



Let's Go!

Major Project (COM553/COM599)

Final Year

Kirsty McCrystal - B00595739

Peer Group – 5

Mentor - Dr Huiru Zheng

University of Ulster

Acknowledgements

I would like to say thank you to my sister, my sister's fiancée and my boyfriend for helping me to come up with the let's go web application idea, testing, giving me advice and supporting me throughout the year.

I would also like to say thank you to Dr Huiru Zheng for being my mentor, for directing me into the right direction and giving my advice throughout the project.

Table of Contents

Abstract.....	1
1. Introduction	1
1.1 Introduce the challenge of the Web application.....	1
1.2 Aims and objectives	2
1.2.1 Aims	2
1.2.2 Objectives.....	2
1.3 Overview of work to be undertaken	3
1.5 Overview of the rest of report	3
2. Concept definition and testing	4
2.1 Project Idea	4
2.1.1 Idea generation	4
2.2 Requirements specification	5
2.2.1 Purpose of the Project.....	5
2.2.2 Project goal	5
2.2.3 The stakeholders.....	6
2.2.4 Assumptions and consideration.....	7
2.2.5 Constraints	7
2.2.6 Scope of the project	7
2.2.7 Requirements.....	8
2.3 Paper prototyping	10
2.3.1 – 6ups	11
2.4 Feasibility testing.....	13
2.5 Methodology selection	15
2.5.1 Risks and Challenges.....	15
2.5.2 Legal, social, ethical and professional issues.....	16
2.5.3 Chosen methodology.....	16
2.5.4 Work plan and time management.....	16
3. Design.....	18
3.1 User Experience design evolution.....	18
3.1.1 Colour scheme	18
3.1.2 Simplicity Structure.....	19

3.1.3 Scannable Content.....	19
3.1.4 Search Engine Optimization	20
3.1.5 The search criteria page	20
3.2 System design.....	23
3.2.1 Client-Server Model.....	23
3.2.2 - MVC Diagram (Model, View, Controller).....	24
3.2.2 Web System Architecture	25
3.2.3 Chosen Languages, frameworks and API's	26
3.3 Logic design	29
3.3.1 Design Patterns	29
3.3.2 PHP Includes.....	29
3.4 Database design	30
3.4.1 Entity relationship diagram	30
4. Implementation.....	32
4.1 Technology/tool selection	32
4.2 Technology/tool use	33
4.3 Notable challenges.....	33
4.3.1 – Drawing freehand the search radius	33
4.3.2 Calculating the search area	35
4.4 Notable achievements	36
4.3.3 – Building the search query.....	36
4.3.4 - Displaying venues details	39
4.3.5 – Log in system with account.....	44
4.3.6 – Storing and deleting venues within the Database	46
4.4 – Other notable achievements.....	48
5. Testing.....	49
5.1 - Test approach selection.....	49
5.1.1 - Functionality testing	49
5.1.2 – Interface & usability testing	50
5.1.3 - Compatibility testing	50
5.1.4 - Performance testing.....	50
5.1.5 - Security testing	50
5.2 - Test process	51

5.3 - Test results	51
5.3.1 - Functionality results	51
5.3.2 – Non-functional requirements result’s (Compatibility testing)	53
5.4 - Requirements and objectives responses	54
5.4.1 - Task Completion Rate.....	54
5.4.2 – User Feedback on the functionality and usability.....	55
6 – Evaluation	56
6.1 - Evaluate test results.....	56
6.1.1 - Requirements and objectives results	56
6.2 - Evaluate project outcomes.....	57
6.3 - Evaluate the methodology	58
7 - Conclusion	60
7.1 - Summarise report	60
7.2 - Reflect on what happened	60
7.3 - Reflect on your role	60
7.4 - Future work.....	61
Reference	62
Appendices - A.1 – Requirements Survey Responses.....	64
A.2 – Constraints Solutions	66
A.3 – Use case diagram	68
A.4 – Functional Requirements	69
A.5 - Style board for web application	73
A.6 – Model View Controller	74
A.7 – Tables showing the structure of each data	75
A.8 – Cross Browser Testing	75
A.9 – Responsive Testing	75

Abstract

This report presents the development of the Let's Go web application which aims to help users find venues within an area of their choice by defining the results through a set of criteria's. Users can then store the venues within their personal accounts and then gain directions from their current position to the venue. Stating and using the requirements and constraints is used to make sure the application can be the user's needs by managing the project and methodologies to reach the application aims and objective. The results of testing the application shows how it can be used by multiple users within different age groups, and has met all of the requirements, objectives and aims.

1. Introduction

1.1 Introduce the challenge of the Web application

The web has become more advance and diverse since the invention of the World Wide Web, from static graphics and texts to advance animation and dynamic features. The web today offers the ability to process, store and transmit data over the web, such as locating venues. The Let's Go web application allows users to search for venues and store venues within their own account, therefore the first challenge was to create personal account for the user, which includes a venue storage system. Web applications today now have to be more responsive to suit different technologies, such as mobiles and tablets as user's are moving away from desktops to easily accessible devices. Thus creating a new challenge in making sure the application is fully responsive and all functions are functional across a variety of devices, including the maps functionally such as drawing on google map and viewing venues only within the drawn area. This means a number of algorithms or methods will have to be used to help make this function work. The last and important challenge is the ability to only display venues within the map matching the user's input, area drawn and location. Each criteria's data will have to be brought together to determine what venues should appear through the Foursquare API. Each of these challenges must obey any web standard guidelines, for example the web application must meet usability and accessibility guidelines.

1.2 Aims and objectives

1.2.1 Aims

The Let's Go web application aims to create a high level, accessibility and secure application with simple user interaction. The application will allow users to choose venues, based on their chosen location, day, cost and timeframe and then display multiple transport options and directions to the venue of their choice. Users will be able to save favourite venues within their own personal accounts, to quickly gain directions when logged back on.

1.2.2 Objectives

- Identify all of the user's requirements and constraints.
- Determine and choose an appropriate design methodology.
- Study and analysis new technologies such as PHP5/MySQL to determine the suitability for the web applications security and usability.
- Use suitable application interface programming technologies, such as Google Maps and foursquare, so users can find venues with correct details.
- Understand the intended audience by carrying out online research/survey's to understand the user's needs and how they would interact with the Let's Go web application.
- Discover any weaknesses within the application using different forms of test approaches to help improve the functionality and ease of use.
- Recognise principles and theory such as accessibility and W3 standard.
- Use different forms of design theory's to achieve a fully usable web structure for ease of use and responsiveness.
- Evaluate the application to verify the aims and objectives have been met.

1.3 Overview of work to be undertaken

Looking forward into the development of the project, several tasks will need to be undertaken to reach the aim of the Let's Go web application. Selecting the correct methodologies will help in planning the design process which will help to generalize a list of requirements, define the scopes and meet the aims and objectives. Thus helping to make sure the project stays ahead of time allowing for any sudden fall backs. Then each technologies will be selected and justified on why certain technologies are more suitable than others such as determining why a scripting language. The web application's database and user account will have to be secure with solid validation, avoiding any hacking or misuse of personal data, therefore taking into consideration the data design and system design such as entity relationships and architectural design. The main objective functionality must be fully functional and operational and must show correct results from the users input. The user experience design will cover the human computer interaction, design theory and usability, making sure users can fully understand the search functions, map drawing, storing venues and simple navigation system throughout the web application. Noting that responsive web design is vital to allow accessibility on all mobile platforms. The final stage testing will cover the different testing approaches, process and the results, evaluating if all requirements have been met and thinking of solutions to help fix any issues that do appear.

1.5 Overview of the rest of report

This report will cover the theory and process of Let's Go Web Application. Chapter 2 of the report covers the concept definition including the idea generalization, paper prototype stage, feasibility and methodology selection for planning out each task by splitting them into suitable categories. Once the concept of the project has been covered the next is to discuss and rationalize the designs. Chapter 3 covers the design theory including the web application systems design, user experience design, the logic behind the system and the database design. The User experience design will explain how and why certain design theories were used to help meet good design practices. Following into chapter 4, the implementation stage, will discuss the technologies and tools selection and usage, along with any notable changes and challenges throughout the project. Once the application has been built, Chapter 5 will cover the different test approaches, process and results making sure all of the requirements have been met. Chapter 6 will then take these results and evaluate if each result have met the user's needs. This section will also cover the evaluation of the project outcome and the chosen methodology stating if each method have reached the aims and objectives.

2. Concept definition and testing

This section will cover how the Let's Go application materialized, the requirements, paper prototyping, Feasibility testing and the Methodology selection to generalize how the application will be built, using requirements to meet the objectives and testing if the application can be built within the timeframe by selecting a suitable methodology design.

2.1 Project Idea

The Let's go web application is aimed to allow users locate venues by selecting a day of the week, start and end time, draw the location area, select a maximum spending and finally select a type of venue. After the search input the web application will generate a selection of shops, cafes, restaurants, bars and tourist attractions, this list will be comprised of suitable venues near to the location, while factoring in opening times and cost using Google Maps/Foursquare API's. Users can see details about each venues which also includes the option to gain directions with multiple options of travel to help reach their venue destination. Users will also have the option to save venues within their own account for quick access to saved venues and their directions.

2.1.1 Idea generation

Many people today will travel at point in their life, but what if they travelled to an unfamiliar locations and try to locate suitable venues within the area. For example, a user arrives at their destination but has to wait until the next day for their connecting flight. Instead of idling within the airport or the airport hotel, they can simply use the Let's Go web application, draw on current area or search an area, input the time they are available, draw an area and maximum spending. The current solution available is to search on google maps for venues, but users would have to select each individual venues to find out the opening times and cost. Users today want a fast, easy-to-navigate web app with key functionalities, with a native and engaging user interface (Compuware 2012).

The anticipated solution is to help users minimise their search duration and enable them to easily search for venues without having to continually look through all of the venues to check for price and opening times. The main user interface feature will be the hand drawn search feature draw similar to Property Pal (2014) ability to draw an area to search for housing properties. The site Weotta (2014) also gives a similar solution to the concept, allowing users to look up events, popular bars and restaurants in the area, but the site lacks the control of time and cost criteria compared the Let's Go web application.

2.2 Requirements specification

The requirements specification covers the purpose of the project and stating the goals of the project. After discovering the purpose discovering the stakeholder, assumptions, risks, scope, requirements and feasibility testing will help to determine who the application is intended for and if project can be completed within the given timeframe. The methodology selection will then analysis and determine which design is most suitable for the application.

2.2.1 Purpose of the Project

The Let's Go web application has potential since more people are relying on technology to solve problems, find locations and communicate globally, such as Google Maps and Foursquare. The application allows users to explore different venues locally and abroad, therefore will take on these location based technologies to help users search for venues fast and efficiently. An increase of 81% users are now using mobiles and tablets for location-based search while internet users have seen a fall of 10% in 2014 (Marketing Land 2014). The statistics shows an increase of users using mobile technologies for location based searches, such as applications and maps, therefore highlighting the importance of having a fully responsive web application.

2.2.2 Project goal

The overall objective of the application is to allow users to search up venues using their own set of criteria's, choose a venue and get directions to the venue. Therefore the main goal would to get the users from their current position to the selected venue without getting lost or be given inaccurate directions. The advantage of setting out the main goal is simply to help users get to a venue they've chosen, resulting in the expanding popularity of the application. Meaning more recognition for the let's go web application. It's vital to make sure the application reaches the market areas requirements and need to achieve this goal. Which can be done by simply determining the stakeholder, explained in the next section.

2.2.3 The stakeholders

Thinking about and defining the personalities of the stakeholder will help to define the web application and establish the ethics by providing valid decision and justify how the web application will meet the users requirements.

The application is aimed for a variety of ages as the application offers a range of activities, from bars to public parks. This means the application will be accessible, simple and easy to use for the younger and older audience.

To find more information about the customers and what they want, a survey was built and provided online to a number of random individuals. Out of 30 responses shown in appendix A.1, 57.14% were female and 42.86% were male with 66.67% being between the ages of 18 to 24. 13.33% of 25 to 34, 10.00% of 35 to 44 and 10.00% of 45 to 54. With the majority being around 18 to 24 it's clear to see the web application will benefit more for the younger generation. But one response from the 35 to 45 age range left a useful comment below.

"I have promised wains that we will do such a thing only to turn up when it nearly closed or shut"

This details how this application will be beneficial for trips with her children instead of appearing at venues last minute, giving a new view on how the web application will work by providing opening hours for each venues.

The web application will be aimed towards a variety of ages, leaning more towards 18 to 24 years old but still taking into account of the older generation. Overall the web application will benefit tourists, party goers and family day trips out by meeting the objectives and requirements.

2.2.4 Assumptions and consideration

Making assumptions will help to determine what will affect the web application and any boundaries that could be considered. Thinking about the location based web applications techniques some assumptions include:

- Searching the location should stick within the drawn area.
- Create an account to store saved venues securely.
- Account details will be protected under the data protection act.
- The directions should show correct route and suggested routes to chosen venue
- Transport available should be local within the area of the chosen venue.
- Google API and Foursquare API will display up to date information.
- Users will be able to use the application anywhere with data network available.

Considering each of the assumptions we can see how some can be organised within constraints or be listed as a requirement. The hand drawn search area will be required to stop excessive searching, therefore this will be classed as a requirement. The account which stores personal data and venues can also be classed as a requirement for the implementation stage in chapter 4.

2.2.5 Constraints

The protection of the data which will follow the Data protection law stated by the British Computing Association (BCS) must be accurate, adequate, kept up to date and secure. Therefore will be classed as a constraint, affecting the majority of the Let's Go web application. Within the implementation environment, the web application gives the ability for users to find and gain directions to venues anywhere, so therefore the application must work on any devices with small screen resolutions such as mobiles and tablets as users may not be home to access a desktop computer. Other constraints will also include keeping up to date information of transport and direction and also the availability to choose more than one route when getting directions, more constraints can be viewed within Appendix A.2.

2.2.6 Scope of the project

This will take into account the users requirements, the constraints and the goal of the project, determining the boundaries between the user and product, determining what sections are to be automated or inputted by the user. The red box within the use case diagram (Appendix A.4) represents the boundary with the Product Use Case (PUC) located inside the box and the user outside the box. The users will have an account within the systems database holding the account

details, Location API (google maps) and Information API (Foursquare API). The user's interaction will involve creating or updating their user account, input search criteria, select and save venues. The rest of the diagram shows an automated system, detailing the Location API (Google Maps) and Information API (Foursquare). The Google map will generate the location and directions list, while the Foursquare API generates the venue information, meeting the project goal.

From the use case diagram we can see the user will only have to interact with the web application a few times by:

- Creating or updating the user account
- Selecting or saving a venue
- Choosing the search criteria

Comparing to the user's age range from the survey results, the best solution was to keep the interaction simple and easy to use due to a wide range of age groups. The system will operate the Location API (Google Maps) and Information API (Foursquare) to generate the location and direction list, while Foursquare API generates venue information.

2.2.7 Requirements

The requirements section determines the build of the application and stating what the project has to do, making sure the aim of the project has been met. Taking into consideration of the objectives, aims, goals and the user survey helped to decipher the requirements by analysing each objectives and determining how to achieve this. Each requirements are then split by functional and non-functional requirements using snow card to describe, rationalise and determine a fit criteria.

Functional Requirements

Within the survey (Appendix A.1) the users were asked what features would they prefer to help search for their chosen venues. The first question "Within in the search criteria, would you prefer to search a location by:" the results came back that 60% of users would prefer to type the name of the location. While circling the map only coming in second at 30%. To help with more efficient use it was decided to add both typing the location and then circling the map for a more refined search, therefore meets the objective to allow user to search for venues.

Requirement	#9
Description	The search criteria must allow users to search and draw location on the integrated google maps.
Rationale	To allow users to search a location and then draw a circumference of the chosen area within the map.
Fit Criterion	The users should be allowed to draw an area on the map, which will then be added to the list of queries for the search button. For the time frame 57% of the users would prefer to select a start and end time, 63% want to use a slider feature for the price range and 50% would like to select multiple checkboxes for the type of venue.

Table 2.3.1 – Snow card showing search criteria requirements.

For the time frame 57% of the users would prefer to select a start and end time, 63% want to use a slider feature for the price range and 50% would like to select multiple checkboxes for the type of venue.

Requirement	#10
Description	User can select day, start and end time, max spending and type of location to search for their desired venue.
Rationale	Help users to define what venues they would like to see on the map using select boxes.
Fit Criterion	Select boxes used for day time and type of location, while max spending will feature a slider.

Table 2.3.2 – Snow cards showing selection requirements.

The requirement for the “venues” display was originally set to view the opening times, location, name and contact details, but within the comment section one response stated to include a website address as well, therefore more social media inputs will also be required such as twitter. Other functional requirements can be found within appendix A.4. Detailing each needed requirements for the Let’s Application taken notes from the survey, and noting this is what the user requires and not what the developer requires and making sure each aim and objectives have been met.

Non-Functional Requirements

The visual design of the web application will appeal to a wide variety of ages and will appear authoritative, unpretentious and enjoyable. The Let's Go application will require to have a smooth performance with adequate loading time between the application and server. The application shall be safe and secure meeting the data protection guidelines and objective, especially with users account details to avoid any hacking or corruption of data. Following a good methodology, ethics and professionalism will help to manage time and cost to make sure the product gets completed on time while meeting all of the requirements at a high quality of standard.

2.3 Paper prototyping

After analysing the requirements for the Let's Go web application the next step is to plan and design the application's structure, helping to meet usability objective. Paper prototyping is a beneficial as it allows for easy iteration, budgeting and gaining the initial feel of the sites structure. The paper prototype will help to establish the correct methods to use for human computer interaction and usability.

The first section shows the navigation structure within the web application, questioning if users can find the navigation easy to use. Figure 2.1 shows a simple navigation design of how each page will interact with each other. User will be able to gain access to any page from anywhere on the site with the index acting as the starting point. The account page will only be accessible when the user log's in/register for security purpose. Originally the application was to have a total of six pages (figure 2.2) but by merging the search page and direction page helped to reduce it to four page to help improve the usability and interaction. An online article by Strojny.D (2013) helps to back this theory by stating "Less is more" when it comes to navigation. Simpler navigation also means less confusion locating the desired content.

