map on the Veggie Quest homepage by using the coordinates that Node-Geocode generates (with the OpenStreetMap API). LeafletJS also allows you to create custom icons, so that each restaurant on the map has a different icon that changes depending on what cuisine the restaurant serves (Fig.17)(Fig.18). The map imaging (at the minute, just a standard map) is supplied by MapBox.
LeafletJS is fully open source.

```
var myIcon = L.divIcon({
    className: 'custom-div-icon restaurant' + data.restaurantID,
    //Each cuisine type has it's own custom icon.
    html: '<img class="mapMarkers" src = "../images/food-icons/' + data.restaurantCategory + '.png"/>',
    iconSize:     [60, 60],
    shadowSize:   [50, 64],
    iconAnchor:   [22, 22],
    shadowAnchor: [4, 62],
    popupAnchor:  [-3, -36]
});
```

Figure 17

Figure 18

The API that I used in Veggie Quest is the OpenStreetMaps API. OpenStreetMaps provides the geocoding services that I access by using Node-GeoCode. basically, Node-Geocode passes the street address of a restaurant to OpenStreetMaps, and I get the longitude and latitude (and a lot more data) back.

Code Editor

I used the Visual Studio Code editor to work on my project. It's really useful because it includes a built-in instance of Terminal (the inclusion of which really streamlines your workflow). It also offers built in debugging and 'IntelliSense', which autocompletes code elements and offers quick parameter information. Having a code editor like this made working on the project far easier.

Notable Challenges & Achievements

Challenge - Time Constraints

One of the major challenges that I faced in the development of the Veggie Quest (and the rest of my university work) was time constraints. Time constraints arose due to the Covid-19 pandemic, which meant university classes being cancelled from mid-March 2020, all university facilities being closed, and having to stay your house. I have ADHD and find it hard to work in a noisy environment, so I would normally do work in a library or in the Block16 Mac Suites. Additionally, due to a family member being 'high risk', I had to move out of my house and into a friend's house.

Due to these time constraints, I wasn't able to implement some of the features that I had originally planned for Veggie Quest, such as food search and a food filter. I have no doubt in my ability to code such features, it was just the time constraints that led to them being excluded from the final website.

I also wanted to have a verification process in place for the restaurant signup process, instead of letting just anybody sign up as a restaurant. This would probably have to be done manually, but that is the case even with Deliveroo and HappyCow, the two websites I was inspired by.

Challenge - Database Relationships

Another challenge I faced was getting the MySQL database to work in certain situations (such as when trying to delete rows that had foreign key constraints).
This mainly came up when trying to delete foods as a restaurant. It was fine if a food hadn't been favourited by a user, but if a food had been favourited, it wouldn't delete. I got around this by first checking if a food had been favourited at all, then sent a different SQL query to the database depending on the result. It isn't the best work around (Fig.19), but it works.

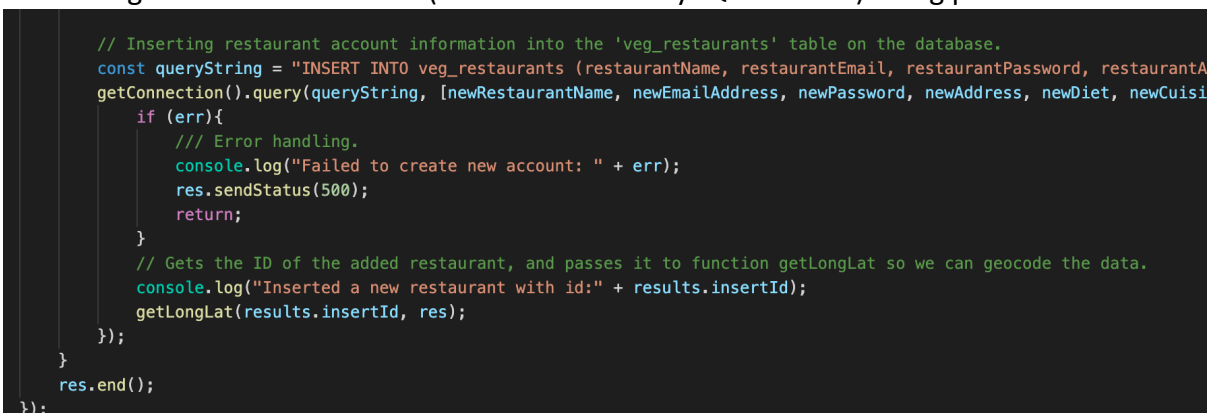
```
app.post('/deleteFood/:id',  function(req, res) { // Allows restaurant to remove food from their menu.
    foodID = req.params.id;

    // Checks if a food has been favourited by any users.
    getConnection().query('SELECT * FROM veg_ratings WHERE foodID = ?', [foodID], function(error, results, fields) {
        if (error){
            console.log("Could not delete food: " + error);
            res.sendStatus(500);
            return;
        }
        if (!results.length) {
            if (results)
                // If it has not been favourited, call deleteFood();
                console.log('No Favs');
                deleteFood(foodID, res);
        } else {
            // If it has  been favourited, call deleteFoodWithFavs();
            console.log('Faves');
            deleteFoodWithFavs(foodID, res);
        }
    });
}); // Allows restaurant to remove food from their menu.
```

Figure 19

Achievement – Geocoding

One notable achievement was getting the Node-Geocoder module working so that I could process restaurant street address, get the coordinates (longitude, latitude) and input it into the correct restaurant row in the database 'veg_restaurants' table.
To do this in the restaurant account creation process, I set the coordinate columns in the database to have a default value of 'null' (meaning they could be empty). When a restaurant is creating an account, Node.js sends an insert SQL query to the database to insert the restaurant into the database as normal, then fires a function that gets the coordinates. If for some reason this doesn't work (an incorrect address) and coordinates aren't calculated, the restaurant is deleted from the table again, and the user is alerted to the reason their account hasn't been created and lets them try again.

