Niceway.to
Crowd Sourced Scenic Route Sharing

Submitted March 2016 in partial fulfilment of the conditions of the award of the degree MSci (Hons) Computer Science

Craig Knott
4159378
psyck

With Supervision from Max L. Wilson

School of Computer Science and Information Technology
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____/_____/_____

Abstract

Project abstract

Words in text | 1,604

Calculated with the TeXCount web service
`http://app.uio.no/ifi/texcount/online.php`

Contents

1 Introduction

In recent years, the technologies behind satellite navigation and routing services have greatly advanced, allowing them to produce the quickest route between two points in only a few seconds, even with technological limits [11]. As a result, travelling via car is quicker and easier than ever before, and many drivers are now focused solely on reaching their destination as quickly as possible. This has resulted in a shift in the mindset of society where the scenery that we pass on the road is simply a buffer between segments of our day, and its beauty is left unappreciated. This mentality promotes a culture of instant gratification, impatience, and self-involvement among the driving community, which has a huge detrimental effect on drivers, where any interruptions on their journey are cause for anger. In fact, it has been shown that since 1990, incidents of aggression during driving have risen 51% [22].

Some research has already been completed in an attempt to shift the focus of driving from simply travel, to also being an enjoyable recreational activity. This includes work such as recommending "nice" routes (determining this using social network data [17][21][18]) and how to personalise routes that have been deemed "efficient" [5]. Outside of the world of research, many services already exist that provide users with a collection of scenic routes between two locations. The purpose of these services is to encourage drivers to enjoy the experience of driving more, and be exposed to more of their surroundings. Examples of these include Google's "My Maps"[1], MADMAPS[2], and MyScenicDrives[3], which are discussed further in section 3.1.

Unfortunately, these systems have some flaws that mean they do not fully solve the above problem, and therefore have not made a huge impact. The largest two being the method of delivery, and the inability for users to contribute content. Google's "My Maps", and MyScenicDrives are optimised for desktop browsing and their support for mobile devices is limted. This is a large portion of users lost, considering that in the United Kingdom 33% of Internet users believe that smartphones are the most important device for going online[16]. MADMAPs, whilst mostly focusing on the selling of physical maps, does also provide a mobile application, but this is clunky, and poorly designed. Alongside these larger flaws, some other minor flaws are also present, including small user bases, primitive search functionality, slow and unresponsive webpages, and some services costing money.

Niceway.to has three main aims: to build a community of travel enthusiasts, improve the travelling experience of those that use it, and allow for users of all skill groups to access it. It will provide a way for users to discover scenic and visually interesting routes between two locations, all of which have been provided by other members of the community. These routes will each contain a social commentary, with users being able to rate them, comment on them, and share them (these will be incentives for users to provide quality content, and to remain loyal to the site). To address the problem of previous software systems, mainly the method of delivery, it will be built as a fully-responsive web-application that functions equally well on desktop and mobile devices.

As a final note, it should be mentioned that this project will not be focussing on the classification of whether or not routes are "nice" or "scenic". Instead, the content will be entirely user driven, with the assumption that they would only contribute routes that are interesting and visually appealing. To further encourage this, a rating system for routes will be implement, so that "better" routes are more visible than those deemed less so.

2 Motivation

Niceway.to is an application envisioned by my client, Matthew Pike, who works for a small start-up in China. The project has already been worked on once before, but the end result was not to a standard that my client was content with. As a result of this, he would like for the project to be redesigned and recreated from scratch, as little of the original software is reusable.

The main motivation of this project is to change how we think about driving: from simply a tool for travel, to an enjoyable recreational activity, where, instead of being focused solely on reaching our destination, we can take time to appreciate the beauty in the world around us. In a 1995 study, almost 90% of motorists had experienced some form of road rage within the preceding 12 months, and 60% admitted to losing their temper whilst behind the wheel[7]. To facilitate this proposed shift in mentality, this project will be a tool that offers a collection of user submitted scenic driving routes, with a heavy focus on utilising the social characteristics of the Internet. The idea being that when a user wishes to travel somewhere, they could do so by travelling a route lesser known to them. This route may be slower, but would make up for the time lost, by providing a visually enriching experience that the driver would have otherwise been deprived of, and to help avoid common annoyances on the road. This would also help to relieve the monotony of driving the same, mundane, routes repeatedly - which has been shown to have serious implications in terms of accident causation[20].

A big part of this project, which my client has stressed, is to foster a community of travel enthusiasts. Specifically the ability to have a social commentary surrounding each route, where registered users of the application are able to express their opinions on the content, give a numeric rating for the content, as well as share the content (both internally and externally). The reason for this is because a user is far more likely to return to, and remain engaged with, a website if there are other users doing the same, especially if they are directly communicating with those other users[10]. Allowing users to submit their own content coaxes them into feeling more of a connection to the site, and will increase their chance of returning (so that they can check how well received their content is). It has been shown that feedback is a useful tool to boost engagement[15] and hopefully this will encourage users to associate the site with positive experiences, and inspire them to produce quality content (in order to receive more of this feedback).