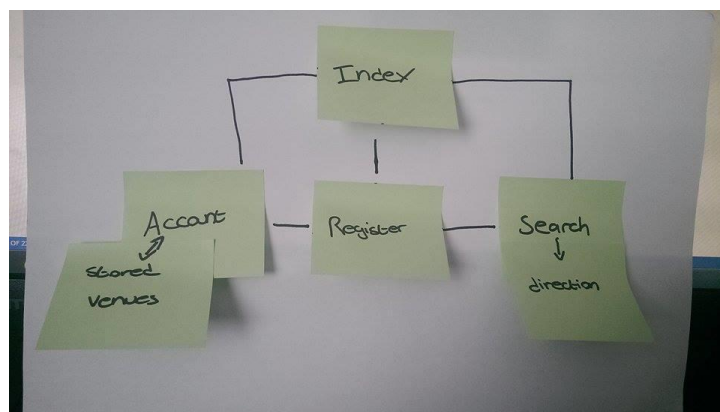


Figure 2.1 Initial navigation

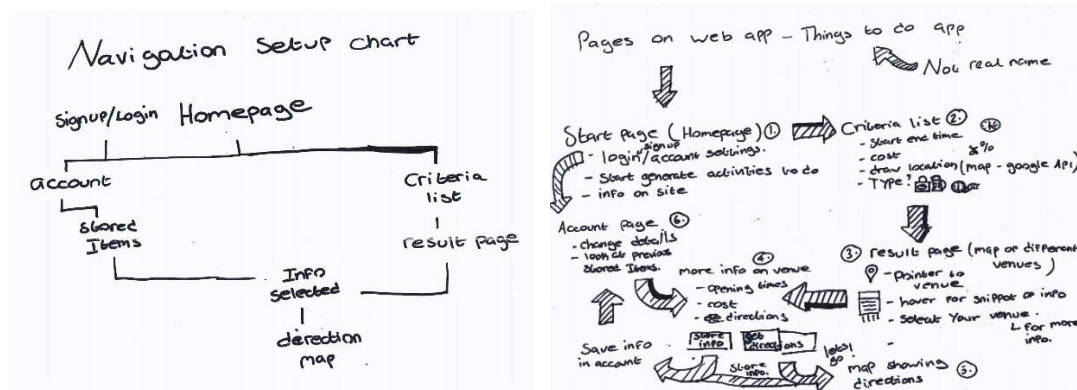


Figure 2.2 Mind Map of content

2.3.1 – 6ups

The 6ups helps to explore different versions of techniques and structures for the Let's Go Web application pages. Figure 2.3 shows six different layouts of the search page which highlights the preferred features which was then added to the final paper sketches. Within image five the heading section is the preferred feature while in image six the middle search criteria is the preferred layout option.

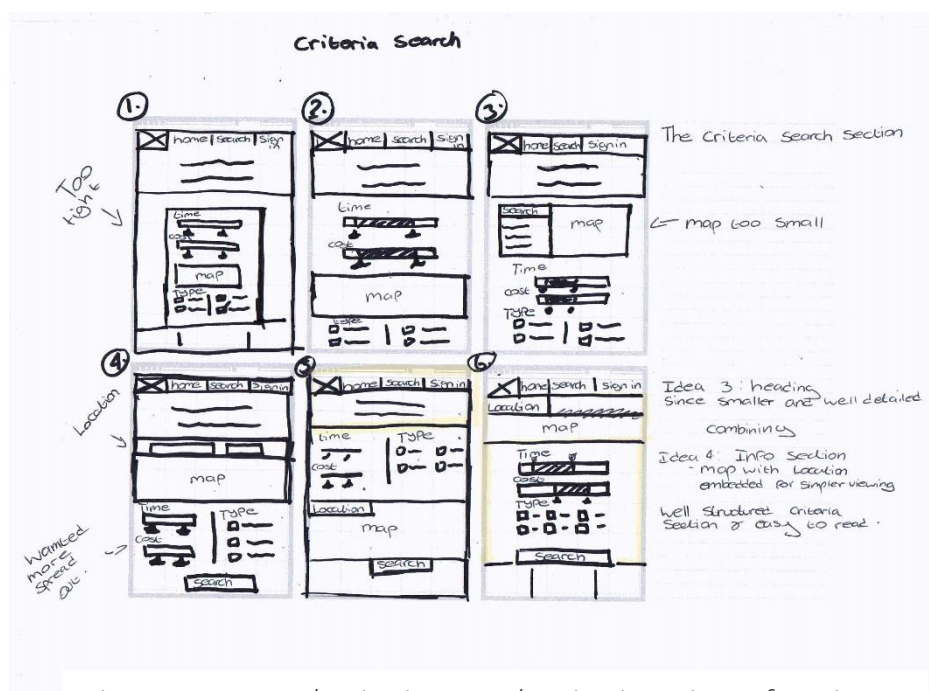


Figure 2.3 – Search criteria page showing iterations of versions.

The next step of the process is to build a mock up/wireframe on the user interface design matching which should match the requirements. The main functionality of the application is the search, storing and getting directions to the venues. From the homepage to the search page users will have the choice to search location or if there happy within the current area they simple move to draw an area onto the map. Each search will use select boxes apart from the maximum spending as it is easier for the users to slide to the maximum amount (Figure 2.3). The results are then shown on the same page within the map hiding any venues not matching the search options. The icons are clickable and will create a small pop up box with small details about the venue (Figure 2.4). A more info link will then generate larger pop up box containing information about the venue. From here users can then click to save the venue and get directions (Figure 2.5). The directions will still appear on the same page, so user can easily change venues and gain directions. The mode of transport table will appear beside the search area making sure the user can gain quick access to the search area and change their results, thus helping to lessen the number of clicks the user will have to do on the web application.

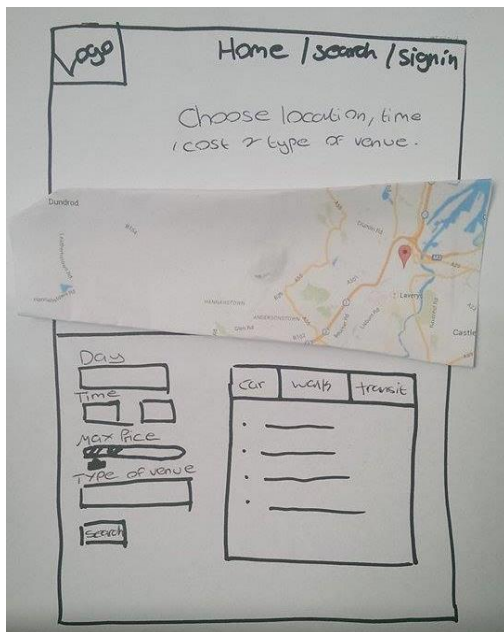


Figure 2.3. Beginning of search page

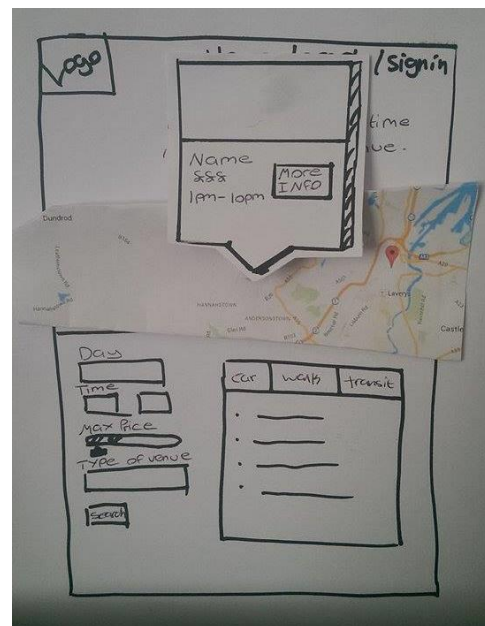


Figure 2.4. After search is complete and an icon is clicked will pop and info window



Figure 2.5 when more info is clicked another popup appears showing more venue details and option to save and get directions.

2.4 Feasibility testing

Reviewing over the user/system requirements and the purposed paper prototyping, the next step was to check if the site will be feasible before continuing onto to the design stage. The feasibility takes all of the requirements and constraints to work out if the project can be developed on time at a high quality. Time management is the key to complete the project on time, as the due date is already been set for the 1st of May 2015, giving a total of 33 weeks to complete the project. With the help of development libraries and previous training this target can be reached. But to help keep on track, the scope and cost will therefore need to stay the same to avoid changing the time constraint. The cost of the project will be low as most of the software, technologies and resources (internet, books, and development libraries) are free to use within the university. Since cost is not an issue the scope/quality can be increased to meet the quality standards. Denial of the scope will lead to problems later on in the project, consequences including failing to finish the project on time.

Another form of feasibility testing, to check if the project fits within the scope is by building a functional prototype. The functional prototype (Figure 2.6) took the highest requirements mentioned under section 2.2.7, which was the ability to allow users to draw, search and display venues, which was then developed into a live version to test and analysis if these requirements are possible. Completing the functional prototype also helped to build the main structure of the site and also confirmed the project will be feasible. Undertaking the functional prototype produced useful results on determining the most challenging section was going to be calculating the area within the drawn area, therefore classed as top priority within in the requirements.

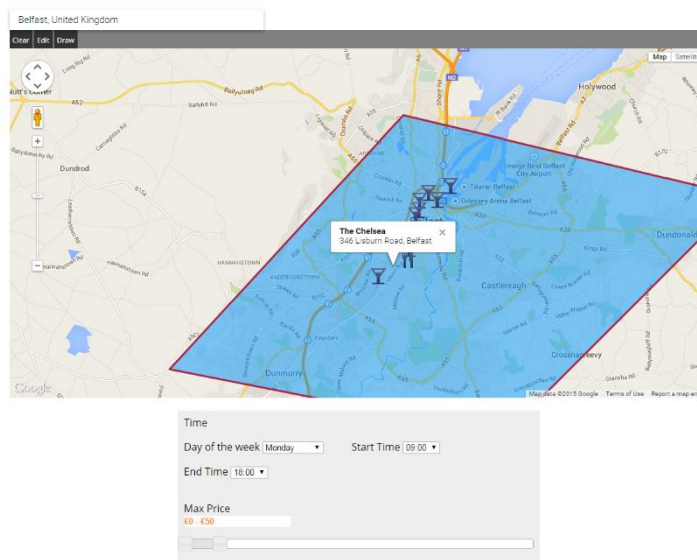


Figure 2.6 – Screenshot of the functional prototype.

2.5 Methodology selection

To help determine what needs to be done and how they should be done, a methodology designs needs to be chosen to help with the planning out the process of development for the Let's Go web application, covering time management, tasks, requirements and aims. Each form of methodology will have unique process on how solve process problems and schedule out tasks for the project.

The water fall methodology is the most common and traditional methodology used for projects. This model, also referred to as a linear-sequential life cycle, have fixed stages. Each stage must be completed before the next stage starts, therefore stages do not overlap. Thus making easy to manage and will work perfectly with smaller models. But the model will not allow any backtracks increasing the chance of risks appearing.

The agile method is known to be a new form of methodology and is most commonly used within large team environment. Agile allows for many methods such as SCRUM, which includes a product owner, scrum master and the team and concentrates on how to manage tasks within a team-based environment. Agile aims to replace all of the other methodologies by filling in their errors.

Rapid Application Development (RAD) aims to deliver projects on time at a high quality. Each stage are time boxed, delivered and then assembled into a working prototype. Clients can give any feedback about the requirements and delivery. This form reduces development time but a high skilled developer are required and bad teamwork can result in failure of the project.

The prototyping methodologies gathers all of the requirements and creates a prototype to gain the 'actual feel' of the system. But this could get completed for larger projects and scope is more difficult to handle. But errors can be detected easily and feedback is quick.

2.5.1 Risks and Challenges

Listing the risks and challenges will help to determine possible fall-back's therefore helping to keep in line with the scope. Each risks will be taken forth and considers within the choice of methodology. Loss of data can be a risk if not backed up as this will affect the timeframe and delay other tasks. Working with new technologies can hinder the progress of the web application as more research will be required. Unwanted bugs occur within the system will affect the quality of web application and again effect timeframe. To help avoid risks and challenges from hindering the project the best approach is too use a form of methodology.

2.5.2 Legal, social, ethical and professional issues

During the development it's also important to consider any ethical, legal, social and professional issues, such as the data protection act. This act aims to protect the user's personal details from any security issues such as hacks, misuse and deletion of data with consent. Therefore as noted before the user's details have to be secure or the act will be breached.

2.5.3 Chosen methodology

The methodology is a model used to help design, plan, implement and achieve the project objectives at high standard. Reviewing over the requirements, constraints, risks and duration, the most suitable methodology would be the Waterfall method. This method is manageable, straightforward and suitable for small projects, compared to the prototyping methodology due to being a lot more difficult to define a scope and extra repairs are required. Since this project is developed by one user a start-end dependences will organise each task in order not overlap, as overlapping could disrupt time and increase errors.

The Rapid application development method is useful for quick development and to gain feedback from customers, but its known high skilled developers are required to develop this methodology. Meaning it would be too difficult to use as more time will be spent on building the methodology. Agile is known to be one of the most popular forms of methodology such as SCRUMS, but would be more suitable within a team-based environment. This project is developed by one member therefore would defeat the purpose.

2.5.4 Work plan and time management

Taking the waterfall method, sub-tasking each of the task helps to organise what needs to be done so the project can be completed within in the timeframe. The task can be categorised as:

- **Research** – research the client's expected view of the concept, note the aims, objectives and requirements, know about the technologies being used, gather materials and check out similar competition.
- **Design** – determine the layouts, website structure, wireframes, content fonts, colour scheme, logo design and any images which need to be edited in Photoshop.
- **Development** – back end development starts, setting up the structure using HTML5 and CSS3, UI design using JQuery and Google API's, with the server-side build using PHP and MSQl for a database.

- **Testing/launch** – testing will include testing the UI interface, functionality especially for location search and making sure the data result is being represented correctly from the search criteria.

With the duration being 33 weeks in total each task will need to be schedule correctly to fit the tight timeframe. Therefore each task will be in the form of a finish to start dependence meaning the task can't begin until the previous task is completed thus helping to avoid any bottlenecks.

The major constraint will be time as set deadlines have already been announced and can't be changed. Following well defined requirements will help track and manage the scope. The project will only be worked on by one person, so having start/finish dependencies is suitable within the Gantt chart will help to organise each of the tasks shown in table 2.5.1.

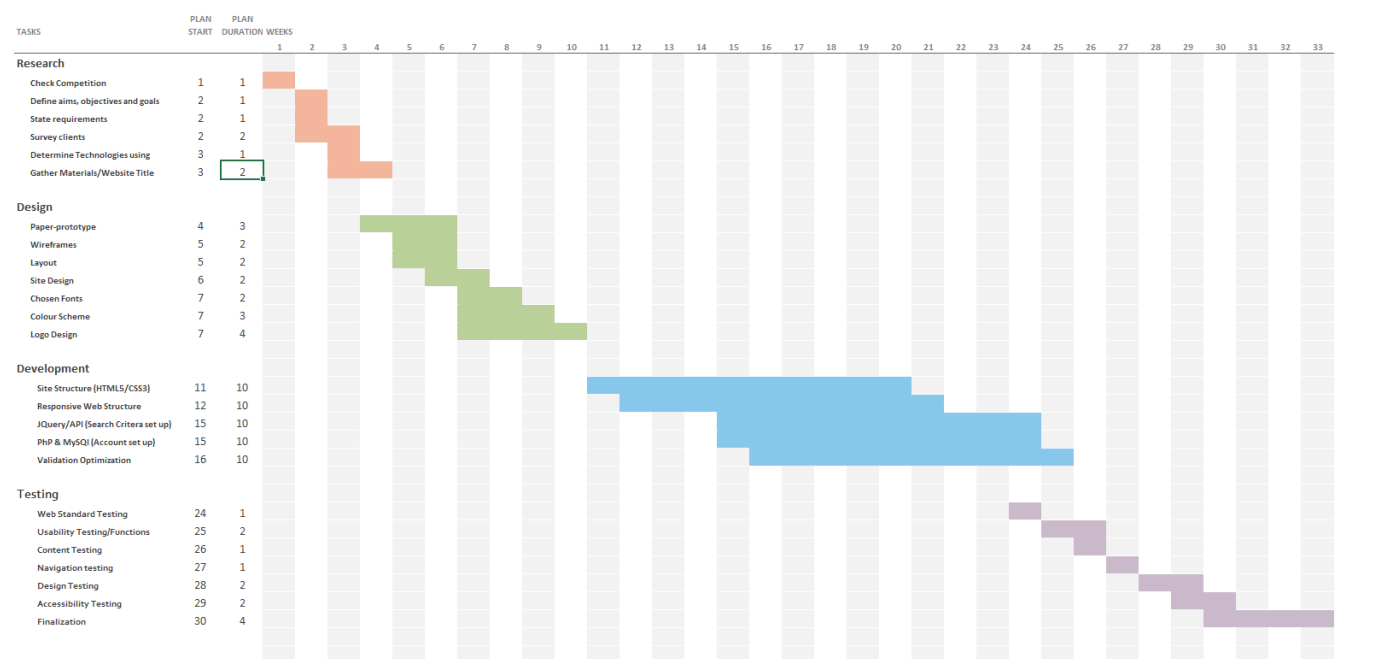


Table 2.5.1 – Gantt chart showing planned out schedule.

3. Design

With the concept, requirements methodology checked and verified, the next step of the process is the design stage. This area will cover the user experience, system, logic and data design of the Let's Go web application.

3.1 User Experience design evolution

The user experience design covers the ease of use and the usability of the Let's Go web application. Designing the web application to suit the user is vital as it helps to make their journey through the application more enjoyable and less frustrating. The UX design will take specific users problems with the design and create many approaches to help fix issues (Lo min ming 2014).

Before starting the design stage a few design notes were vital to know about creating the user experience design. The UX design should always put the user first, considering the requirements since they will be the operators of the web application. To craft the UX design will need to look at the colour, accessibility, human factors, human computer interaction, content and usability.

3.1.1 Colour scheme

Picking out the perfect colour scheme is vital to get correct, as colour can easily manipulate the user's emotion towards the Let's Go web application. For example blue can mean calm, safety and peace while red could indicate danger, love and power. Colour also has to match the aim of site, wither it be serious or a pleasurable experience.

The Let's Go web application will be used for travel purposes and searching activities, giving a fun, adventurous, friendly and trustworthy experience. Running through multiple colour schemes ideas for the web applications the most suitable colour palette would be bright but soft colours (Figure 3.1).