Fig.20 shows the the SQL query being send, and then the 'getLongLat' function being called and having the id of the last row (a feature of the MySQL JS driver) being passed to it.



```
        // Inserting restaurant account information into the 'veg_restaurants' table on the database.
        const queryString = "INSERT INTO veg_restaurants (restaurantName, restaurantEmail, restaurantPassword, restaurantA
        getConnection().query(queryString, [newRestaurantName, newEmailAddress, newPassword, newAddress, newDiet, newCuisi
            if (err){
                /// Error handling.
                console.log("Failed to create new account: " + err);
                res.sendStatus(500);
                return;
            }
            // Gets the ID of the added restaurant, and passes it to function getLongLat so we can geocode the data.
            console.log("Inserted a new restaurant with id:" + results.insertId);
            getLongLat(results.insertId, res);
        });
    }
    res.end();
});
```

Figure 20

Fig.20 shows how the 'getLongLat' function works by getting the address of the last input restaurant and passes it to the Node-GeoCode function in 'addLongLat'. Fig.21 also shows the error handling that has been put in place, so a veg_restaurant row doesn't end up without any coordinate values.

```
function getLongLat(insertId, resE){ // Using node-geocoder, we get the address from the newly added restaurant, and get the longitude and latitude of it.
    console.log(insertId);
    const queryString = "SELECT restaurantAddress FROM veg_restaurants WHERE restaurantID = ?";
    getConnection().query(queryString, [insertId], (err, results, fields) =>{
        // Error handling.
        if (err){
            console.log("Could not load restaurant address: " + err);
            resE.sendStatus(500);
            return;
        }
        // Gets address and passes it to fuction addLongLat.
        console.log("Loaded restaurant address successfully" + results[0].restaurantAddress);
        addLongLat(results[0].restaurantAddress, insertId, resE);
    });
    function addLongLat(newAddress, insertId, resE){
        console.log("Getting longitude and latitude!")
        // Using node-geocoder (with the OpenStreetMap API) to calculate the longitude and latitude of the restaurant.
        geoCoder.geocode(newAddress)
        .then((res)=> {
            // If we can't locate the restaurant, we delete it from the table, and alert the user.
            if (res === undefined || res.length == 0){…
            } else {
                // If location data is found, input the longitude and latitude into variables.
                newLongitude = res[0].longitude;
                newLatitude = res[0].latitude;
                console.log('Lat: '+ newLatitude + "   Long: " +newLongitude);
                // Extra bit of error handling.
                if (newLongitude.length = 0){
                    const queryString = "DELETE FROM veg_restaurants WHERE restaurantID = ?";
                    getConnection().query(queryString, [insertId], (err, results, fields) =>{
                        if (err){
                            console.log("Internal Error: " + err);
                            resE.sendStatus(500);
                            return;
                        }
                        console.log("Restaurant Not Added.");
                    });
                    resE.status(statusCode).send('Could not locate your restaurant. Try including just your street address. Or try removing the postcode.');
                    return;
                } else {
                    // If everything is fine, updates the restaurant's row in 'veg_restaurants' with the latitude and longitude.
                    const queryString = "UPDATE veg_restaurants SET restaurantLat = ?, restaurantLong = ? WHERE restaurantID = ?";
                    getConnection().query(queryString, [newLatitude, newLongitude, insertId], (err, results, fields) =>{
                        if (err){
                            console.log("Failed to add cordinates: " + err);
                            resE.status(statusCode).send('Could not locate your restaurant. Please try again.');
                            return;
                        }
                        console.log("Inserted coordinates.");
                        resE.redirect('restaurant-login.html');
                    });
```

Figure 21

Achievement – Plotting Map Coordinates

Another notable achievement was plotting the restaurants onto the LeafletJS map on the home page (and allocating each one a custom icon based on what cuisine they served). To do this I had to get the latitude and longitude for each restaurant (which I've described above).

In the JavaScript file for the home page, I first initialised the LeafletJS map (Fig.22). When I pulled in the data from the 'veg_restaurants' table I added them to the map by using the 'L.Marker' layer function (Fig.23). As you can see, it plots the map point by coordinates, and it also specifies that icon will be used for that plot point.

```
// Initialising LeafletJS map
var mymap = L.map('mapid').setView([54.5950596, -5.9293545], 14);
L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{y}?access_token={accessToken}', {
    attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors, <a href="https
    maxZoom: 18,
    // Using MapBox to provide map layers.
    id: 'mapbox/streets-v11',
    tileSize: 512,
    zoomOffset: -1,
    accessToken: 'pk.eyJ1IjoiOThsaWFta2VsbHkiLCJhIjoiY2s5Y3E1bm0yMDBmdDNtbWp5ZGxqbWJnZCJ9.DCGQEDt8rBgLT4lpjRRmqw'
}).addTo(mymap);
```

Figure 22

```
// Adds a new marker to the map for every restaurant.
L.marker([data.restaurantLat, data.restaurantLong], {icon: myIcon}).addTo(mymap).bindPopup(data.restaurantCategory + " food at " + data.restaurantName);
;
```

Figure 23

For the icons, I used the 'L.divIcon' function to specify what icon each of the cuisines will use. (Fig.24)

```
// Loops through each of the objects in the JSON file.
$.each(data.restaurants, function(i, data){
    // Creating an array of custom icons to use for the LeafletJS map, based on the cuisine type of a restaurant.
    var myIcon = L.divIcon({
        className: 'custom-div-icon restaurant' + data.restaurantID,
        //Each cuisine type has it's own custom icon.
        html: '<img class="mapMarkers" src = "../images/food-icons/' + data.restaurantCategory + '.png"/>',
        iconSize:     [60, 60],
        shadowSize:   [50, 64],
        iconAnchor:   [22, 22],
        shadowAnchor: [4, 62],
        popupAnchor:  [-3, -36]
    });
```

Figure 24

Figure 25

Testing

There are two main methodologies when it comes to software testing, 'Black Box Testing' and 'White Box Testing'. Black Box testing is testing the software without knowledge of the backend functionality and code, and it is usually carried out by dedicated testers. White Box testing, on the other hand, is testing that is done with a working knowledge of the code behind the software and is usually carried out by software developers themselves.