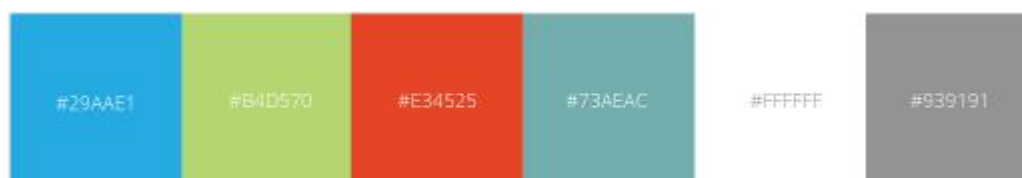


Figure 3.1 – Colour palette for Let's Go Web Application

The blue colour will be the base colour while the green will incorporate with blue to help achieve a more friendly and trustworthy feature. The green colour will act as successful while a red colour will display as a warning (Appendix A.4).

3.1.2 Simplicity Structure

The let's go web application structure focuses mainly on location based functionality such as google map and the geolocation. This means the map interaction will be the main feature for the web application and will be centred on this area. Thinking about the whole layout of the site it was decided to stick with a vertical structure by splitting each content into section vertically down the page as this will help with readability and users can follow the structure starting from the top working to the bottom (Figure 3.1.2).

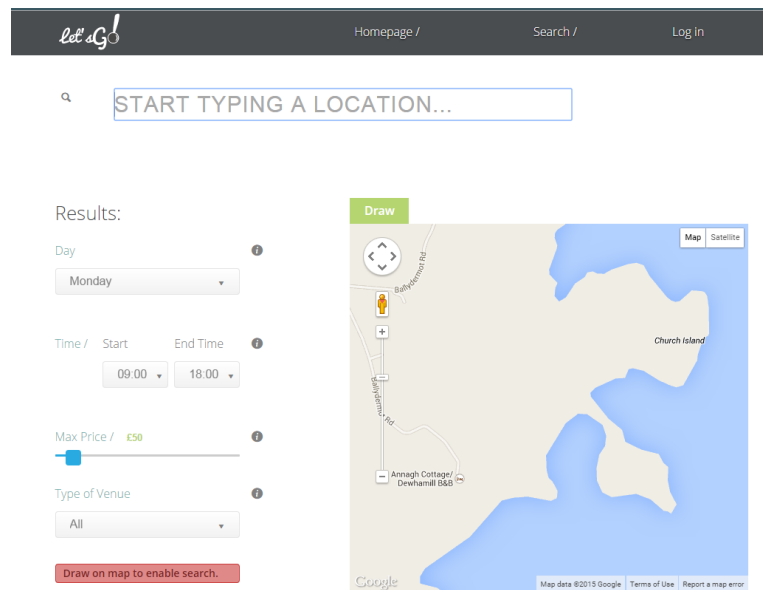
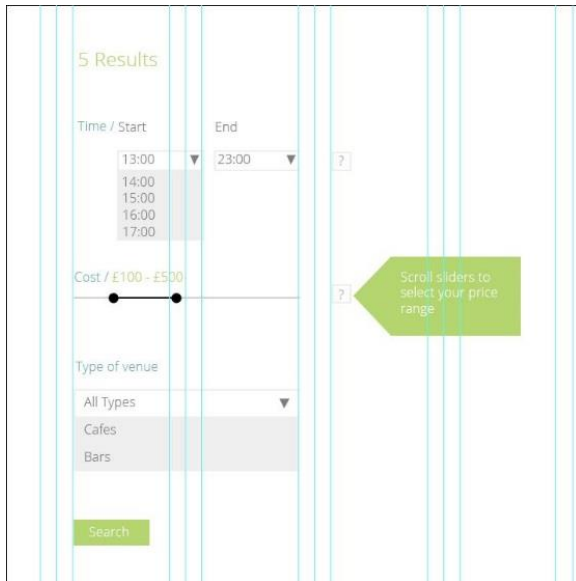


Figure 3.1.2 – UX design map repositions.

But within the search page a column like structure will be used, locating the search form and map beside each on other so user can easily edit the search form while interacting with the map. This was replaced from having the map across the top of the page as users weren't able to scroll up and down the page easily, also users could not see the search form while viewing the map. Adding more scrolls interaction, which could lead to irritation.

3.1.3 Scannable Content

Having a vertical structure means the content can be easily scanned and digested when the user is traveling down the page. Content should be limited and not lengthy but this does not mean content should be classed as unimportant (Zoltán Gócza 2014). The content for each page will be kept simple, easy to read but full of information. This will be vital for the search page as the users will need to be able to understand how to use the search features to help locate their venues.



Just below the map will be the search criteria (Figure 3.3) where the user will be able to define what venues they would like to see. Each of the options will be a select box apart from the cost as it will be easier to select a price range by simply moving the counter.

Figure 3.3 – The search criteria section showing each criteria set up and help notification

The correct venues will appear on the map and each marker will display an icon matching to the type of venues they are helping to easily distinguish between each venue. When clicked this then displays a snippet of each venue's info, and if the user wants more info they can simply click more info. Within the venue information two buttons will stand out to the users indicating they can get directions to the venue, store the venue within their account or they can do both.

The two pop up boxes will be a simple layout with a main image appearing for the mini pop up window (figure 3.4) and an image slideshow for the more info pop up window (figure 3.5).

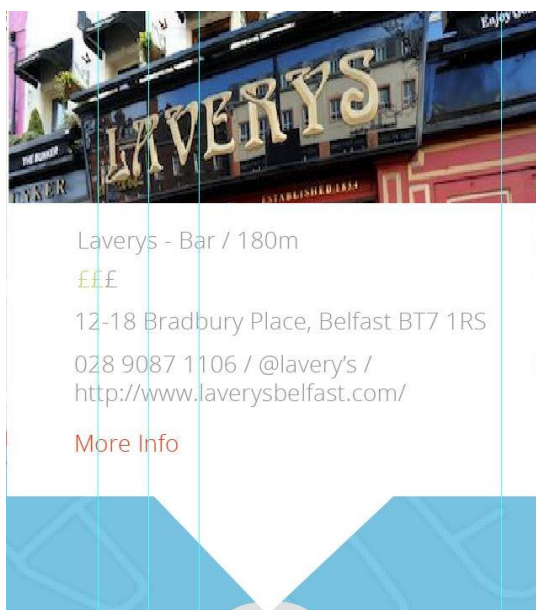


Figure 3.4 – Small pop-up window

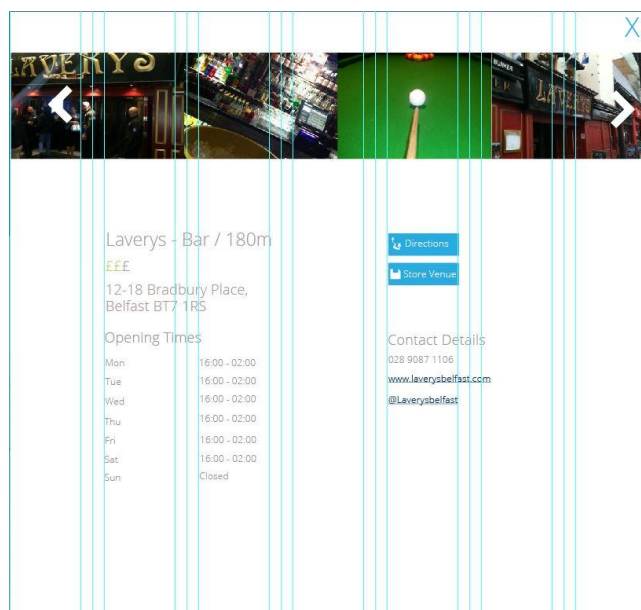


Figure 3.5 – After more info pop up window

The save venue button when clicked will simply indicate if the user has already saved, they need to log in or have successfully saved within the users database account. The directions button will simply keep the users on the same page but the map will now show the directions to venue and the directions panel will appear. The direction panel will be a tab format showing several mode of transports. On each change the direction information will change along with the maps direction.

Overall the web application has been developed into a simple, minimal and user friendly site. The colours used matches the fun and energetic use of the app.

3.2 System design

The system design of the Let's Go web application will be able to identify, compare, evaluate and specify different technologies being used within the application. This will cover the client server model, the technologies chosen, the model view controller, logic designs, and database designs. Analysing the Let's Go system design will help to select the correct technologies to use and justify how the whole system will operate comparing to the aims and objectives.

3.2.1 Client-Server Model

The client-server model identifies various technologies that can be used for the web application showing a number of perspectives. The model is split into three sections, the client, the internet and the server.

The Client-Side

The interaction between the user and computer allowing users to interact with the web application's functionality and features. For example in the web application the user will be able to hand draw an area using the Google map API within the search criteria section requirement (Motive Glossary 2014).

The Internet

The interaction between the client and server using protocols to transmit instructions between the client and the server.

The server

The processing centre of the web application which is not visible to the users. The script-side server grabs information sent from the user, processes it and then outputs the results to the client. For example, the requirement of displaying venues from the users unique search criteria will send the data to the server, the server will process the data and then return venues matching the criteria.

Figure 3.6 shows the different technologies that could be used to develop the web application and a system level perspectives. It's important to work out and distribute the task and work load between the clients and sever to help work the communication. In this case the internet will be the main communication device between the client and server. The internet will be the browser the client uses to interact with the web application since the web application is to be hosted online.

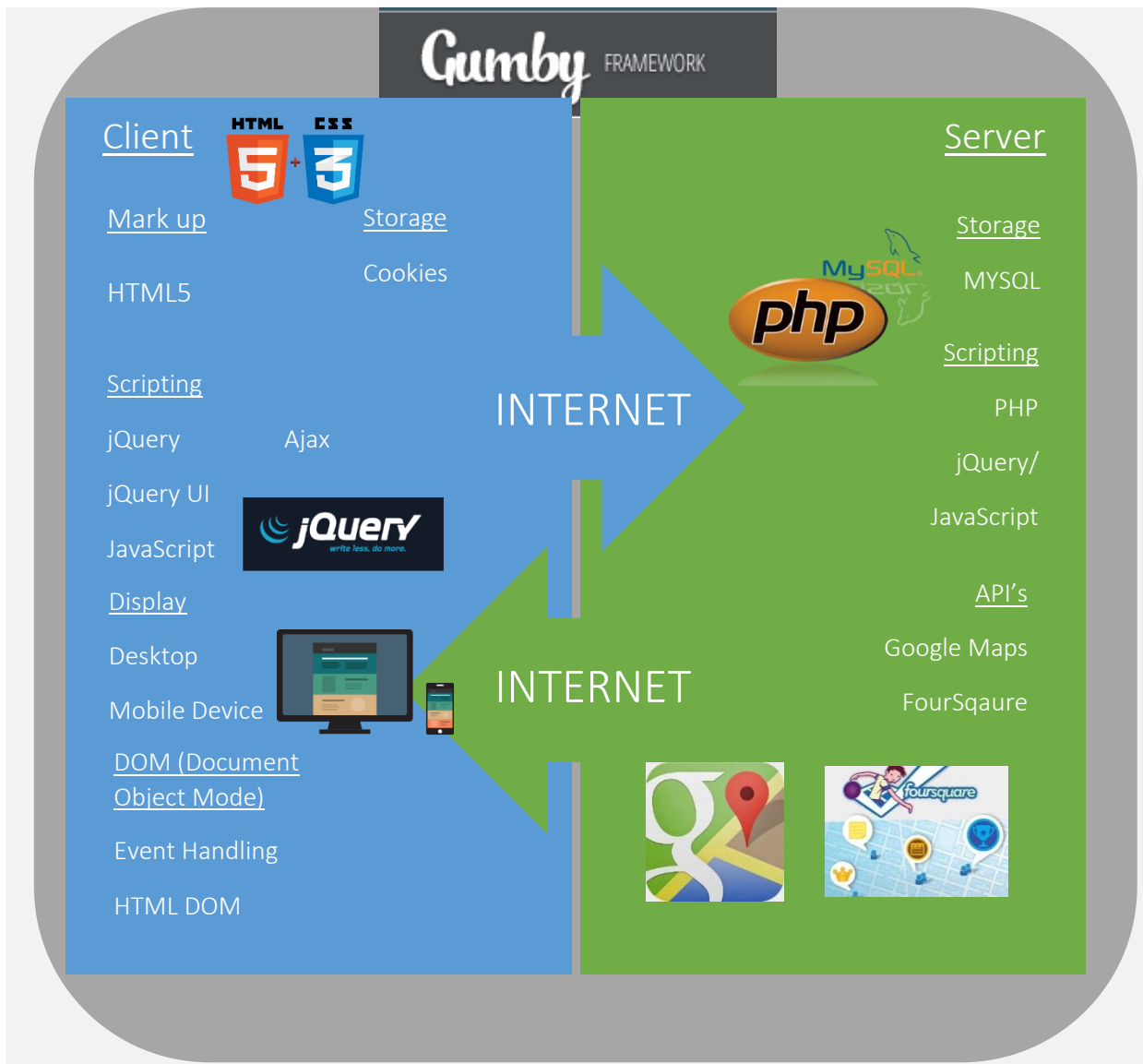


Figure 3.6 – Client-server Modell

3.2.2 - MVC Diagram (Model, View, Controller)

The Model, View, Controller shows detail on the process of data, queries and interactions. Taking the main objective, the user will interact with the controller by inputting a search query which then passes from the controller to the model. The model will then compare the queries to the database and return. Once the controller receives the results, this then passes to the view to be displayed to the user (Appendix A.6).

3.2.2 Web System Architecture

In figure 3.7 the users will interact with the application using devices such as keyboards, mouse and touch screen for mobile purposes. The users will then input an action which will interact with any of the event handlers found within the document object model. For example if the user searches for the directions to their chosen place, this will then activate google maps API, run through any scripting and then is outputted onto a device. External storage will be used for users account and saved venues. The external storage will also allow users to save directions offline, allowing users to view direction to venues while offline.

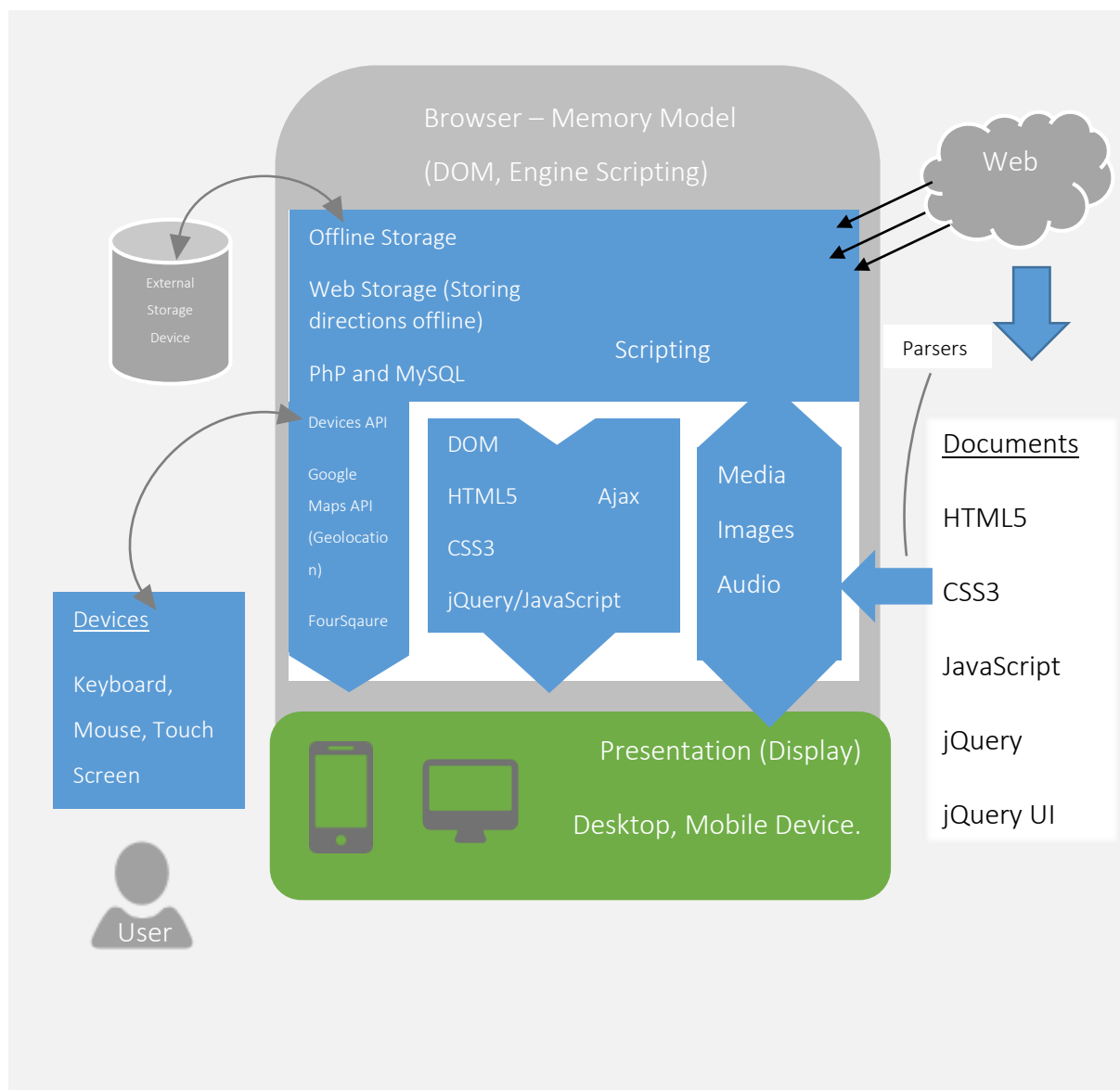


Figure 3.7 – The let's go web application structure

3.2.3 Chosen Languages, frameworks and API's

Table 3.7 shows, describes and justifies each of the different technologies being used for the Let's Go Web application.

Technologies	Description	Justification of Use
Mark up & Scripting Languages		
HTML5	Latest mark-up language used to create a website.	These will be required to build the web application. Latest version and no other mark-up options except for older version.
CSS3	A cascading stylesheet used to style a website.	
jQuery & JavaScript	Used on websites to enable interface interactions. The library simplifies animations, event handling and any other form of interaction.	A definite usage for all interactions within the web application. For example disable the drawing tool when shape has been drawn on map, unless edit is clicked.
PHP5 & MySQL	The scripting language can process any data and collect them all into a database.	Processes the users account details and stored venues.
Sass (Syntactically Awesome Style Sheets)	A powerful, mature and stable extension language. The language allows for extra features that don't exist within the CSS language yet for faster development. The preprocessors can be used with multiple frameworks.	This will benefit throughout my development, since less time will be used in styling as the language allows the user to style sites faster. Using new features such as nesting and variables.

Front-end responsive HTML5 Framework		
Gumby 2.6	The frameworks offers a clean and fully reasonable web site and uses Sass to quickly customize and build your personal website.	Using this framework for the web application will be useful as the framework is fully responsive and allows for quick styling. The Let's Go web applications needs to work on both desktop and mobile devices, so full flexibility will be required.
Compass	An authoring framework for cleaner mark-up with presentation class. Full of reusable design patterns and makes easy to use.	Useful for fast and flexible styling to help meet tasks deadlines within the gnat chart.
Libraries		
jQuery UI	Allows for more interaction features and animation.	Used to build the slider feature within the search form.
Slick	A fully responsive carousel image slider plug in.	This plugin offers a responsive slider suitable to display each venues image gallery within the large pop up window.
Strength.js	A password strength checker used to indicate the inputted password strength level.	This plugin offers an additional feature to check for password strength. Useful for keeping a secure user account.

API's (Application Programming Interface)		
Google Maps API	Combined with many API option such as direction API, geolocation and Places. Google Maps API offers many map interactions and functions using JavaScript.	The main API to be used for the web application, as shown in the functional prototype.
FourSqaure	The fourSqaure API offers all venues information such opening times, names and location.	The foursquare API will be used detail the venues information such as the API offers more information about the venues including reviews.

Table 3.7 – Table showing the chosen technology's and languages.

The previously chosen framework was Twitter Bootstrap, but after more research and experimentation, the Gumby 2.6 came on top for responsive web design. After experimenting with the two frameworks on ease of use, Gumby 2.6 turned out to be a lot easier to use compared to bootstraps strict styling. Both frameworks use the preprocessors Sass for clean and quick styling.

For the database and server-side system will be PHP5 & MySQL. This server-side scripting language are very powerful when used together and stores/generates any data. Which will be needed to generate the search results for the chosen venues. Other options included was Ruby, a new scripting language which offers simplicity and rapid development. But this meant more time was needed to learn the code therefore taking up too much time, similar to why ASP.NET wasn't chosen. But ASP.NET required a .NET framework so additional set ups were needed.

While creating the functional Prototype map section, it turned out that the Google Maps API has a feature which displays each of the venues information. This meant that fourSqaure API could be excluded. Patil, A (2013) compares both API's by Rate Limits, Cost, Coverage and Accuracy. It mentions the Google Rate limit is 1,000 request/day while FourSqaure allows 5,000. Although FourSqaure offer more requests, Google offers a wider coverage due to a strong marketing background. FourSqaure will only have coverage penetrated areas since the app is crowd-sourced.

After reviewing it felt best to keep both API's as one offers many map features, while the other offers more detailed information.

3.3 Logic design

The logic design section will talk about the logic behind the structure of the site, such as discussing about using different methods such as reusable codes design patterns and PHP includes.

3.3.1 Design Patterns

Coming up with suitable design patterns is very beneficial to keep design consistent throughout the website. Using design patterns will also help to increase development time and reduces the chances of errors. The Gumby framework offers many design patterns in aiding the development, such as a grid system, user interface kit and components.

The grid patterns used for the web application will be a basic 12 column grid using row classes to assign the page's max-width. Each of the columns won't have a fixed width but instead vary based on screen resolution to allow the let's go web application to adapt to any devices and viewport. The same design will be the same layout throughout to help with flexibility.

There are many options within the user interface kit such as buttons, forms and tables offering metro and semi-flat styles. Forms have always been tricky to build but using the frameworks form pattern will offer simple input area adjustments, validation designs and even conjoined inputs.

This will be the navigation located at the top and the footer of the Let's Go web application. In the desktop view the framework's user interface kit offers a simple navigation pattern. The navigation will be a simple fixed inline structure with current site highlighted. Breadcrumbs will also be included for usability purposes. For mobile view the navigation bar will change into a burger menu for viewing purposes, as the original navigation bar would be too cluttered and tight.

3.3.2 PHP Includes

Thinking about the whole structure of the web application, there are certain elements/patterns repeated. To help reduce errors and save time, using scripting language such as PHP include function will automatically get the style and set within the chosen web page. For example the header and footer will be the same throughout the web application, therefore creating the header and footer style within separate files will allow script to grab the style and include within webpages contain the include method. This means the header and footer only needs to be edited once, instead of multiple times on separate webpages.

3.4 Database design

The database design focuses on the development of the system database which includes the user accounts and get directions table. This section will discuss the entity relationships, table structures of primary's keys and foreign keys and the relationship between the tables.

3.4.1 Entity relationship diagram

Figure 3.7 shows the entity relation diagram of the let's go web application database structure detailing the relationship between each tables. The database uses the venues as its main base of operation with most of the relationship between the tables having a one to many relation. Each user will have many venue saved and venues will have one user. Finally the direction table can have many venues, while venues can only have one directions co-ordinates. Originally the user's details were kept within a separate table but this felt pointless since only one relationship was found between the user and venue, helping to avoid overcrowding the database. Adding an extra table to keep directions to the system will allow users to save the venue and store the co-ordinate within the directions table, so they can easily get directions from their stored venues.

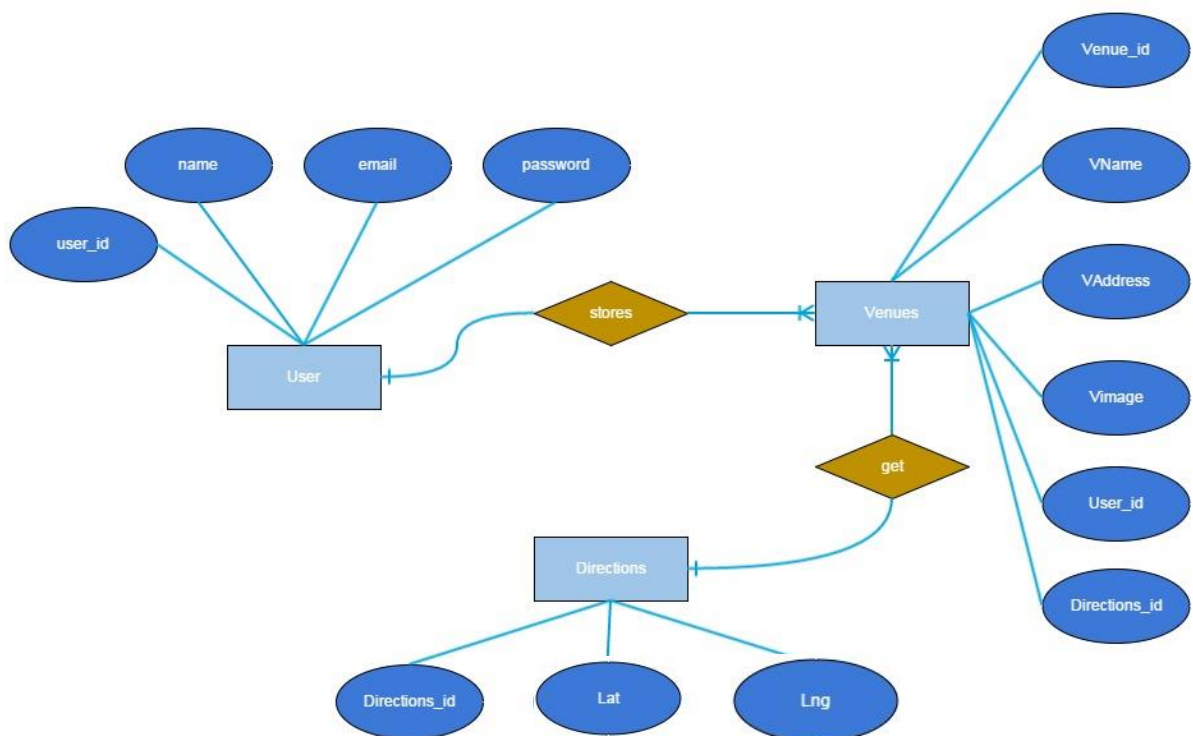


Figure 3.7 – Entity relationship diagram for the Let's Go web application

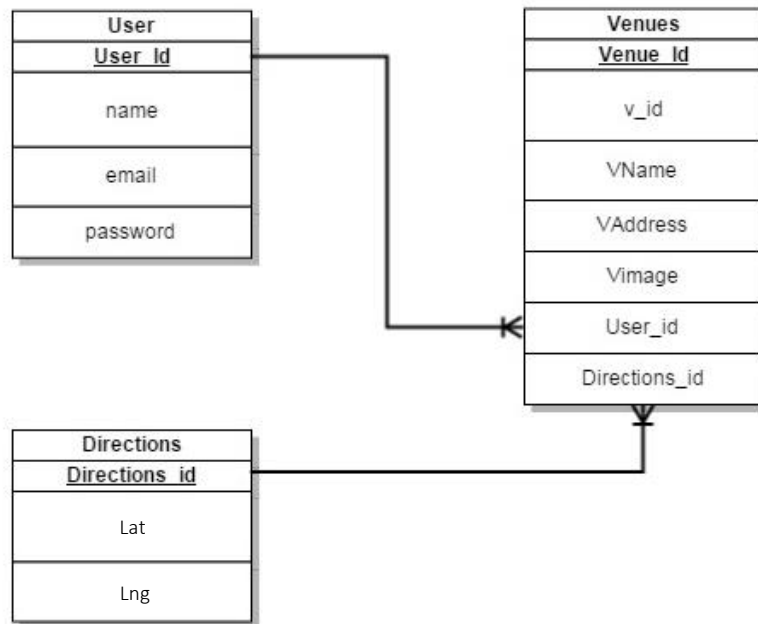


Figure 3.8 – Entity relationship diagram for the Let's Go web application

After pinpointing the relationship (Figure 3.8) between each table next is looking into the data structure of each table (Appendix A.7). The user account table will consist of an id, name, email, password and account created sections. The id will act as the primary key and set to auto increment, meaning each user will receive a unique id. Name, email and password all are set as varchar since the email and password can have a mixture of letters and numbers. The created column uses a timestamp to record when the users created their account, so user can check when they've created their accounts. The data located within the venue table contains the Venue_id, v_id, Vname, vimage, user_id and direction_id. The user_id and direction_id is classed as a foreign key matching with the user accounts primary key and directions matching the directions table primary key. This means each venue listed will be matched up with the user's account and correct coordinates when saved.

4. Implementation

Once the system, user experience and logic design have been justified, the next step is to the implementation stage. The implementation stages discusses and validates what technology/tools were used to help implement the Let's Go web application, the notable challenges and any notable achievements.

4.1 Technology/tool selection

It's important to choose the right technologies/tools for support in building the Let's Go web application as it helps with fast editing, effective functionality and updating. Web development have many elements and features that can be included within a web application, such as responsive development, dynamic functions, WebGL, audio and many more.

Gumby 2.6 is a responsive framework used to help develop a fully responsive website. The framework not only covers a responsive design but also include user interface kits, components, plugin extensions and JavaScript options. Compared to other popular frameworks, the Gumby 2.6 is much lighter and easier to install using tools such as SASS and Compass, therefore will fit in with the current timeframe available.

Another handy tool for live testing is Xampp. An open source cross-platform web server using scripts such as PHP5, MySQL, Apache and many more. This server will be beneficial to view any interactions between the client and server, for example when the user stores the venues within their account. One of the best benefits is the ease of installation, as the server can be installed all at once without having to install separate components.

A lot happens when building a web applications, therefore it's important to be organized and productive during the development stage. There are many tools that can help in organising tasks, such as Trello. A fully flexible organizational tool which uses lists filled with moveable cards to help keep track on what's being developed or what's to be tested. For the Let's Go web application the lists will be laid out with a to-do list, in development list, to be tested list, issues and a completed list to help keep track on each of the different functionality's.

The Let's Go Web application will not be developed within the same environment such as computers and laptops. Therefore being able to access the file online is important so a file sharing tool will be needed to give the ability to upload and access these files. A popular online file sharing tool is Dropbox which allows users to upload and download files online. But Dropbox does not

handle version control while another open source called git does. Git is a version control system which allows for fast and efficient handling small and large project. But this system requires learning different commands and installation taking up a lot of time, therefore Dropbox was used instead.

4.2 Technology/tool use

To text editor being used to help me through the development of the let's go application is known as Brackets. The text editor offers many features which helps to make it's easy to design within the browser. But the reason for this choice of browser is due to its simple and organisable layout, allowing me to split the text editor screen and updates the browser automatically when a piece of code has been changed.

Within the Let's Go web application an image slider is located within the account and the more info modal. This slider uses a jQuery plugin called slick.js and offers many functionalities to suit the project, for example a responsive carousel effect.

The database for transferring data will use PHP5/MySQL has it offers of many functionality including a md5() function which hashes a string of data. This means it can be used to display the password within the database as random row of letters. Thus helping to meet the security level for the Let's Go web application and fulfilling one of the objectives.

4.3 Notable challenges

During the development of the Let's Go Web application a few notable challenges were encountered, some more difficult than others. These challenges have offered an insight to learning new technical skills and logical thinking as some functions requires thinking and problem solving skills.

4.3.1 – Drawing freehand the search radius

On the Let's Go web application, one of the requirements gives the users the ability to search a location and then hand draw an area on the map using the drawing tool and venues will only appear within this area. The first challenge of the functionality was to give users the ability to draw an area on the map. Originally at the start of the project users had to draw a polygon by point and click with a minimum of three points to complete the selected area. But this meant users had to click within the map to complete the area which can be tedious especially when used on mobiles.

Changing from a point to click exercise to a freehand drawing tool will allow the users to have more freedom to draw on the map and not feel restricted.

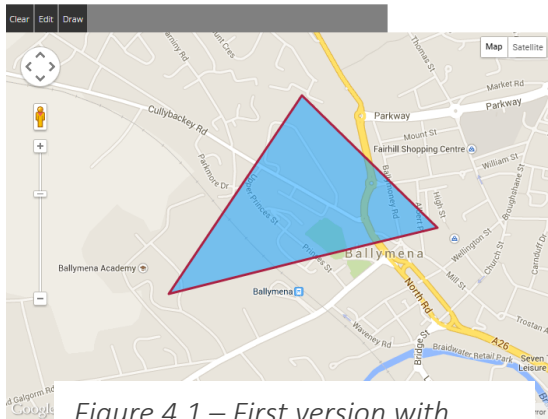


Figure 4.1 – First version with point and click exercise.

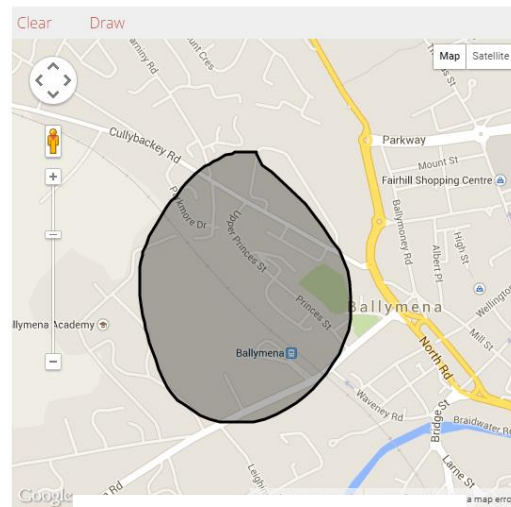


Figure 4.2 – second version with freehand drawing

Although, Google maps API do not clearly state on how to enable freehand drawing, therefore creating more complications. But the API does offer the use of Polyline which can be used to build a polygon. The Polyline is an overlay class that creates a line using segments which is then implemented onto the map. When the user clicks the link labelled 'draw', a preventDefault event stops the window from loading so the JavaScript function can run. To prevent the user using the maps controls the disable function is called first on click to disable the controls by setting the controls to false. Once the user starts to draw the mousedown listener activates the drawFreeHand function, activating the polyline class. Another listener then listens for mouse movement and gets the polylines path, pushing the latitude and longitude to the e parameter. When the user has completed or lifts the mouse button, the mouse up listener will listen once and remove the mouse move listener, grabs the latitude and longitude from the drawn polyline and retrieves the path. The polygon class is then called using the polylines coordinates to complete a polygon shape on the map. The enable function then reactivates the control options so users can interact with the map again.

4.3.2 Calculating the search area

After the search area has been drawn the next challenge was to make sure the venues are only displayed within the drawn area. The venues are generated from foursquare API therefore will have to follow their documentation on how to search for certain types of venues. The API use's parameters to help refine the search with latitude and longitude classes as required. The latitude and longitude acts as the centre point of the API's search, therefore needs to know the location centre point of the freehand drawn polygon.

The Polygon is passed through the search function and an instance called LatLnBounds is set to a bounds variable. The instance creates a rectangle from its south west and north east corners. The polygons coordinates are selected and passed through a JavaScript for loop and each of the latitudes and longitudes iterated is returned to bounds which extends the rectangle area (figure 4.3). So far the area of the polygon has been calculated but the next stage is to retrieve the centroid using a method called getCenter(). This method calculates the area and returns the central coordination from the drawn polygon. But the foursquare api specifies that the latitude and longitude must be inputted into the search query therefore the central coordinates has to be split using the lat() and lng() methods (figure 4.4).

After discovering the search areas centre point, the next challenge was to make sure the venue stayed within the drawn area. Just as for getting the centroid getting the area will also have to follow the api's documentation. Within the documentation there are two parameters which limits the results within a certain area, called radius and south west/north east parameters. Using the radius parameter was the first option as it limits the results within the area, but this led to issues. With the radius parameter set the search results were not staying within the drawn area indicating that the parameter was not using the polygons correct area calculation of the drawn polygon. Looking into the issue it turns out that the area algorithm was returning the value in meters per square, but the radius parameter required meters. But even if the results did return the correct value, the algorithm still would not able to detect the polygons lines and coordinates.