Unfortunately, due to the Covid-19 crisis, I wasn't able to get anyone to carry out Black Box testing. However, I was able to carry out White Box testing myself.

I also carried out testing across multiple different browsers to ensure the website will display consistently and maintain full functionality throughout the different browsers.

Test Process

In order to test my website, I created a list of requirements from the aims that I wrote up before the development stage (found in the Introduction section of this report), as well as a few extra testing requirements that arose to newly implemented features (that weren't envisioned in the pre-development phase). Below is an example of a test case. The rest can be found at **APPENDIX F**.

Test ID	Description	Outcome	Response (if fail)
1	Users will be allowed to create an account.	PASS – Account created, and user redirected to login page.	

Test Results

In total, there were 34 test cases, 31 of which passed, 3 of which failed, giving me a pass rate of 91.18%. The failed tests show the areas in which I can improve on in future development of Veggie Quest and show what could have been implemented with less time constraints or a better knowledge of node.

User Survey

As part of the testing process, we were supposed to carry out a user survey on people not in our household. They would be queried on things such as their age and gender, their level of computer literacy, and their experience using the website.

Unfortunately, due to Covid-19 and the social distancing restrictions that came in place as a result of it, carrying out the user survey was impossible. Below are the questions which I was going to ask:

1. What is your age?
2. What is your gender?
3. What level of computer literacy do you hold?
4. What would you rate your experience of using Veggie Quest? (1-10)
5. What would you rate the design and feel of Veggie Quest? (1-10)

6. How easy would you rate Veggie Quest in terms of ease of use? (1-10)
7. How likely would you be to recommend Veggie Quest to a friend? (1-10)
8. You are in the city centre and you're hungry? How likely are you to use Veggie Quest to find something to eat? (1-10)

Evaluation

Evaluation of Project Outcomes

First of all, looking back at the stated project aims and user stories (**APPENDIX A**) at the beginning of the design phase, it is clear that I did not meet all of the stated aims. Personally, I would put this down to a combination of time constraints (made worse by the Covid-19 situation), but also to maybe a lack of prioritising early on. If I had made more progress on Veggie Quest in the first few months of 2020, I might have been in a better situation, however, I could not have foreseen what would happen due to the pandemic.

Some key features that I missed out on having included the ability to search for certain foods, and filter via category or dietary requirements. These were big selling points of the project, and I am regretful that I wasn't able to implement them on schedule. The regret isn't only based on the fact that the Major Project makes up a significant percentage of our mark, but it's also based off a personal want to make the website better, to make it a viable website that real people want to use.

Another thing I don't feel is the best it could be is the CSS and styling of the website. This is always something I have struggled with. I am far better at the back-end coding aspects of things. It reflects my job aspirations for after graduation, but I also feel that Interactive Multimedia Design has equipped me with decent design skills.
One thing I should have done in this project, is taking more time to fine tune CSS, and spend more time learning various techniques, or how to take more advantage of the Bootstrap framework. I also could have learnt how to use SCSS, because it may have allowed me to realise more ways to style websites, rather than just CSS and Bootstrap, which is my usual go to.

Working with databases (especially relational databases) is something I need to improve at. The implementation of a database in the major project is fine, but it could have been better. In future, I am going to experiment with MongoDB. There are some workarounds I had to do in order to make the database do what I needed it to, and I feel as though I would not have had to do those work arounds had I knew more about how relational databases work. However, the fact that I was able to do these work arounds shows that I am competent enough in coding (and in Node.js) to solve problems that are presented to me.

With all that said, however, the final project still presents a working, functional, MVP. It still allows restaurants to:

- upload menus online
- edit those menus
- edit their information

It also allows users to:

- users to browse vegan and vegetarian food near them
- change their information
- add foods their favourite foods list

Additionally, it shows the location of vegan and vegetarian restaurants on a map in an engaging way.

All that considered, I am more than happy with Veggie Quest in its final form (to date). I am happy to present it as my COM559 Major Project, and it is something that I will continue to work on beyond the scope of a university module.

One thing that I made clear from the start was the desire to keep my website built on as much open-source technology as possible. I fully support the community-driven aspect of open-source software (and the idea of open-source in general). I'm glad to be able to say that I was able to *nearly* stick to this goal 100%.
Open source technology I used throughout the project includes:

- Node.js
- LeafletJS
- jQuery
- JavaScript (not open source, but open)
- OpenStreetMap API
- Various different Node.js modules

The only aspect of the website that isn't open source is the map imagery, which is provided by MapBox. Although it's a shame that it isn't open source, the vast majority of the website **is** built on open source technology, something that I am passionate about.

Evaluation of Development Methodology

The development methodology I decided to use for this project was the Agile. It proved to be successful as it allowed me to easily work through the requirements. Sometimes I find it hard to keep track of all the little, individual things that need done in a project (especially if I am working alone), but Agile is good for keeping me aware of what needs done. I have experience using the Agile methodology in a professional environment (in SCRUM form), so I was expecting it to be an effective project development plan methodology.

Conclusion

Summary

The COM559 major project was, essentially, the culmination of my entire time at university. It is supposed to be challenging and it's supposed to be an opportunity to apply all of the relevant skills I have learnt throughout the various modules of Interactive Multimedia Design. It was challenging, but I enjoyed having to problem solve and learn new technologies in order to meet those challenges.
I now have more experience with and far more knowledge of Node.js than I had at the start of the project, and that is something that I consider to be a huge plus going into the graduate job market. During the development of the project, I found other technologies that I would like to gain experience with, and I think I will look into working with the different MEAN stack technologies (MongoDB, Express.js, AngularJS and Node.js).