The sw and ne parameters on the other hand do take the latitude and longitude of the bounding polygon by taking the south western and north eastern corners coordinates. Since each of polygons coordinates are collected already using the getAt() method, the getNorthEast() and getSouthWest() methods grabs the correct values, then splits into latitude and longitude as the parameters require the coordinates to be separated.


```

var bounds = new google.maps.LatLngBounds(); //CREATES THE BOUNDS
var paths = polygon.getPath(); //FREEHAND DRAWN COORDINATES ARE COLLECTED
var path;
//FOR LOOP TO ITERATE THROUGH THE POLYGON PATH
for (var p = 0; p < paths.getLength(); p++)
{
    path = paths.getAt(p); //GET THE COORDINATES
    var sw = path.lat(); //SET EACH LATITUDE INTO VARIABLE
    var ne = path.lng(); //SET EACH LONGITUDE INTO VARIABLE
    bounds.extend(path); //EXTEND THE BOUNDS TO MATCH DRAWN POLYGON
}
var polyCenter = bounds.getCenter(); //GET THE CENTROID OF THE POLYGON
var ne = bounds.getNorthEast(); //GET THE NORTH EAST COORDINATES OF THE POLYGON
var sw = bounds.getSouthWest(); //GET THE SOUTH WEST COORDINATES OF THE POLYGON

```

Figure 4.3 – Code showing the methods used to grab the area.

```

//FOURSQUARE SEARCH QUERY
var CLIENT_ID = 'SZIHVFHTJSL5ZSZEJLY8VFBPSR4NS4WUDCZGKKNQWTSXUUBC';
var CLIENT_SECRET = '1mL4t1MPCALY05WLC8SQKH2HAKD3RI1X2IwE13ZBY3QQQYX';
var lat = polyCenter.lat();
var lng = polyCenter.lng();
var swLat = sw.lat().toFixed(5);
var swLng = sw.lng().toFixed(5);
var neLat = ne.lat().toFixed(5);
var neLng = ne.lng().toFixed(5);
var type = $('#category option:selected').val();
var start_val = $('#select[name=starttime]').val();
var end_val = $('#select[name=endtime]').val();
var day = $('#select[name=day_week]').val();
var API_ENDPOINT = 'https://api.foursquare.com/v2/venues/search' +
    '?client_id=' + CLIENT_ID +
    '&client_secret=' + CLIENT_SECRET +
    '&v=20140806' +
    '&ll=' + lat + ',' + lng +
    '&intent=browse' +
    '&ne=' + neLat + ',' + neLng +
    '&sw=' + swLat + ',' + swLng +
    '&categoryId=' + type +

```

Figure 4.4 – The api search query setting the latitude and longitude.

The most challenging part of free hand drawing was trying to keep the search results within the search area due to lack of documentations and the constraints of Four square's parameters. But working through the references available, it became clear to see how different bounding class can be used to simply restrict the results using coordinates.

4.3.3 – Building the search query

After the maps functionality were all working correctly and venues were showing, the next challenging step was to now incorporate each of the search queries, making sure only venues that match the criteria should be shown on the map. The search query is built within a form with days, time range and type situated within a select option, while the price section uses a slider feature for easier usability. The search button located at the end of the form will then activate the search functions within the search_map.js file.

The Day select bar

The day bar simply includes each of the days of the week as values, allowing users to check if a venue is open on the selected day of the week. The values are set using numeric values to represent each of the days of the week, for example 1 = Monday. Each of the numeric values had to match to the days of the week since the API sets each day of the week into an array as numbers with Monday equalling 1 and Sunday equalling 7. When a user selects Monday the value '1' is then grabbed and set into day_val variable. But the search venues request does not offer a parameter to define the hours the user wants to search. So the solution was to create a separate function which will loop through each venues and select the correct day.

When the user selects the search button a function is called to search for the venues using the calculated area and category and receive a JSON response. But this hasn't checked if the venues are opened on the selected day. Within the ajax success function (getSearch Results) each of the markers are displayed on the map, but then the function GetPriceRange() passes the json response and markers, then calls the getHours() functions with the venues and markers to request the venues hours response. The venue id is then put within a new uniform resource locator request to grab the hour's data array of each venues. The day_val variable is set with the selected option value and parsed to an integer to select the correct day within the query. If the day is located then the marker will show, but if false then the marker is hidden (Figure 4.5).

```
if (daysOpen = val.response.hours.timeframes[h].days.indexOf(parseInt(day_val, 10)) != -1)
```

Figure 4.5 – Line shows the day element is selected within the JSON array by matching the users selected day value.

The day result is combined with the time range since both data are available within the request and does not contain a search parameter within the search request.

Getting the correct time range.

As mentioned before the time range does not have a search parameter within the API similar to the day selector. Therefore the time range will also need a function to run the calculation, to hide markers that don't fit within the time frame. Similar to how the day function is set, the time range will called once search request has been successful within the same function as the days (getHours()). The hour values have been set to 24 hour time format as it is machine readable and the API will only accept this format. The selected values are then set into variables, one for the start time and the other variable for end time. While each venues are being looped through it grabs the start and closing time matching the day selected , for example user selected Monday so only the opening time and closing time for that is grabbed. The most challenging part was to compare the opening and closing time from the JSON result with the user's selected start and end time.

The theory of the query is to check if the selected start time is greater or equal to the opening time AND the selected end time is less or equal to the closing time. If true then the correct venue markers are shown, but if false the markers are hidden. But this logical statement set some of the venues markers to be hidden even though they were open within the timeframe. For example the

user wants to check for shops open between 6am to 11am, the start and end value are 6:00 and 11:00. The popular opening time is 9:00, so the query will be:

If (6:00 >= 9:00 && 11:00 <= + '+' + 17:00)

6:00 is not more or equal to 9:00 therefore the venue marker is set to be invisible since the logical operator '&&' requires both statements to be true. So the solution was to change the logical operator to 'OR' instead of 'AND', therefore if the first statement is false, it will simply move on and checks the next statement (Figure 4.6).

```
if ( start_val >= openHours || end_val <= + '+' + closeHours)
{
    console.log(start_val);
    console.log(openHours);
    console.log(closeHours);
    marker.setVisible(true);
}
else
{
    marker.setVisible(false);
}
```

Figure 4.6 – Logical statement using and 'OR' operator instead of 'AND' to show correct venues.

The select type of venue allows the user to choose what venues they would like to see displayed on the map. Compared to the time range and day selection, the API did include a type parameter within the search venue section called 'categoryId'. This parameter uses ID's set by the API to limit the type of venues. Each of the category ID's are set to each option within the select box, then simply passes ID value to the search_map.js script, set within a variable and attached to the API search venue request(Figure 4.7).

```
<select id="category">
  <option value="">All</option>
  <option value="4d4b7105d754a06376d81259">Bars</option>
  <option value="4d4b7105d754a06376d81259">Nightclubs</option>
  <option value="4bf58dd8d48988d17f941735">Movie Theater</option>
  <option value="4bf58dd8d48988d181941735">Museum</option>
  <option value="4d4b7105d754a06376d81259">Shops</option>
  <option value="4bf58dd8d48988d16d941735">Cafe</option>
  <option value="4d4b7105d754a06374d81259">Restaurants</option>
</select>
div>
```

```
'&sw='+swLat+', '+swLng+' +
'&categoryId='+type+';
```

Figure 4.7 – ID set as value for each type and the added to the search venue API's endpoint.

Setting the Price

The price section is set out using a slider feature instead of a select option, as usability of selecting maximum price will be easier. The max price will help to limit the result by cancelling out venues that don't meet the price criteria. Again just like the times and hours the parameter for price is not

available within the search venue request. The pricing can be found within the venues request, called within the `GetPriceRange()` function. The API's works in tiers, 1 being the cheapest and 4 being the most expensive. But the slider works with number values ranging from 1 to 500, meaning a logical statement needs to be used to match the values with the API's price tiers.

Within the request success the value is grabbed from the slider and stored into the variable. Then the values are passed through a range statements each checking if the user's price value is less than indicated number, if so then price is given the correct tier number. If the price is available this then calls the `getHours()` function which query's the time range and hours.

The overall challenge of the search query was working out how each of the query items interact with each other since only the type of venue was included as a parameter within the venue search request. This was resolved by calling the search first since the markers had to be called first, as this was the only option to hide venues markers that didn't meet the time range, day or price criteria using the `setVisible()` method. The `GetPriceRange()` is then called passing the venue data and markers. The price was calculated first as an if statement was able to call the `getHour()` function and check the time range and day. The time and day required a separate URL request to grab the venues opening times therefore calling another function. It's also noted that Ajax is asynchronous, which means data would appear undefined when passed outside the request, therefore the function is called within the request. The day value is collected first since the day value was needed to make sure the selected days opening and closing time was collected.

4.3.4 - Displaying venues details

The next challenge was to be able to grab each of the venues and display within an info window which pops up when one of the venues icon has been clicked. The info window contains the venues title, distance, icon, type, address, number, social media links, and a website if available.

A link is also available which gains access in displaying more information about the venue which includes a photo gallery, opening hours, directions and an option to save the venue within the user's account.

Each of the markers are made clickable by adding a click event listener to the venues marker which then calls the `getVenueDescription()` placing each of information into the info window set from the function. The first issue was only one venue information was being called and outputted into the window, meaning each venues had the same detail. Looking further into the issues it turned out the

request was correctly returning all venues information but was only taking the last value. This was because the results were not being stored therefore as the looped preformed the click event, each venue detail was overwritten. To make sure each detail is stored the marker variable, which holds the marker event will hold each of the venues. When clicked, the `getVenueDescription()` function uses `'this.item'` to grab the correct details when marker is clicked.

Getting the address details

There's a lot happening within the `getVenueDescription()` therefore ordering and sorting was important to help avoid slow loading. The start of the function contains the URL call to collect data for the clicked venue using the venue's ID, when successful the first item collected is the best photo used to display on the top of the info window. But the best photo was awkward to get since the prefix and suffix had to be called at different times then joined together, with the size of the image inputted between the prefix and suffix. The variable containing the joint links is then put with the source attribute in the image element, allowing the image to load within the info window.

The function `'getVenueAddress()'` gets each venue address but getting it was tricky as the venues address details were separately located in the location object from the venue response. The venue location objects contains the city, state, postal code and street address for each venue therefore a loop was needed to check the number of venues found. If the venues have address details available, then each venue is given the correct address by matching the venue ID and the field array objects, the location array objects and then pushed to the address array. The field array contains the names of each object and then used within the loop to match with the API's locations objects.

```
function getVenueAddress(venue, fields) {
  var address = [];
  fields = fields || ['state', 'address', 'city', 'postalCode'];
  for (var i = 0; i < fields.length; i++) {
    if (venue.location[fields[i]]) {
      address.push(venue.location[fields[i]]);
      if (fields[i] == 'address' && venue.location['crossStreet']) {
        address[address.length - 1] += ' (' + venue.location['crossStreet'] +
      );
    }
  }
  address = address.join(', ');
  return address;
}
```

Figure 4.8 – The `getVenueAddress()` function retrieving the correct venue address.

Each venues passed through was set with the state, address, city and postcode objects first but the `crossStreet` was not always available within the response array. Therefore excluded from the array

and set into an If statement instead to check if both address and crossStreet is available, if true then the crossStreet object is attached to the address object. Finally when all logical statements have been checked the address array is joined together with commas to separate each object and then returned to the main display function.

Within the API's location object array, an object called 'formattedAddress' includes the full venue address in one object, but with the full address in one object it was difficult to style and position as it outputs the venues address on one line. Compared to the individual objects which can be set into separate variables.

Getting the icons details

Icons were set into a separate function similar to the address since the icon will be used multiple times throughout the web application. The icons have been used as markers on the map, displayed in the info window and pop up box. Similar to how the address function the venue id is passed to the icon function to match each venues with the correct icons.

The icons are found within the categories array and contains the prefix and suffix objects to grab the icon image. Each venue have multiple categories for example a bar can be classed as a bar and nightclub, creating two arrays within the object array but only one icon is to be selected. To solve this issue each categories length is checked and only the first array is selected by giving the value 0. This then selects the URL of the icons including the image size between the prefix and suffix.

But some venues do not have icons available since no category is set, causing the original google maps marker to appear as a default. To solve this, validation was included to check if the icon is a string, if true returns the icons URL and if no icon is found then the default API's icon is returned to the main display function (Figure 4.9).

```
function _getVenueIcon(venue) {
  if (venue.categories && venue.categories.length > 0) {
    return venue.categories[0].icon.prefix + 'bg_64' +
    venue.categories[0].icon.suffix;
    if (typeof(venue.categories[0].icon) == 'string') {
      return venue.categories[0].icon;
    }
    else{
      return venue.categories[0].icon.prefix + venue.categories[0].icon.sizes[0]
+ venue.categories[0].icon.name;
    }
  }
  else{
    return 'https://foursquare.com/img/categories/none.png';
  }
}
```

Figure 4.9 – getVenueIcon function gathering the venues icon.

The venue name, telephone number, social media and website objects don't require a separate function since it was a simple get and set method. Each did not require a loop to iterate through several object therefore If statements were just used to check if the venues have these details, if false then string message lets the user know no such item exists, for example no website is available. The website and social media response were the only two which had to be edited. The twitter respond did not include the iconic @ symbol, therefore was added to the link telling users it's a twitter link. The URL for the venue was edited so the http was hidden when displayed using the `replace()` method.

Each of the venues details are stored within variable in the venue description function and then added to the html variable. The html variable (containing an html format with the venues details) is then called back to the overlay info windows function and set using the `setContent` method.

The venues description are also displayed within the modal box along with added information such as the opening times and a photo gallery for the clicked venue. But instead of iterating through each of the venues details request functions again, each returned value was simply set into the modal box using IDs. But the opening times and photo gallery were required because both request have separate URL endpoints and responds.

Getting the individual opening hours

The hours within the modal box have to show the venues opening times including the day of the week. But as mentioned before the venues hours' response sets the object within the timeframe array with a day array and open array. But if the days do not have the same open segment they are then put into another segment with the timeframe. For example day 1 to 3 opens at 9am and closes at 6pm, but day 5 opens at 9am and shuts at 9pm. Therefore each timeframes have to be iterated to get each day's open and close array.

The hours function first sets an empty array to store the opening and closing time for the clicked venue and then checks if the venue has opening hours, if not then a simple 'no opening hours is displayed'. If true the loop iterates through the timeframe checking how many arrays are included. The length of the array is then past to a nested loop which checks for the number of days within each day array, thus picking out all of the opening and closing hours and pushing to the times array.

The next issue is to now match each opening hours and day, then displaying onto the modal, but the days of the week are returned in numerical format and not as a string format. So each day value

had to be matched with the day's name. Originally each time values were set into separate variables by manually picking out the array index and matched with the name of the day. But this method would only work if all venues are open seven days a week, which creates an index array from 0 to 6. But if a venue was closed on a Wednesday and Sunday the array would have a range of 0 – 4, therefore the opening times will not correspond with the days as it can't see the two closed days.

Getting venues photo and setting JQuery Plugin image carousel

The photo gallery section, also located within modal box, shows the venues photos taken from foursquare API. The gallery is to be positioned at the top of the modal box in a horizontal line where users can view the photo gallery. To get the photo's another URL request had to be made to receive all of the photos available for the clicked venue.

Grabbing the photos was simple enough as the photos response inputs the photos under one item array. This means one loop is needed to iterate through the items array and pick out each venue image's prefix and suffix which then can be added to the images source attribute. But each of the images were just sitting in one long straight line taking up the majority of the modal box. To solve this issue a JQuery plugin called Slick (Wheeler, K 2015) creates a carousel gallery effect.

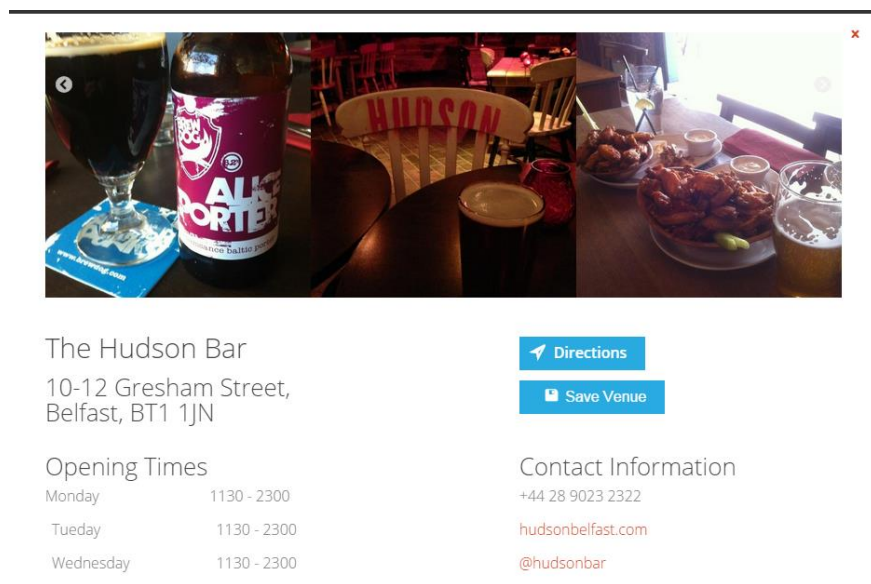


Figure 4.10 – Images are displayed within the slick plugin, creating carousel effect.

The plugin offers a range of features such as mouse dragging, responsive display, adaptive height and auto-play. The plugin is initialized by adding the slick method to a class, then slickAdd adds each

of the images found within the iterated loop to the photo gallery. An additional feature called auto play was also added which allows the photo gallery to automatically play when users open the modal box containing the venue details.

4.3.5 – Log in system with account

One of the main features within the Let's Go web application is the user registration and login system, allowing users to create their own personal account. Having their own account will allow the users to store venues from the search page. The account system uses PHP5 object oriented programming and MySQL. Before building the login and registration modals, the first step is to configure the database connection which can be initialize on any page.

Using server-side scripting language is one my weaknesses due to having little experience, therefore making the account system was challenging. The web application has multiple uses for the server side scripting, such as a log-in, registration, storing venues and deleting venues. With numerous functionalities happening the files became unorganised and lengthy, for example each functions had to the call the database connection multiple times implementing bad code practise. This also slowed down the system since so many duplicate calls were happening.

But a tutorial by Muni() provided useful material on how to build a user account using object oriented programming. Even though the tutorial states it's been made for a different framework, the script provided can still work with framework being used for the application.

The three main features included is the database connection file (config.php), the database class file (DBclass.php) and user file (User.php). The connection file contains the configuration to connect with the database and uses a special function called `_autoload()` which helps to reduce the number of require or include function when wanting to call a class file. The DBclass simply contains the database connection which includes a `'_construct'` function so the database connection parameters can be passed to the `$con` variable. Making it easier to call the database connection each time by simply using the `$con` variable within a class.

The user.php is the main file for carrying out all functionalities when called. Each functions are split into the public functions situated within a class containing the connection variable. To activate each function from the main page, a PHP code embedded at the top of the page which calls the class containing the logic functions and selects one depending on the name used. But the login and registration forms were both located within a modal box and each time a form was submitted to

activate a PHP function the modal would close hiding any validation errors. Meaning the user would have to reopen the modal box just to see the validation error. The solution was to use Ajax to prevent a page load so that the modal box stays open, but data still gets sent to the server-side.

Using AJAX and PHP5 functions together

Using the client server language helped to stop the modal box from closing to show any error message to the user while logging in or registering. When the user submits the form, the data is then passed through the request to the activation function not moved to a separate file. The correct public function is then activated, runs the data through the code and returns either an error or success message. But there is more than one function to be called through the site such as the log in and registration, therefore the script won't know what function to call. An extra piece of data was then decided to be passed through with the form data to help verified the function to call. For example when the user selects sign in, the data contained within the form is passed through the request with an additional 'reg' name stored within action variable. The form data are set as a string and the variable found in the data array is serialized so it can be used within the scripting language. The new serialized data is then sent to a PHP checks if the action variable has a value, if so then variable object (reg) is passed to a switch statement as an expression and compared with the case values in the structure. If a match is found then the registration function located within matched case is activated and 'break' to avoid the next case from activating. The function will then activate the corresponding function within the Cl_user class and return a statement so the request can decide what validation message needs to be shown (Figure 4.11).

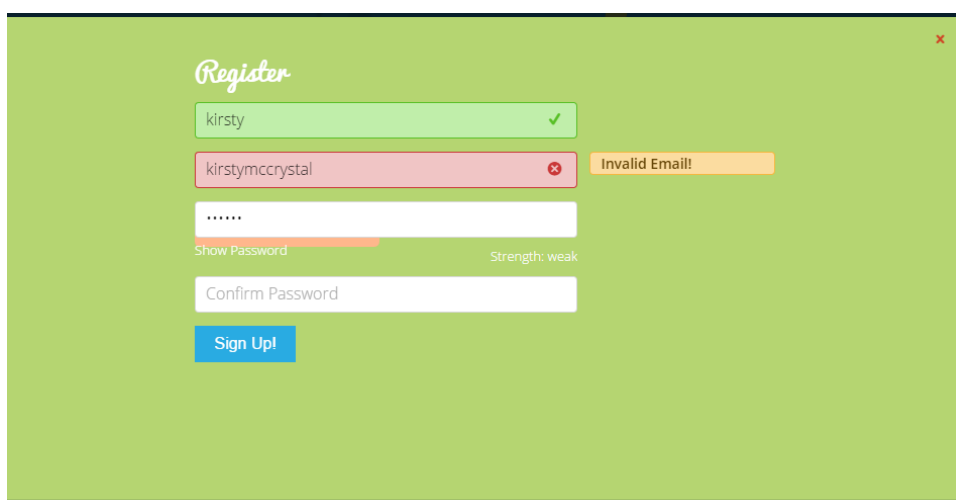


Figure 4.11 – Register showing validation and password strength checker using client-side and server-side scripting languages.

These actions are performed for the log in, delete, storing venues and deleting venues, each having their own unique set of server-side functions.

4.3.6 – Storing and deleting venues within the Database

Another challenging functionality within the application is the ability to save and delete venues within the user's account shown in the Figure 4.12. This section requires scripting language PHP5 and MySQL and use of a database to make sure each stored venue is displayed to the correct user.

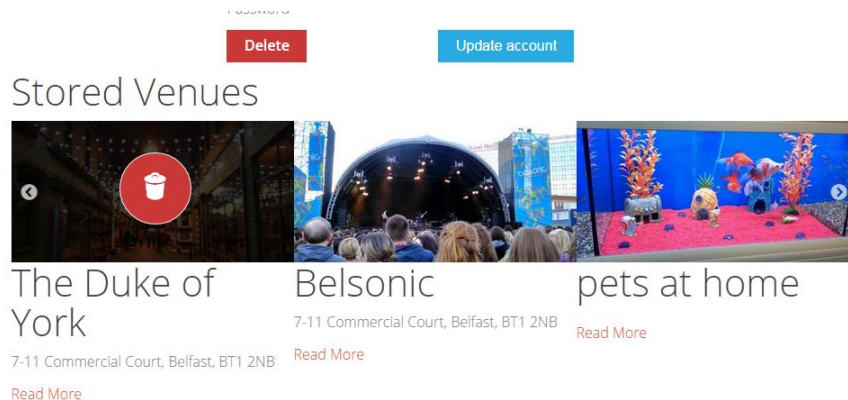


Figure 4.12 – Users account page containing their stored venues and an option to remove the venue as well.

Storing the venues within the database

There are two questions to ask when storing a venue within in an account, who's saving the venue and what is the venue? Answering these question correctly will help make sure each venues information is passed through securely to the correct user. Users can store venues by simply clicking on the stored venues button located within the modal box containing more information about the selected venues. When the user clicks the store button, a form submit is activated which passes each of the venues name, id, image and address within the modal to the database using the server scripting language. Each of data passed through are trimmed and escaped for security purposes. But each time new venues are stored within the database it refreshes the page by default and resets the search form and map section. This was problematic as users will then have to re-enter the same search criteria defeating the purpose for a fast and reliable web application.

To help fix this issue, Ajax was then introduce to stop a page reload and still send data to the user's database. The data will now be sent through the Ajax function using the POST request to the location of the PHP store function. The function will run a query checking against the users unique

ID and then inputs into the 'venuetable' table within the database. But at this stage users will be able to input duplicate venues which will clog the database and the stored venue within the user account. The only solution is to include some validation within the PHP store function. Each venue gains a unique ID (Primary key) when inputted into the database, which can be used to check for duplicates. Within the query all properties are selected from the venue table and checked against the venue currently being stored, also including the user id. An if statement is then used to check if the results output a row matching the query, if true the string value "Already saved!" is passed to the Ajax success function to notify the user they've already saved the venue (Figure 4.13). If the query can't find a user id then the response will be "Login or Sign up to save venue!" else the venue will be saved.

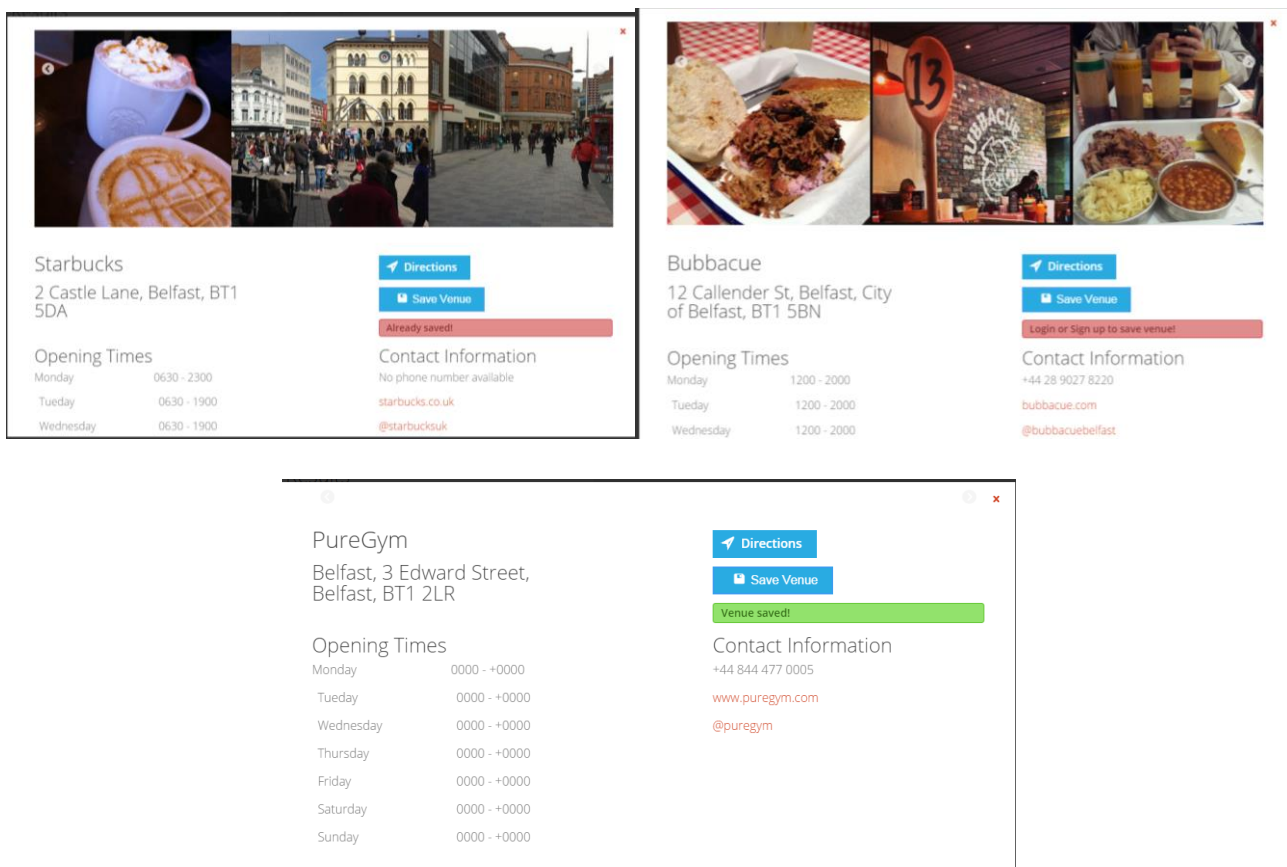


Figure 4.13 – Validation showing if user has either saved, not logged in or already saved.

Get the venues details

Each stored venues are displayed in the user account using the slick plugin to create a carousel effect. To get each of the venues was simple enough as a select was used to grab all data where the user id from the user table equals the user id within the venuetable and matches the active users ID. The complicated part was inputting each of the returned values into the carousel plugin using PHP. Since the query selects multiple matching images, each image then needs to be displayed within the carousel but when the plugin is passed through the loop, the carousel effects breaks and images are displayed vertically down the page. Looking more into the issue it turns out that each of the images has to be set within a div containing an img-slider class, as this gives each image element an index number. The index number is used to put each of the image on order within the carousel, therefore the img-slider class has to be included within the PHP loop along with venue name and address.

Delete each venue

Each of the images within the carousel will have an overlay containing the delete button when hovered. Thus giving the users the option to remove the venue from their list. The delete button is located within the carousel and includes a delete query within the PHP function. The delete function gets the data venue id from the stored venue form and matches the clicked venue Id with any ID that matches within the venuetable table. Similar to the store venue function, the page refresh was not allowing the delete function to run. So if a match is found the venue ID is returned using the Ajax response and then removed the form containing the venue ID using a slow effect. This means users will be able to see each of the venues disappear immediately.

4.4 – Other notable achievements

Other notable achievements within that should also be mentioned is the use of a password strength checker using a plugin called strength.js. The plugin provides security measures each marked by using scores depending on the amount of characters, lowercase, uppercase and numbers used within in the password. The feedback is displayed by showing a line below the password textbox which expands and changes colour depending on the strength of password. The password is then converted into a hash of strings using PHP5 md5() hash generator.

The notable achievement overall was the ability of having Foursquare API's and google maps working together to retrieve and display the correct venues that match the inputted search query, accomplishing the projects objectives.

5. Testing

5.1 - Test approach selection

The testing stage of the web development should be carefully planned out to help avoid any misfortunate bugs and errors appearing after going live. Testing the system will also assure if the system has met the user's needs also known as functional requirements. To help discover any defects or bugs a test approach will help in mitigating risks, ensuring for a fully functioning and successful web application. Taking into consideration of the interaction between html pages, communications and connections.

To discover what areas are needing to be tested the requirements and assumptions within section 2.3 which will help to create a list of objects to test. Each of the requirements can then be sorted under different testing methodologies to determine that each function works according to their purpose. Some of the test cases can even appear multiple times within in different methodologies. Helping to reduce the risk of missing any bugs in the testing process. The Let' Go web application aims on assisting users to search and locate venues within an area by inputting their own preferences. The users are then able to store or gain directions to their chosen venues, therefore the visualization and navigational consistency is the highest priority for testing throughout the application, since this matches up with requirements and user's needs.

Running through the requirements and assumptions it's clear to see the form of testing processes that needs to be carried out. From functionality testing to performance testing. The following types of testing to be carried are:

5.1.1 - Functionality testing

The functionality testing, also known as black box testing, covers the functionality within a web application taking into account of the navigation between webpages, connectivity between the application and database, submitting and getting information from forms and any other additional functions. The let's go web application has a lot of functionality such as searching for venues and displaying on the map, therefore this test will be top priority.

5.1.2 – Interface & usability testing

The interface & usability test is another vital test to consider since the Let's Go web application requires users the ability to use the application with ease. These tests will cover the navigation, accessibility, cross browser testing, error messages and the layout.

5.1.3 - Compatibility testing

One of the non – function requirements states that the Let's Go web application must be compatible on multiple platforms, devices and browsers. Therefore a compatibility test will cover all of these non – functional requirements to ensure the application can perform within different environments. The cross browser compatibility is important since all users won't use the same browser to view the web application.

5.1.4 - Performance testing

The application uses a lot of requests from web using API's therefore a performance test is important to make sure any data is loaded quickly. Web load testing check how the web application performs on different internet connection and will it work at peak load times. Stress testing also can determine if the application can perform will by going over the specifications limits, for example ensuring the map and venue don't break if the user clicks the search button multiple times.

5.1.5 - Security testing

The application requires for a log in account, holding personal data from the user. Therefore a security test is important to test for data protection, correct authentication and data integrity. No other users should be allowed to access any other account apart from their own. SQL injection is a serious but common issue when using databases, as hackers could easily run a SQL command to gain access to the user's personal data. Causing damage to the data or theft.

The testing strategies being used within for the Let's Go web application will be the functionality testing which will include testing each of the functional and non-functional requirements. Other tested will be a compatibility test and a requirements survey response. These tests felt suitable since each of the requirements operates the application, therefore if all requirements work then the application will work. The other tests will cover the usability and interaction of the web application.

5.2 - Test process

The testing process details how each test were planned out, analysis, implemented, evaluated and then closed. Within the planning stage each of the risks were identified first, as these risks were important to avoid. For example invalid results could send users to a closed venue, therefore this object will have to be tested first, along with the requirements stated in chapter 2.2. Once all of the tests have been listed next was to analysis each the test case and work out what needs to be prioritized and which test is suitable. Since all the objects needed to be tested are requirements, they were then just tested individually to confirm they work and meeting the aim and objective. Once each test's has been set the next step was to implement these tests using different techniques to determine if any bugs appeared. For example writing in different forms of password to test the strength indicator or testing if the email format validation works.

Finally once all of the testing has been completed, the next stage was to evaluate the test results. To confirm all tests have passed or check to see if any major bugs were found have been fixed. Once all confirmed each of the tests can then can be closed.

5.3 - Test results

5.3.1 - Functionality results

After working out what tests were important to carry out, the next stage is the testing stage. The first test's result shown in Table 5.1 shows the functionality test's results. As you can see all functional requirements have be met with only a few minor bugs within the search criteria. The venues are shown depending on their type, day, price and location but the hours is partially working, as there's always one venue that does not match the venues time range.

Requirements	Tests	Expected Result	Result
The product shall allow users to Log in to their own account.	Test by logging in using existing username and password.	User is logged in and has access to account page.	Passed - Log in system worked.
The site shall allow users the availability to create their own personal account.	Test if users can create their own account.	Account has been added to the database and user is logged in.	Passed - Account was created and automatically logged in.

User account must be safe and secure and can only be access using a username and password.	Test to see if only the users email and password can access their account.	Error message showing if either username or password is incorrect.	Passed - Inputted incorrect username and password separately and both showed error message.
Allow users to draw freehand the location on the embedded map.	Draw a search radius onto map.	Free draw the search radius onto map and automatically close.	Passed – Was able to free draw on map and closed automatically when finished.
The search button shall take into account the map, time, cost and type queries and output correct result.	Test if correct information from the search form is being processed correctly.	Correct venues icons to appear within the map, matching the search criteria.	Partly Passed – Only venues matching the criteria appears within map. But some venues appear not matching the timeframe.
Icons must be displayed within the hand drawn search radius.	Test to see if icons stay within the radius.	All venue icons should appear within the search radius.	Passed – All icons located within circle.
Users should be able to search for a place, then automatically displayed on the map.	Test search bar to be shown on map.	When place as been inputted into search bar, the map should automatically display it.	Passed – Map successfully displays correct place correctly.
Users should be able to store the selected item in personal account.	Test store venue button and check if data has been passed	Venues should be stored within the users account and views within their profile.	Passed – Store details are passed to database and

	to database in correct account.		shown within user account.
Users must be able to gain directions to venue from current location.	Test that users can get directions to correct venue from current location.	When directions button is clicked, directions from current location to venue appears on map.	Passed – correct directions shown on map from user’s position to selected venue.
User must be able to choose the form of transport.	Test by selecting different forms of transport.	Transport is clicked which then displayed automatically on map.	Passed – Map displays new transport directions
User must be provided with different route options.	Suggestion routes provided and changes route.	At least three route suggestions which changes on map, once selected.	Passed – Three correct suggested routes available and changes map on when selected.
User must be provided with written directions.	Check if correct directions is shown.	A list of directions shown to correct venue.	Passed – Full list is available.

Table 5.1 – Table showing a list of functionality test’s with indication of passing or failing.

5.3.2 – Non-functional requirements result’s (Compatibility testing)

Moving onto the non-requirements result’s the web application has been tested various types of the browsers, devices and operating systems to determine it meet’s, not only the non-functional requirements but also the functional requirements.

Cross-browser

To test the browsers fully each of the requirements from the functionally test will be used. To find any issues with working on different browsers. The results shown in appendix (A.8) shows any issues any issues that appeared while cross browser testing. The results shows testing done on four of the most popular browsers(Helena 2015), all receiving positive results apart from a couple of CSS issues on internet explorer and Firefox.