I have a far better understanding of what it's like to make a RESTful API, and what is required in the creation of a website intended for public use.

I also think that the working on Veggie Quest for the COM559 Major Project has revealed to myself my strengths, but also moreso my weaknesses (such as styling, designing, time management to a degree), giving me a chance to reflect on and work on them.
Even though the final project doesn't have everything that I had envisioned it to have, I am still happy with the work that I have done, and it has given me a clearer vision of what I need to do in order to better myself, and my professional skills (which becomes even more important after graduation).

Even though a clear plan was set out, we have seen that plans don't always go forward as intended. I think in future, I will prioritise more time at the start of a project hammering out the low-level (but important) features so that I can dedicate more time to working on, implementing and perfecting the big complex features when it comes to crunch time.

Future Potential

I fully plan on continuing to develop Veggie Quest beyond the scope of the COM559 Major Project. I fully believe that it has potential to be used by the public, and that the demand is there for it.

I am going to implement the features that I set out in the original aims that I failed to implement for the final hand in data, and I am going to try use 100% open-source technology to build the site (at the minute, everything is open-source except for the map imagery).

I am excited to continue working on Veggie Quest, and excited to learn and develop more skills while working on it.

Appendices

Appendix A – User Stories

#:	USER STORY:	SCENARIOS:	OUTCOMES:
1	As a user, I want to create an account so I can save likes.	9. User goes to account creation page and fills in their data in accordance with the requirements. 10. User goes to account creation page and fills in their data but does not meet requirements.	1. Account for user created, able to log in. 2. Account for use isn't created, user alerted to requirements they must meet (password length, etc.).
2	As a user, I want to log in and view the website so I can save likes.	1. User goes to login page and enters their username and password correctly. 2. User goes to login page and enters an incorrect username and password combination.	1. User able to log in and view the site. 2. User alerted that they have input incorrect details, allowed to try again.
3	As a user, I want to look at food that I liked previously so I don't have to manually find the menu item again.	1. User has previously liked menu items. 2. User hasn't previously liked any menu items.	1. User's previously liked menu items will be displayed in the 'liked food' section. 2. The 'liked food' section will be empty.
4	As a user, I want to search or filter results so I can find something to my liking.	1. User uses search function to type in query. 2. User uses filter function to filter out results.	1. Menu items that match the input query are returned. 2. Only menu items that match the filter terms are returned.
5	As a user, I want to see more details about a specific food.	1. User clicks on a menu item box.	1. Details about the food are shown on to the user.
6	As a user, I like a menu item, so I want	1. User clicks on the restaurant title of a menu item.	1. A page detailing the specific restaurant is brought up, showing

	to see what else the restaurant sells.		the other foods they have on their menu.
7	As a user, I want to contact a restaurant so I can book a table or make a query.	1. User clicks on phone number or email address on restaurant page.	1. User's phone attempts to call number or open email app depending on what contact detail user clicked on.
8	As a user, I want to share a menu item or restaurant with someone else.	1. User clicks on 'share button'.	1. Link is copied to user's device via click board.

#:	USER STORY:	SCENARIOS:	OUTCOMES:
1	As a restaurant, I want to create an account so I can use the site.	1. Restaurant goes to restaurant account creation and fills in required data.	1. Website admins review application and create account for restaurant. Email sent to restaurant.
2	As a restaurant, I want to log in so I can use the site.	1. Restaurant tries to log in with correct data. 2. Restaurant tries to log in with incorrect data.	1. Log in successful. 2. Log in unsuccessful, alerted to enter correct information.
3	As a restaurant, I want to view my menu so I can assess it.	1. Restaurant navigates to the 'restaurant dashboard' page.	1. Menu of restaurant displayed on screen, with options to add, edit or remove items from the menu.
4	As a restaurant, I to add a menu item so it's displayed on the website.	1. Restaurant clicks on 'Add Menu Item' button and is brought to the menu item creation page. Data entered meets requirements.	1. Menu is updated with a new menu item reflecting the entered data.
5	As a restaurant, I want to edit a menu item to update it.	1. Restaurant clicks on a menu item and clicks the 'Edit Menu Item' button. Data is changed.	1. Menu is updated with the menu item displaying the newly updated data.
6	As a restaurant, I want to remove a menu item, so it isn't displayed any more.	1. Restaurant clicks on the 'Delete Menu Item' button.	1. Menu does not feature the deleted menu item anymore.
7	As a restaurant, I to change our public details so customers can get it touch, etc.	1. Restaurant clicks on 'change details button' and enters new details.	1. Restaurant's public page updated to reflect newly updated data.

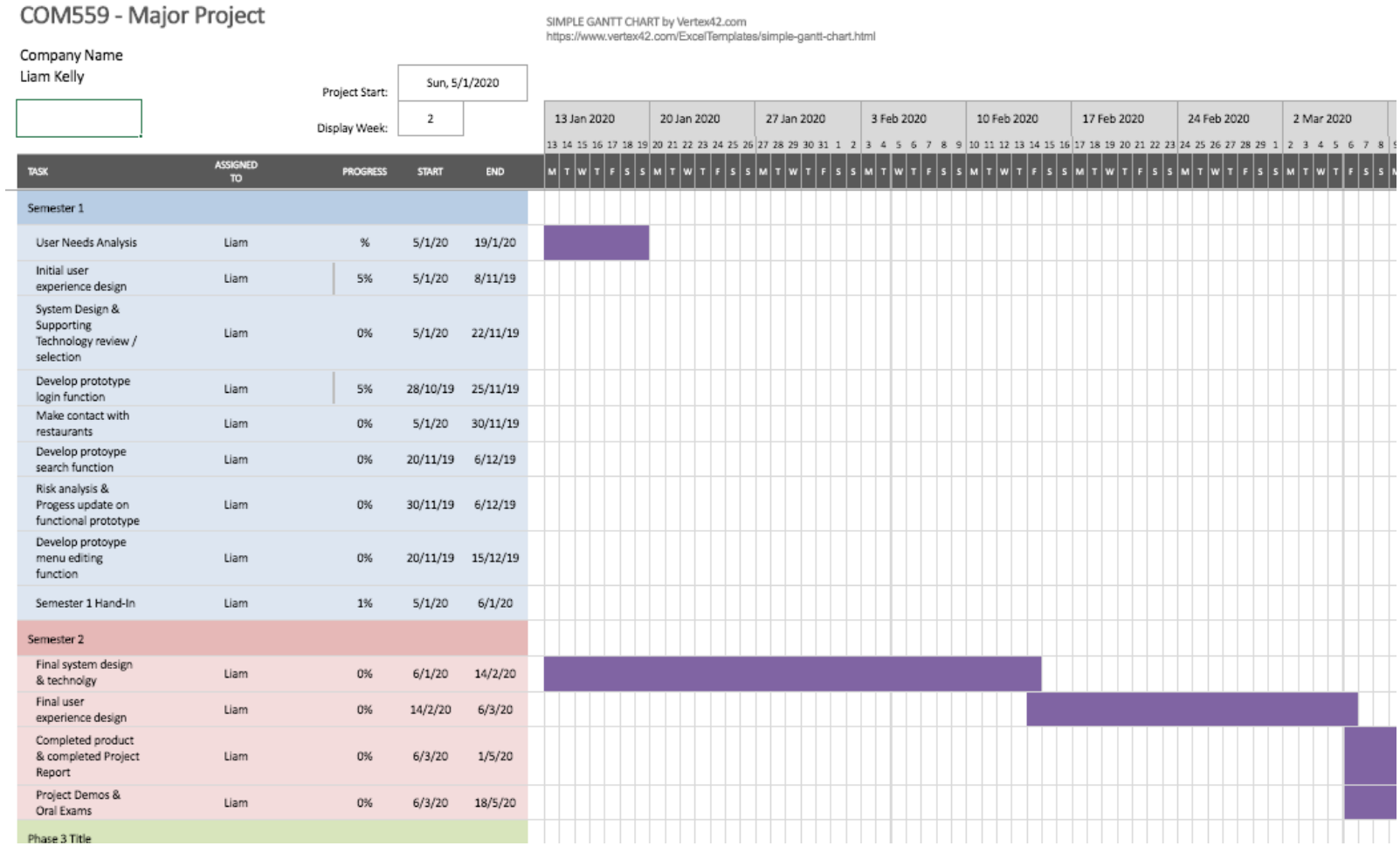
Appendix B – Risk Assessment

RISK	LEVEL	ELIMINATION	BACKUP

PHP skill	6 – Medium	Practicing PHP and doing tutorials should help me mitigate most of this risk.	Use another language or technology, such as NodeJS.
Database setup and design	8 – High	The Vegan Food Finder will have a lot of data, so it's essential to have a database design that isn't cumbersome and lends itself to easy flow of data.	Although small changes to the database might be made throughout the project's development, it's important to have the database design as fully laid out and planned as possible. Now.
Map API	9 – Very High	I plan to use the open source Leaflet JavaScript library for the map functionality. I will use it to display food items and restaurants.	As it's an open source library, there is a lot of resources online featuring guidance and help. I am also fairly skilled at JavaScript, which minimises the risk.
Search & Filter Function	7 – High	I need to create a search function that allows users to search by food items.	I have previous experience in creating search functions, which mitigates this risk.
User Login	7 – High	It's essential for users to be able to log into the website. The login feature must be secure.	I have created login features in the past, but they haven't been secure or used encryption. I will need to learn how to develop an encrypted login system.
Restaurant Login and Dashboard	10 – Very High	It is ESSENTIAL that restaurants can log onto the website, and add menu items, edit their menus, etc. To do this I will need to: 11. Develop a login feature solely for restraint use. 12. Create a 'dashboard' that allows restaurants to view their menus, details/stats	As I've stated above, I've created login systems before, but need to learn how to make them secure and encrypted. For the restaurant dashboard I have planned what it will look like (using a 6UP) so I am confident in creating this feature. I need to create an account creation and verification system for the restaurants. This is something that would have to be done manually. Even major companies such as Deliveroo and JustEat would use a system of manual account creation and verification for their

		about their menus, etc. 13. Figure out how to implement a system that allows restaurants to sign up to the website and get verified.	'restaurant users'. (Restaurants.deliveroo.com., n.d.)
Bootstrap CSS Framework	4 – Low	I am using the Bootstrap CSS framework to style the website.	I have a lot of experience using Bootstrap. I cannot see any major risks associated with this.
Time Constraints	5 – Moderate	It's essential I stay on top of my workload and don't let deadlines start to pile up.	This has been a problem for me in the past, but by using our hard and soft deadlines + Gantt charts, I can better organise my time.
Unforeseen Circumstances	3 – Low	Unforeseen circumstances (such as illness, bereavement, etc.) cannot be planned for.	By keeping on top of my workload, I can minimise the effect that an unforeseen event would have on my project development.