Responsive Testing

Appendix (A.9) shows the results of testing each function requirements on different devices. Overall most of the functionality works but there is a few usability issues. The image sliders located within the user accounts don't show but works in landscape. The responsive testing (Pugsley.T, et al.) site used tests the layout and the how it looks, but can't 100% test the users interaction. Therefore moving to an actual handheld device, it became apparent that users aren't able to draw on the map without moving the screen or select an icon. These issues are major issues therefore need to be fixed. The icon issue was simple as a simple touch event was used to work on touch screen devices.

5.4 - Requirements and objectives responses

5.4.1 - Task Completion Rate

The following results below shows if each participant were able to complete six of the most important requirements within the Let's Go Web application. The user's then rated between 1 (strongly disagree) to 5 (strongly agree).

Participant	Task 1 Create a new account.	Task 2 Search for a bar within Belfast.	Task 3 Select a venue and get directions	Task 4 Select venue and save	Task 5 Find your stored accounts	Task 6 Delete a stored account.
1	✓	✓	✓	✓	✓	✓
2	✓	-	✓	✓	✓	✓
3	✓	✓	-	-	✓	✓
4	✓	✓	✓	-	✓	✓
5	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓
Success	7	2	5	4	7	7
Completion Rates	100%	86%	86%	57%	100%	100%

Table 5.2 – Table showing usability rate for each task completion.

5.4.2 – User Feedback on the functionality and usability.

Table 5.3 show six users feedback indicating if the site was easy to use and helped to pick up any issues, such as difficulty with the search features. Each number represents the user's choices.

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Was the application easy to use				6	
Would you use frequently			3	3	
Were you clear of your current location on site?					6
Was it easy to search for venues?			1	5	
Was venue information (if available) adequate?				6	
Easy to create an account and log in.		2	3	1	
Can easily get directions to venues?				6	
Easy to store venues?			5	6	
Easy to find stored venues and delete?				6	
Was the website well organized?		1	4	1	

Table 5.3 – Table showing user feedback on application usability.

6 – Evaluation

With the Let's Go web application completed and tested it's to evaluate the overall outcome of the project. Including the tests/survey results, the project outcome, methodology used and the planning that took place throughout the project.

6.1 - Evaluate test results

Reviewing over the results of the testing stage and requirements/objectives results, it's clear to see the web application has passed the majority of the requirements and usability testing. The only notable issues that arose within the testing is the time frame not calculating correctly and responsive issues. The time frame error would be classed as a major issues since it is classes as one of the main requirements within the Let's Go Web Application. This would also cause havoc for the users as closed venues would be shown within the map, therefore users will be able to click on the venue, get directions and travel to the venue only to find out it's closed. Thus reducing the reliability and trust with the web application. This also defeats the aim of the project mentioned under the concept heading. The solution was to change the logical operator to a for loop which can iterate through the times and pick out the correct venues.

Not only were the functional requirements but also the non-functional requirements, testing the application for cross browser capability and the responsive. The application has to work on desktops and mobiles devices for flexibility purpose so testing across different browsers and devices was vital. Each non-functional test carried out returned positive results apart from a view minor styling setbacks, when the application is used on Internet Explorer and Firefox. The responsive test also resulted in a few styling issues, but also resulted in a major bug within the maps section. Users are unable to draw on the map without moving the screen and venues description were slow to generate after being clicked.

6.1.1 - Requirements and objectives results

Two separate tests were used to check if the users were able to perform tasks and feedback on how they felt about the usability of the site. Table 5.2 shows that 100% of users were able to complete task 1, 2, 5 and 6. 86% of user able to complete task 3, while only 57% of users were able to complete task 4. Task 1, 5 and 6 were all related to the user account, therefore it was clear see that the account system has been set up correctly. In task 3 one user was not able to perform the task of selecting a venue a get directions, which was also the same user who could not save it either

which only got 57% completion rate. This means the user will need clearer instructions on how to save and gain directions to the venues.

Table 5.3 results shows how all six users found the application was easy to use with a split vote of users being neutral or agree on using the application frequently. 100% of the users strongly agreed on the current location. Moving towards the search feature five users agreed on easily searching for venues and all six agreeing information was adequate, easy to get directions and storing and deleting venues. Overall the users have all mixed feelings about the site as four were neutral, one agreeing and one user disagree on the site structure.

Overall each of these feedbacks have been taken into account and updates have been made to make sure the user can enjoy and understand the Let's Go Web application.

6.2 - Evaluate project outcomes

Many techniques and the designs were used to achieve the web application, using specific technologies, tools and designs aiming for a good project outcome. The Let's Go Web application produced at a high level application after all bugs found within the testing section were solved. Using the chosen client-side and server-side scripting such as PHP5/MySQL with jQuery/AJAX worked perfectly together to build a dynamic application. Helping to preform faster HTTP requests and output immediate validations messages without refreshing the page.

The search page requires a lot of processing from the application to the server, as requests were made to application program interfaces to display correct venues, saving venues to the users account and a log in/register system. Proving an asynchronous web application was required to allow for high processing.

The two API's selected were very beneficial to use as google maps, offered the ability to draw on the map and display the venues onto the map, using coordinates. The Foursquare API therefore was used to grab all of the correct venues and display onto the map, since it offers a wide range of venues and venue details, all provided from individual people instead of businesses owners. This means only the top venues users have rated for will be displayed, matching the inputted search criteria. The drawing tool worked out to be a challenge has users had to be able to draw freehand and see venues within the drawn shape. Using polylines with trigger events allowed users to draw dependant on the mouse action, then combining the lines to create a polygon helped to calculate the area using the polylines coordinates to gain the centroid and edges.

The structure of the site was based on responsiveness and usability across multiple devices following design guidelines. The only major change with the user experience design was repositioning the map and search form beside each other so users had easier access to the form if any changes were needed. One of the many risks for a web application is the security since it's more vulnerable to attacks, Thus why areas with inputs should always have validations, for example the log in form. With each of the forms including validation on the client and server side and tested with the requirements, it resulted in with positive feedback, displaying how each correct validation message appeared for each issue.

The database design shows three table for user, venues and directions. Each users can save a venue of their choice meaning the user and venue table has a relationship. Using the user Id to match with venues was the key to get the correct venues correctly, as each user ID is unique. Securely linking the relationship means user will only be allowed to see their own data and stored venues.

Overall by having good design, technologies and validation produced a secure, accessible, responsive and reliable web application.

6.3 - Evaluate the methodology

The methodology helped greatly to plan, structure and point out any risks and challenge that could appear throughout the project. Choosing the waterfall methodology offered a step by step task setup for easy readability. Since this project was created by one person the chosen methodologies was suitable as it used a start/end dependencies, therefore fitting within the thirty three weeks project timeline. Splitting each of the task into categories helped to organise time, resources and ideas process what needed to be done. Each tasks were divided within research, design, development and testing/launch.

The research stage was to help plan out the suitable technologies and design for the web application. Then organise the areas which needed to be studied and learn, for example how to use google and foursquare API's. Then the design stage helped to discover the best layout and system designs, meeting w3 guidelines, accessibility and usability. Thus helping to define what areas needed authentication and requests. Next the development stage which was given more time as this was the implementation and had a higher chance of issues appearing. Thus leaving room to help manage any issues. After development was the testing stage, which again had more time since issues appeared such as the miscalculation of the time frame.

Leaving a week gap between the testing has helped to reduce the risk of failing to meet the deadline by using this gap encase of any emergency arose during the development of the project. Such as sudden illness, family issues or death in the family.

7 - Conclusion

7.1 - Summarise report

Overall report covered the context, management, design, development and evaluation for the Let's Go Web application project. The report summarises how each stage contributes to the development of the Let's Go web application following the waterfall methodology. Pointing out the main requirements, such as the ability to allow users to search for venues by matching their inputs helped to develop the systems designs and layouts to suit the user's needs. Using technologies such as jQuery, Ajax, PHP/MySQL to help build a dynamic and secure application while following W3 guidelines. The Let's Go application also users to log in/register, search, pick, save/store venues and gain directions to venues. The Let's Go web Application has been fully tested for any bugs that would interfere with the requirements.

7.2 - Reflect on what happened

Each stage has had different outcomes of results, some more difficult than others. During the concept stage defining the scope was difficult as it was hard to discover how long each task would take to do and predict any issues that could appear within the project. Even though the waterfall gnatt chart helped to plan each task, unpredicted issues could out the project at risk which lucky didn't happen. During the design stages, the entail design kept changing throughout the development stage such as the layouts design due to unsatisfactory with the interaction and usability, the map being the main culprit. Within the development stage the draw search and making sure the venues turned out to be the most challenging. With a lot of calculations and figuring out suitable methods while using the combination of Google maps and Foursquare API's. Overall all challenges were accomplished with the majority of the functionalities passing within the testing stage.

7.3 - Reflect on your role

During the development of the Let's Go web application, I've clearly set out each of the requirements determined from the aims and objectives. Chosen the correct methodologies to help discovered the different methods and plan each task suiting the waterfall methodologies. But in future I would have gave myself more time to discover more design solution for the user experience design.

7.4 - Future work

For future improvements on the web application, the main feature that would be updated is making the map static on the page but make the search and get directions panel movable. Allowing users to easily view the venues within the map and interact with the search form and get directions panel quickly, improving the usability of the project. Another additional feature would be to allow users the ability to add friends to their account, so they can send invites to friends, so they can travel to the chosen venue together.

Within the testing stage, a majority of six users tested out the web application to find out if all requirements have been met or if there were any additional needs for the application. But this is only a small number compared to the targeted market. To gain a wider variety of feedback from multiple users, the application could be published online to gather the target market results.

Reference

Compuware (June 2012). *Mobile Apps: What Consumers Really Need and Want*. Detroit USA, Compuware. Available at:

http://offers2.compuware.com/rs/compuware/images/Mobile_App_Survey_Report.pdf. [Accessed 12th October]

Property Pal. Propertypal.com Map Feature. Northern Ireland, Property Pal. Available at:

<http://www.propertypal.com/draw?max=500&maxbeds=3&sta=toLet&term=25&pt=residential&radius=5.0> [Accessed 9th October]

Weotta (2014) Get Inspired. Discover awesome ideas for every occasion. USA, Weotta. Available at <http://www.weotta.com/> [Accessed 9th October]

Strout, A. (26th June, 2014). 5 Key Trends To Watch (And Use) In Mobile/Location-Based

Marketing. Marketing Land. Available at: <http://marketingland.com/5-key-trendsmobilelocation-based-marketing-88394> [Accessed 10th November].

Lo min ming (7th July 2014) *UI, UX: Who Does What? A Designer's Guide To The Tech Industry*.

[Online] Available at: <http://www.fastcodesign.com/3032719/ui-ux-who-does-what-a-designers-guide-to-the-tech-industry> [Accessed 8th April 2015]

Zoltán Góczy (2014). *Myth #19: You don't need the content to design a website*. [Online] Available at: <http://uxmyths.com/post/718187422/myth-you-dont-need-the-content-to-design-a-website> [Accessed 8th April 2015]

Motive Glossary. (24th December 2004). *Client-side, server-side*. The Motive Web Design

Glossary [Online]. Available at: <http://www.motive.co.nz/glossary/client-server.php> [Accessed 16th December 2014].

Patil, A. (3rd September 2013) *What are the pros and cons of each "Places" API? Quora*. [Online]

Available at: <http://www.quora.com/What-are-the-pros-and-cons-of-each-Places-API> [Accessed 10th February 2015].

Wheeler, K. (22nd April 2015) *Slick the last carousel you'll ever need*. [Online] Available at:

<https://github.com/kenwheeler/slick/>

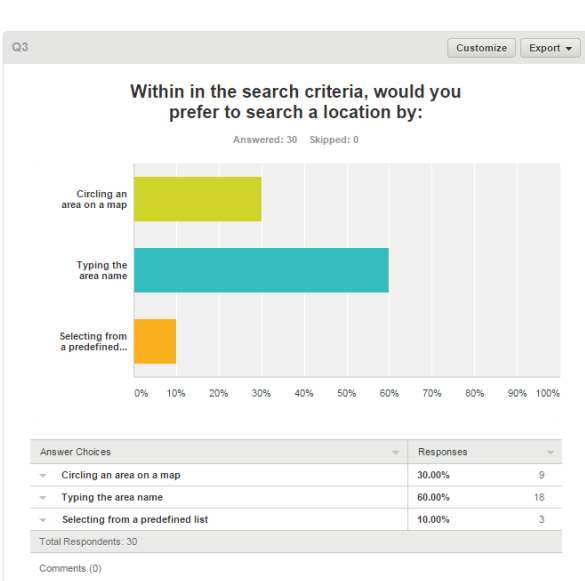
Muni (). *User registration and Login Using PHP5, MySQL, jQuery and Bootstrap*. [Online] Available at: <http://www.smarttutorials.net/user-registration-and-login-using-php5-mysql-jquery-and-bootstrap/>. [Accessed 23rd April 2015].

Helena (2nd February 2015). *Top 5 web browser*. [Online] Available at: <<http://zeendo.com/info/top-5-web-browsers/>> . [Accessed 24th April 2015].

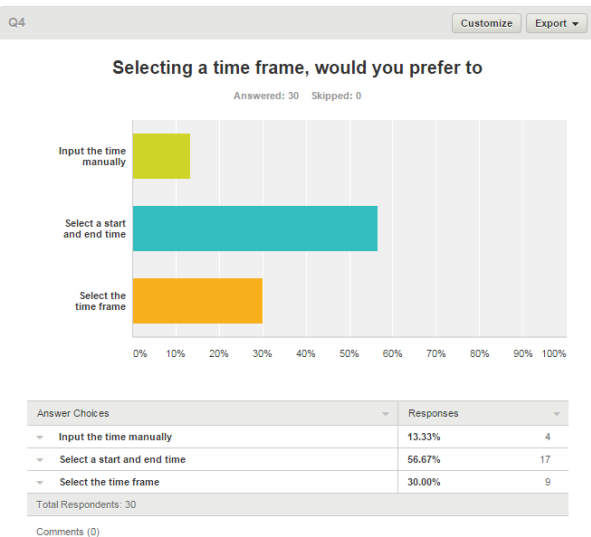
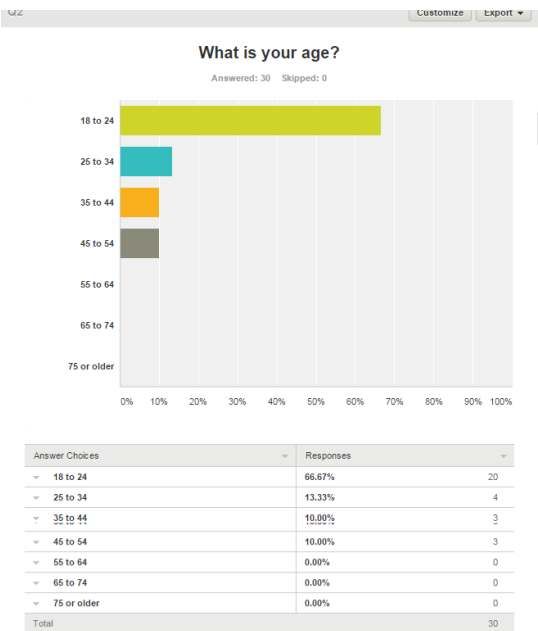
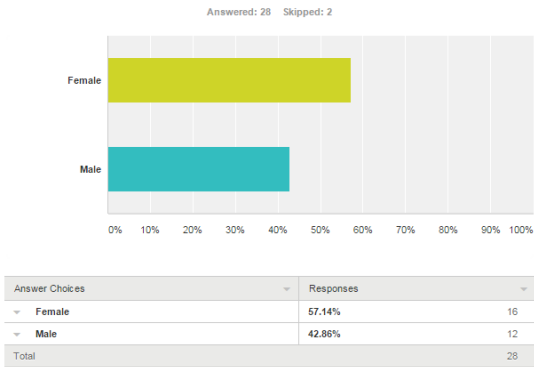
Pugsley. T, Hovey. A. (). *Responsinator*. [Online] Available at: < <https://www.responsinator.com/>> . [Accessed 24th April 2015].

Strojny, D (July 24, 2013). *Website navigation menus: Why less is more*. The Theme Foundry®. Available at: <https://thethemefoundry.com/blog/website-navigation-menus-why-less-is-more/> [Accessed 30th October]

Appendices - A.1 – Requirements Survey Responses



The app will allow users to search and receive directions for a range of activities, for example clubs, pubs and museums based their chosen location, price range, timeframe and type of venue. Users will also be provided with directions to the location and information on various modes of transportation. What is your gender?



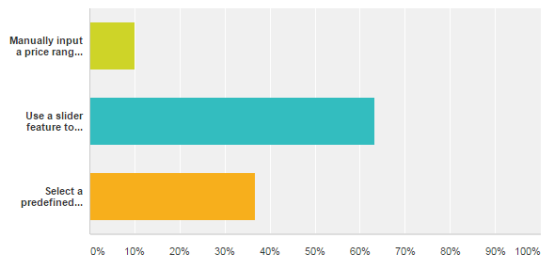
Q5

Customize

Export

How would you prefer to select a price range?

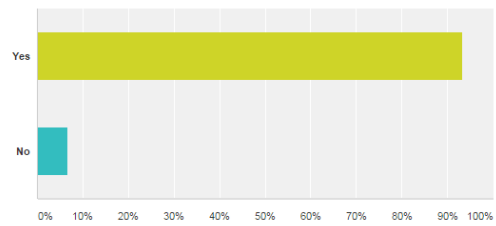
Answered: 30 Skipped: 0



Answer Choices	Responses
Manually input a price range (text boxes)	10.00% 3
Use a slider feature to select the minimum and maximum price range	63.33% 19
Select a predefined price range from drop down box	36.67% 11
Total Respondents: 30	
Comments (0)	

Would you like the option to record/store the chosen venue?

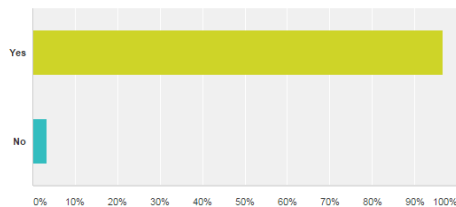
Answered: 30 Skipped: 0



Answer Choices	Responses
Yes	93.33% 28
No	6.67% 2
Total	
30	

Would you like an option to view different forms of transport to your chosen venue?