Appendix C – Gantt Chart



COM559 - Major Project

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Company Name
Liam Kelly

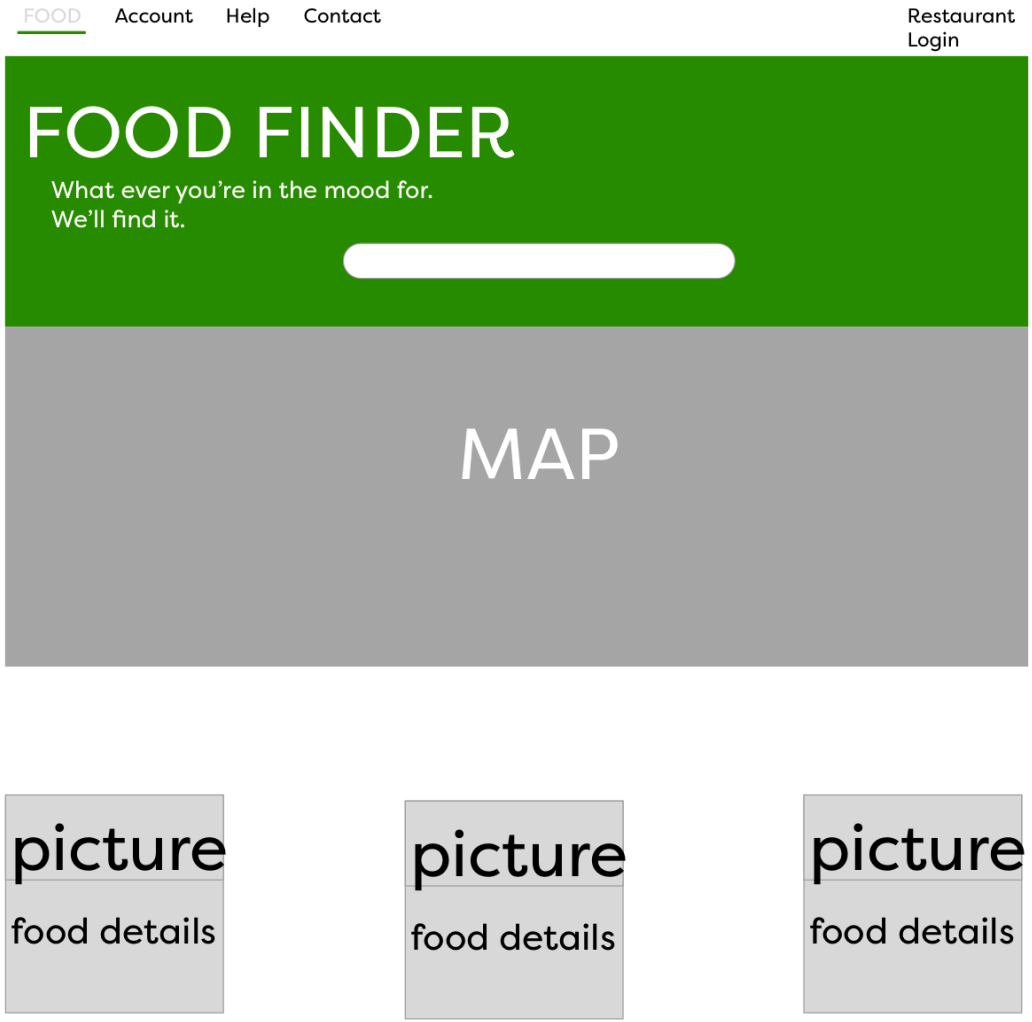
Project Start: Sun, 5/1/2020
Display Week: 2

TASK	ASSIGNED TO	PROGRESS	START	END
Semester 1				
User Needs Analysis	Liam	%	5/1/20	19/1/20
Initial user experience design	Liam	5%	5/1/20	8/11/19
System Design & Supporting Technology review / selection	Liam	0%	5/1/20	22/11/19
Develop prototype login function	Liam	5%	28/10/19	25/11/19
Make contact with restaurants	Liam	0%	5/1/20	30/11/19
Develop protoype search function	Liam	0%	20/11/19	6/12/19
Risk analysis & Progess update on functional prototype	Liam	0%	30/11/19	6/12/19
Develop protoype menu editing function	Liam	0%	20/11/19	15/12/19
Semester 1 Hand-In	Liam	1%	5/1/20	6/1/20
Semester 2				
Final system design & technolgy	Liam	0%	6/1/20	14/2/20
Final user experience design	Liam	0%	14/2/20	6/3/20
Completed product & completed Project Report	Liam	0%	6/3/20	1/5/20
Project Demos & Oral Exams	Liam	0%	6/3/20	18/5/20
Phase 3 Title				