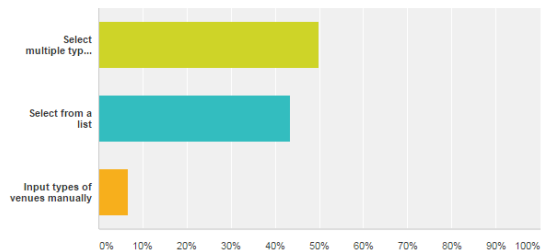
Answered: 30 Skipped: 0



Answer Choices	Responses
Yes	96.67% 29
No	3.33% 1
Total	
30	
Comments (1)	

How would you prefer to select the type of venue?

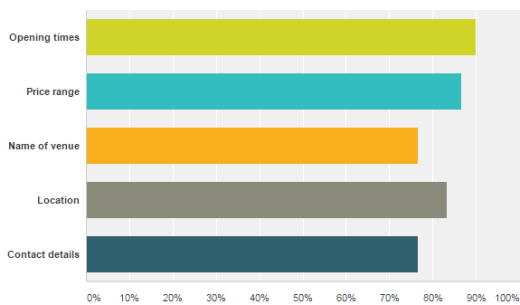
Answered: 30 Skipped: 0



Answer Choices	Responses
Select multiple types using checkboxes	50.00% 15
Select from a list	43.33% 13
Input types of venues manually	6.67% 2
Total Respondents: 30	

What information would you like to see when viewing information about your chosen venue?

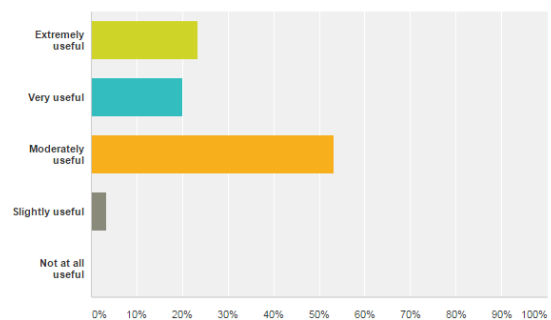
Answered: 30 Skipped: 0



Answer Choices	Responses
Opening times	90.00% 27
Price range	86.67% 26
Name of venue	76.67% 23
Location	83.33% 25
Contact details	76.67% 23
Total Respondents: 30	
Comments (1)	

How useful would the Let's Go web application be to you?

Answered: 30 Skipped: 0



Answer Choices	Responses
Extremely useful	23.33% 7
Very useful	20.00% 6
Moderately useful	53.33% 16
Slightly useful	3.33% 1
Not at all useful	0.00% 0
Total	
30	

A.2 – Constraints Solutions

Constraint	#1
Description	The user will need to have internet connection or access to mobile data networking for the web application to work.
Rationale	User can't gain access to internet or mobile data networking.
Fit Criterion	User will have to make sure they have a suitable connection to use the app. Most areas have good mobile networking.

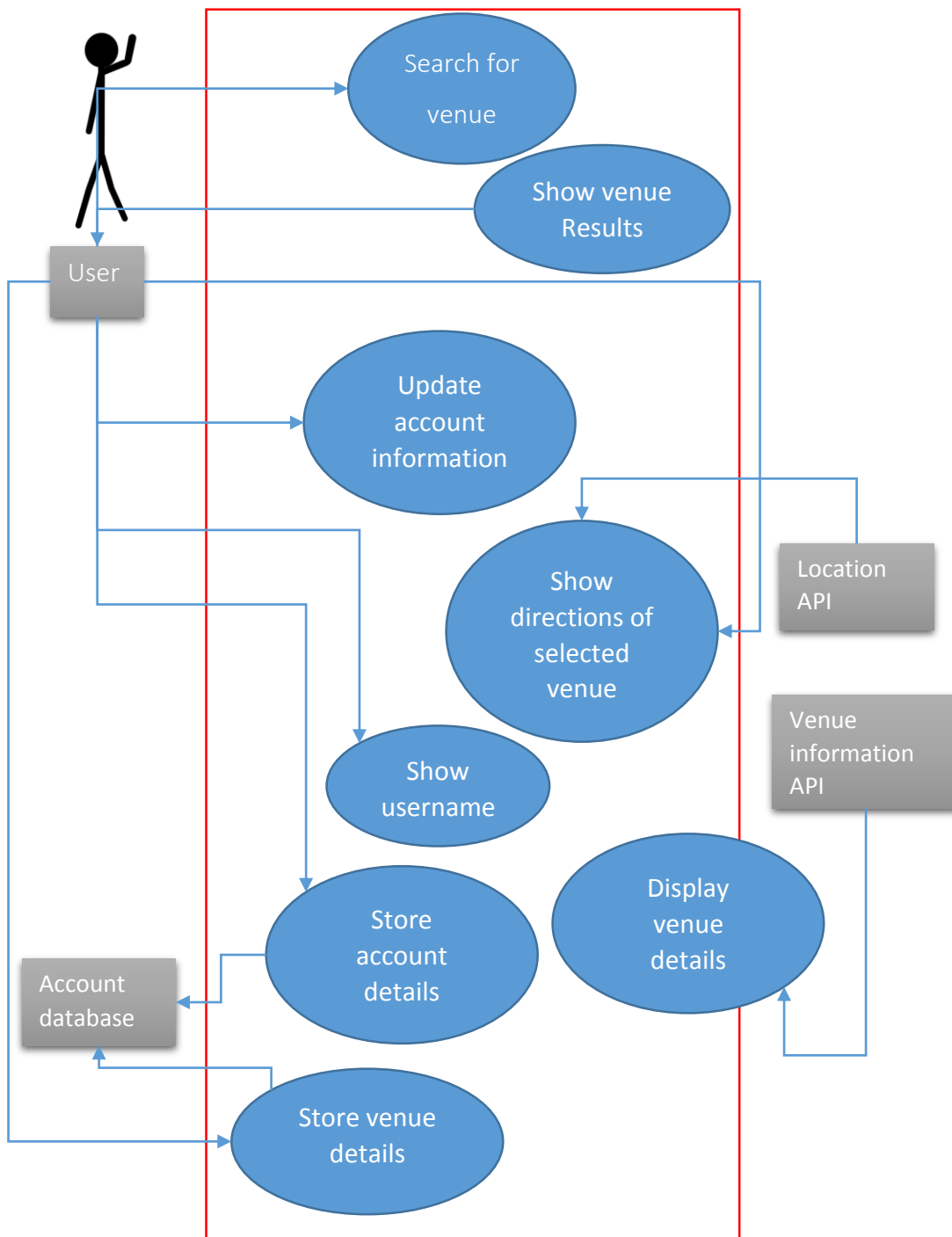
Constraint	#2
Description	Users will be able to choose and view transport and travel information.
Rationale	Transport information not updated from external source (Transport companies) including services available.
Fit Criterion	Make sure travel companies have services updated regularly for up to date travel information for users.

Constraint	#3
Description	The web app shall allow users to view the chosen venue directions.
Rationale	Google map does not show the best route, but instead chooses the longer route.
Fit Criterion	Allow users to change route to venue on the interactive map located on the get direction page.

Constraint	#4
Description	Displays more information about selected venue.
Rationale	Information for venue not available or inadequate.
Fit Criterion	Use the best API plugin detailing numerous venues' information.

Constraint	#5
Description	The project will allow users to create their own account.
Rationale	User does not want to create an account therefore not able to save venue information.
Fit Criterion	Allow non-account holders to use the main feature of the web application (venue searching), but venue information can't be stored unless they have an account.

A.3 – Use case diagram



A.4 – Functional Requirements

Requirement	#1
Description	The product shall allow users to Log in or create their own account.
Rationale	To be able to log in or sign up using the customers email and password previously created within the sign in.
Fit Criterion	The inputted email and password must match the account database, if matched then give user access to their own account. Sign in will require users information to import into database using secure validation.

Requirement	#2
Description	User account must be safe and secure and can only be accessed using a username and password.
Rationale	User's account details must be protected from other users using a unique username and password.
Fit Criterion	Username must not match any usernames and password must meet certain criteria.

Requirement	#3
Description	The product should display an account page detailing the user's name, email and hidden password. The users will also have an option to edit or delete their account.
Rationale	To view/edit their account details in a secure environment.
Fit Criterion	The details must match the user's information.

Requirement	#4
Description	The account page will store/record all saved places the user has visited or wants to visit.
Rationale	To be able to view and get directions of all stored items the user has saved within the account page.
Fit Criterion	The slider of all stored items should have the option to view details and then get direction of venue from the user's account.

Requirement	#5
Description	The user will be able to see venue icons on map and select to view the details of the venue.
Rationale	To be able to see limited information for each venue and decide if users want to view more information or not.
Fit Criterion	Once users hover over the icon the small snippet of information will appear. The information must match the hovered icon.

Requirement	#6
Description	User will be able to gain direction or save venue to their account.
Rationale	Option to get directions to venue and store in user database.
Fit Criterion	Must gain correct directions from user to venue and save into correct account.

Requirement:	#11
Description	Allow users to select the amount they would like to spend.
Rationale	To be able to acquire the minimum and maximum amount the user would like to spend, which will then be brought forward for the query search.
Fit Criterion	Slider feature will allow the user to select a minimum and maximum amount by moving the pointers to destination on the slider plugin.

Requirement: #12	
Description	Allow users to select the types of venues they want to be displayed.
Rationale	To acquire the selected type of venues the user would like displayed in the search result page, which will then be brought forward for the query search.
Fit Criterion	Checkboxes indicating the type of venues should allow multiple checks for variation.
Dependencies	

Requirement: #13	
Description	Checks if all queries have been filled in or selected.
Rationale	To be able to output a warning indicating a certain field has not been filled or selected.
Fit Criterion	The map, time, cost and type are checked for completeness, if not an error message will pop up indicating a fault. Otherwise if true the search results are shown.
Dependencies	

Requirement: #14	
Description	The search button will take into account the map, time, cost and type queries and output the correct result.
Rationale	To collect all queries made on the search criteria page and output the results.
Fit Criterion	Each query section (map, time, cost and type) must be completed and then the correct search result will be returned, taking the user to the result page of the web application.

Dependencies	13
--------------	----

Requirement: #20	
Description	To store the selected item within the user's account
Rationale	To be able to save the venue information for future purposes and remind the users where they've been or would like to go, if the user is logged in.
Fit Criterion	Stored item will be go straight to account page, with message appearing stating the venue has been stored.
Dependencies	1,2,3

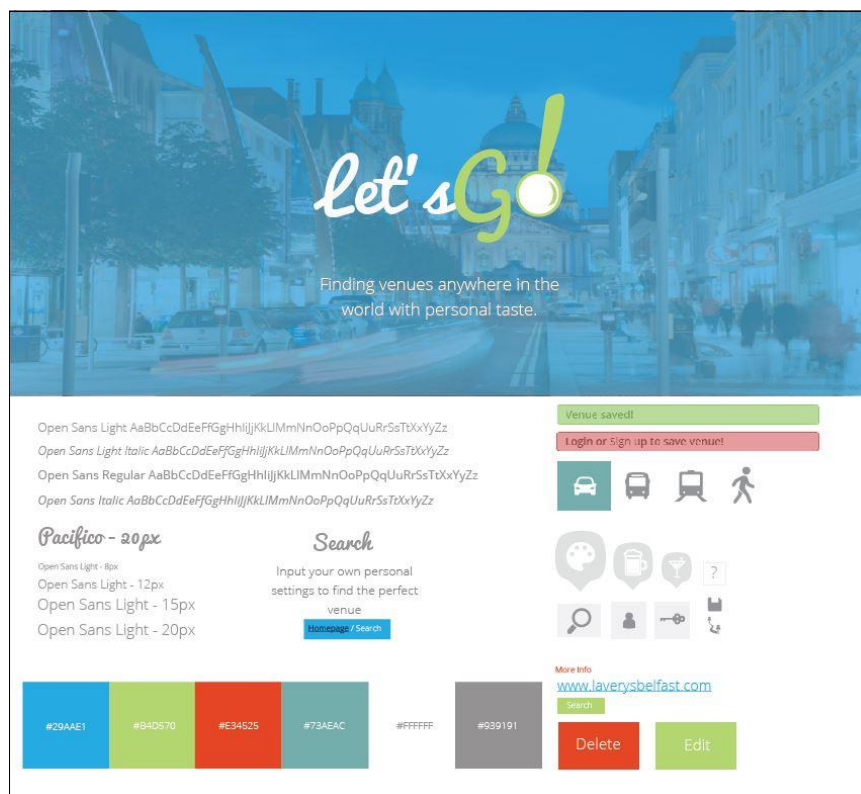
Requirement: #21	
Description	Allow users to gain the directions to chosen venue
Rationale	To able to view the directions from current location to venue taking the user to the get directions page.
Fit Criterion	The get direction button must take the user to the get direction page, allowing them to view the directions from their current location to the venue.
Dependencies	16,19

Requirement: #22	
Description	The web app will show a map route to the chosen venue.
Rationale	To be able to show the best route from current location to the venue.
Fit Criterion	Show the location of correct chosen venue, and the best route from the user's current location.

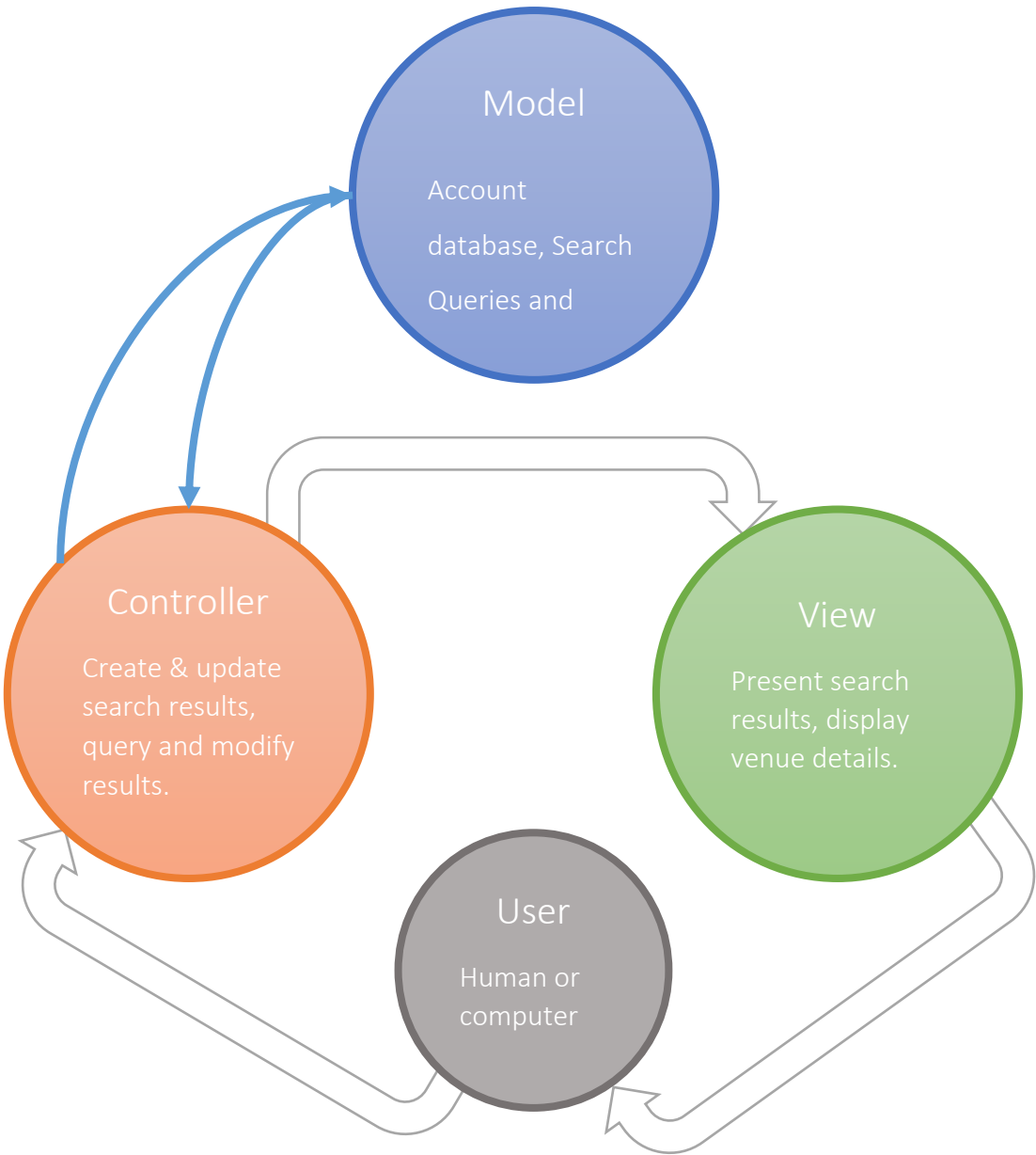
Dependencies	21
--------------	----

Requirement: #23	
Description	The direction page will display a choice of transport and detailed instructions of directions.
Rationale	To allow the user a choice of what type of transport they would like to use and how to get to the location.
Fit Criterion	Tab menu detailing the different transport, once selected the correct direction list will appear along with the duration of the journey.
Dependencies	21

A.5 - Style board for web application



A.6 – Model View Controller



A.7 – Tables showing the structure of each data

User Account Table				
<u>user_id</u>	name	email	password	created
Int(50) Auto_Increment Primary Key	Varchar(250)	Varchar(260)	Varchar(250)	Timestamp

Venue Details				
<u>Venue Id</u>	V_id	VName	Vimage	user_id
Int(11) Auto_Increment Primary Key	Varchar(500)	Varchar(250)	Varchar(500)	Int(11)

A.8 – Cross Browser Testing

Internet Explorer	Safari	FireFox	Google Chrome
All functionality requirements work. Only issue is a couple of CSS issues.	All functionality requirements work.	All functionality requirements work. Only issue is a couple of CSS issues.	All functionality requirements work.

A.9 – Responsive Testing

Models	Usability	Layout	Major Issues
iPhone 5 (portrait)	Unable to click on icon, screen moves	Slight adjustments to be made on homepage and map	Icons need to be made clickable.
iPhone 6 (Portrait)			
iPad (Portrait)			

Nexus 4 (Portrait)	when drawing on map. Can't see images within info box and account.	need to be before search form.	Can't see image slideshow. Screen moves when drawing on map.
iPhone 5 (Landscape)	Unable to click on icon, screen moves when drawing on map.		Icons need to be made clickable. Screen moves when drawing on map.
iPhone 6 (Landscape)			
iPad (Landscape)			
Nexus 4 (Landscape)			