Screenshot of initial Gantt Chart

Appendix D – Mock-ups

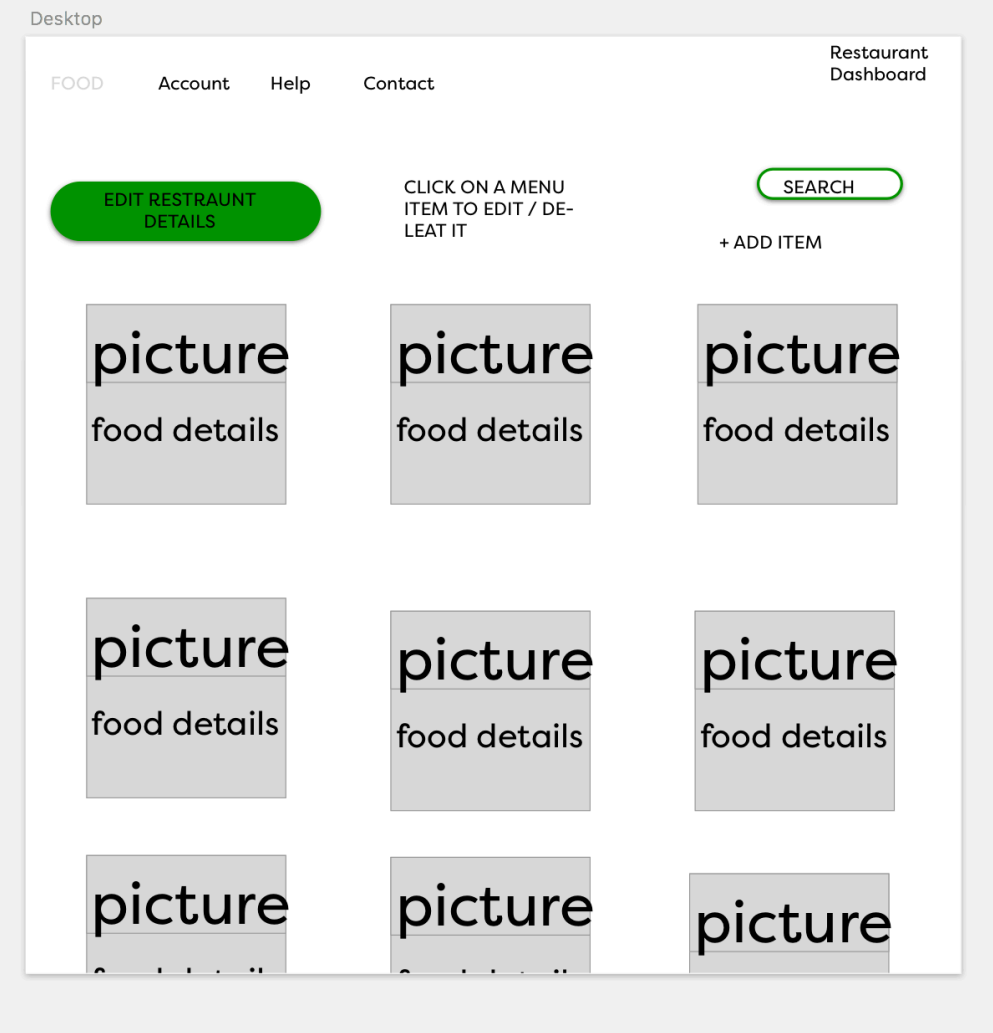
Home Page



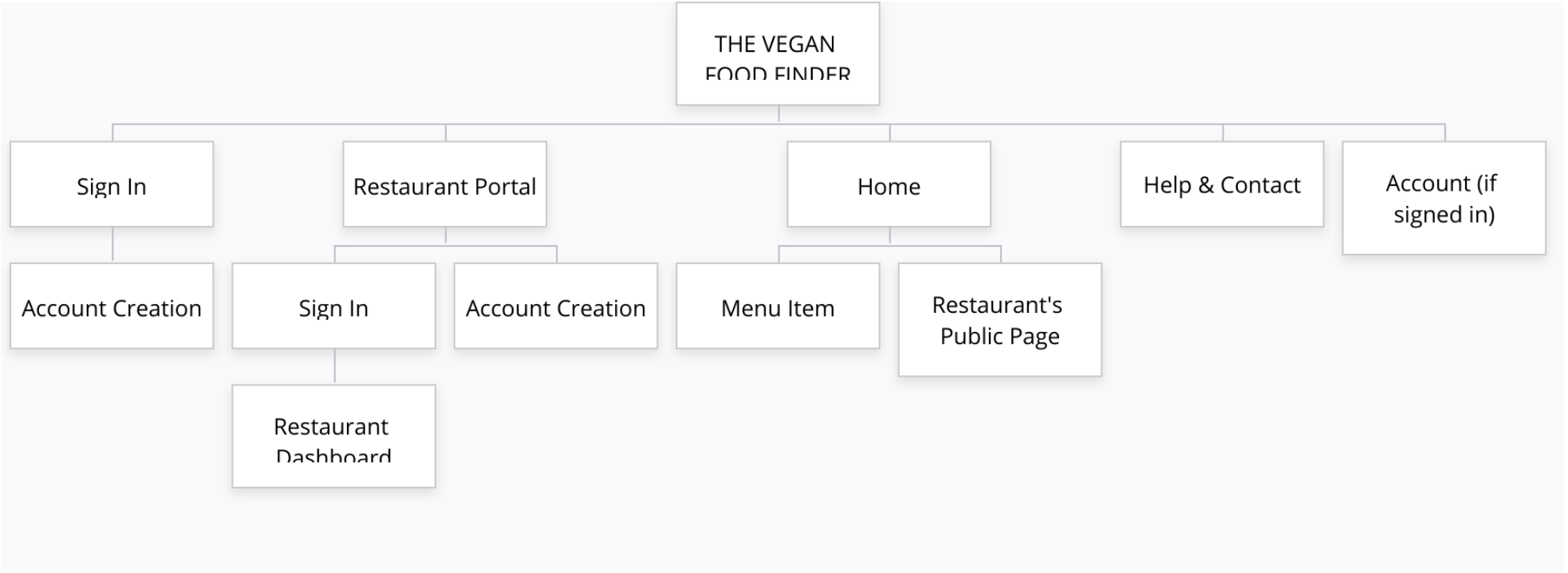
Food Pop-Up

Desktop

Restaurant Dashboard Page



Appendix E – Site Map



Appendix F – Testing

Test ID	Description	Outcome	Response (if fail)
1	Users will be allowed to create an account.	PASS – Account created, and user redirected to login page.	
2	Users will attempt to create an account but enter in an already registered email address.	PASS – Account not created. User alerted to reasons why.	
3	Users will attempt to create an account but won't have met the password requirements.	PASS – Account not created. User alerted to reasons why.	
4	Users will attempt to create an account, but their entered passwords didn't match each other.	PASS – Account not created. User alerted to reasons why.	
5	Users try to log into account with correct email address and password.	PASS – User logged in and redirected to user profile page.	
6	Users try to log into account with incorrect email address and password.	PASS – User not logged in and alerted why.	
7	Users attempt to change their password.	PASS – User password changed.	

8	Users attempt to change their, enter in wrong old password.	PASS – User password not changed; user alerted why.	
9	Users attempt to change their password; new password does not meet requirements.	PASS – User password not changed; user alerted why.	
10	Users favourite foods show up in favourite foods table.	PASS	
11	Users click on one of their favourite foods.	PASS – User directed to relevant food page.	
12	Restaurants will be allowed to create an account.	PASS – Restaurants account created, and user redirected to restaurant login page.	
13	Restaurants will attempt to create an account but enter in an already registered email address.	PASS – Restaurant account not created. User alerted to reasons why.	
14	Restaurants will attempt to create an account but won't have met the password or form requirements.	PASS – Restaurant account not created. User alerted to reasons why.	
15	Restaurants will attempt to create an account, but their entered passwords didn't match each other.	PASS – Restaurant account not created. User alerted to reasons why.	

16	Restaurants try to log into account with correct email address and password.	PASS – Restaurant logged in and redirected to user profile page.	
17	Restaurants try to log into account with incorrect email address and password.	PASS – Restaurant not logged in and alerted why.	
18	Restaurants attempt to change their password.	PASS – Restaurant password changed.	
19	Restaurants attempt to change their, enter in wrong old password.	PASS – User password not changed; user alerted why.	
20	Restaurants attempt to change their password; new password does not meet requirements.	PASS – Restaurant password not changed; user alerted why.	
21	Restaurant's menu loads on restaurant dashboard.	PASS	
22	Restaurant click on one of their menu items.	PASS – Restaurant directed to relevant food page.	
23	Restaurants attempt to change edit their restaurant information, uploading new picture.	PASS – Restaurant information changes.	
24	Restaurants attempt to change edit their restaurant information, without new picture.	FAIL – Restaurant information does not change.	Error in FileSystem Node.js. Error not caught.

25	Restaurants attempt to add a new food item with all fields complete.	PASS – New food added.	
26	Restaurants attempt to add a new food item without all required fields completed.	PASS – New food not added. Restaurant alerted why.	
27	Restaurants attempt to edit a food item, uploading new photo.	PASS – Food item updated.	
28	Restaurants attempt to edit a food item, not uploading new photo.	FAIL – Food item not updated.	Error in FileSystem Node.js. Error not caught.
29	Restaurants attempt to delete a food item.	PASS – Food item deleted.	
30	Restaurant user goes to 'edit-food.html?x' page of a food that isn't from their menu.	FAIL – Restaurant can view other restaurant's food item in the 'edit food' form.	Restaurant can see another restaurant's food data but can't do anything with it.
31	Restaurant user goes to 'edit-food.html?x' page of a food that isn't from their menu AND tries to edit it.	PASS – Restaurant NOT allowed to edit another restaurant's food.	
32	User goes to 'edit-food.html?x' page or restaurant dashboard page.	PASS – User does not see anything.	

33	Restaurants load onto home page, along with being plotted onto map.	PASS	
34	Foods load onto home page, along with being plotted onto map.	PASS	

References

Booth, R. (2019). Vegan food becomes UK's fastest growing takeaway. *The Guardian*. [online] Available at: https://www.theguardian.com/food/2019/aug/28/vegan-food-becomes-uk-fastest-growing-takeaway [Accessed 10 Oct. 2019].

Chedygov, S. (2014). *What makes PHP slower than Java or C#?*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/1407603/what-makes-php-slower-than-java-or-c [Accessed 1 Jan. 2020].

Creativeweblogix.com. (2012). *Advantages of CSS3 over CSS*. [online] Available at: http://www.creativeweblogix.com/blog/advantages-of-css3-over-css [Accessed 1 Jan. 2020].

Kakkar, M. (n.d.). *Interesting Facts About PHP - GeeksforGeeks*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/interesting-facts-about-php/ [Accessed 1 Jan. 2020].

Mack, J. and twitter., F. (2014). *Five Advantages & Disadvantages Of MySQL*. [online] Datarealm. Available at: https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/ [Accessed 11 Dec. 2019].

Pingdom Royal. (2019). *Exploring the Software Behind Facebook, the World's Largest Social Media Site - Pingdom Royal*. [online] Available at: https://royal.pingdom.com/the-software-behind-facebook/ [Accessed 5 Jan. 2020].

Quora.com. (2019). *What are the advantages of node.js? - Quora*. [online] Available at: https://www.quora.com/What-are-the-advantages-of-node-js-1 [Accessed 1 Jan. 2020].

Restaurants.deliveroo.com. (n.d.). *Sign up to become a Deliveroo Restaurant Partner*. [online] Available at: https://restaurants.deliveroo.com/en-gb/apply [Accessed 20 Dec. 2019].

Sainsbury's (2019). *Future of Food Report*. [online] Available at: https://www.about.sainsburys.co.uk/~/media/Files/S/Sainsburys/pdf-downloads/future-of-food-08.pdf [Accessed 7 Oct. 2019].

TechArk Solutions. (2014). *Advantages of HTML5 - TechArk Solutions*. [online] Available at: https://gotechark.com/blog/advantages-html5/ [Accessed 1 Jan. 2020].

Tomlin, R., 2020. *An Introduction To Geocoding Using Node.Js*. [online] Medium. Available at: <https://medium.com/javascript-in-plain-english/an-introduction-to-geocoding-using-node-js-fe1a5d3aa05c> [Accessed 11 March 2020].

Tutorialspoint.com. n.d. *Mongodb - Advantages*. [online] Available at: <https://www.tutorialspoint.com/mongodb/mongodb_advantages.htm> [Accessed 1 March 2020].

Xia, V. (2017). What is Mobile First Design? Why It's Important & How To Make It?. *Medium*. [online] Available at: https://medium.com/@Vincentxia77/what-is-mobile-first-design-why-its-important-how-to-make-it-7d3cf2e29d00 [Accessed 1 Jan. 2020].