In order for this community to thrive, it is vital that the system is simple to use, open to users of all skill groups, and easy to access. Therefore, it is important that HCI principles are kept in mind throughout every step of the design and implementation processes. Key to a good design, is simplicity. This is because complex user interfaces frustrate users, and can deter them from returning. Studies have shown that users lose more than 40% of their time to frustration, and in most cases the user ends up angry at themselves, angry at the computer, or left with a feeling of helplessness[9].

In addition to all of these reasons, I feel personally motivated to ensure that this project succeeds. As someone who does not currently drive, I find myself in need of others to drive me when public transport proves inadequate. As a passenger, I will often observe the driver becoming evermore agitated with other drivers on the road, and seeing this problem first hand helps enforce my feelings of the importance of solving it. Driving is a freedom, but one that is being squandered and perceived more as a chore than an enjoyable activity.

3 Background Information & Research

This section looks at some software systems that are already attempting to solve the problem identified in this report, as well as discussing some of the different technologies considered for the implementation of this project.

3.1 Existing Systems

As mentioned in section 1, software systems for the distribution and sharing of scenic routes already exist. Each of these systems approaches the problem in a different way, and thus all have their own advantages and disadvantages, which will be discussed here. The aim of this is to determine the best and worst features of these, so they can be incorporate or avoided.

Google's "My Maps", `https://www.google.com/maps/d`
Google's "My Maps" service (distinct from "Google Maps") is a tool that offers users the ability to plot maps between locations, and save them to their Google accounts. This allows users to quickly access routes they travel frequently, as well share those routes with others (over various social media platforms). The main advantages of this service are that it provides a graphical tool for visualising routes, the ability to add photos and videos to specific waypoints, and, as mentioned above, the ability to share routes via social media. Another interesting feature that is provided is the ability to plot all the points of a route, and generate the path afterwards, rather than generating it after each point is placed. This should be taken into consideration for a mobile application, because users may wish to save on data (although if users are unaware of this being the reason, it just makes the tool look less responsive).

The key disadvantage of Google's "My Maps" is that it is very slow, and there are long periods of loading in-between successive actions. Responsiveness on websites is key, otherwise users are unaware if their actions are having an impact or not. There are also other disadvantages, including the unintuitive and cluttered user interface, the small user base (being a relatively unknown piece of software, dwarfed by the Google Maps service), and that the only way of sharing your routes is to other social media platforms. This last point is particularly important to address for Niceway.to, because it aims to build a community of users. If they can only share their routes to *other* platforms, the community will have less chance of thriving. This is why all routes on Niceway.to will be accessible and searchable from within the system itself, and the sharing of routes will simply be a tool to draw more people to the site.

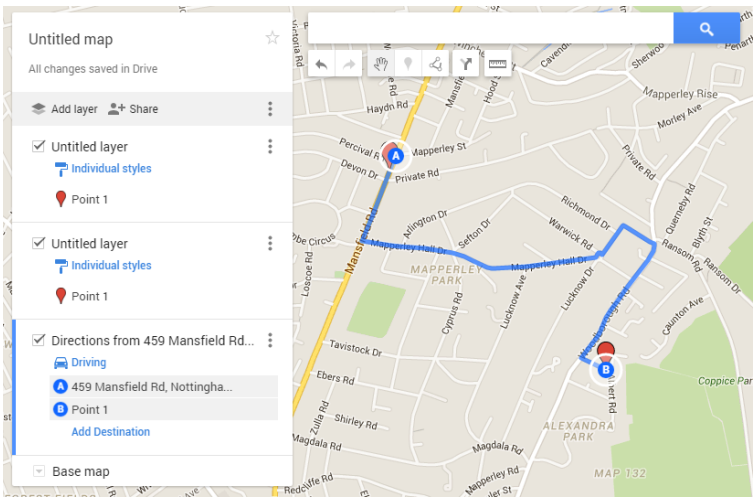


Figure 1: Google's "My Maps" route creation/editing feature

MyScenicDrives, `https://www.myscenicdrives.com`

MyScenicDrives is a website that allows users to search for scenic routes by city, state or zip code (currently the service is only available in the United States), and view extremely detailed information about these routes. This includes a very lengthy explanation of all the things can be seen and done on the journey, interesting facts about the locations visited, all the roads that will be driven on, the best seasons to take the journey during, and even which service stations will be passed. This extremely rich content is the main selling of MyScenicDrives.

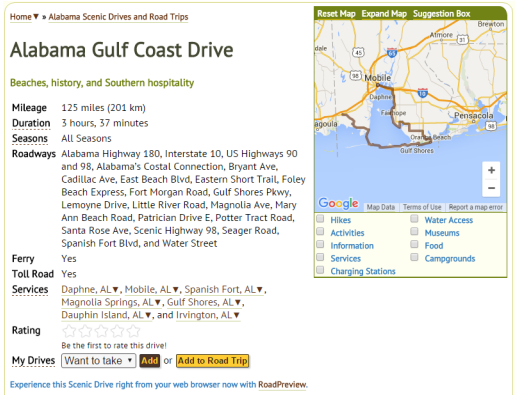


Figure 2: MyScenicDrive's route details

However, no matter how high the quality of this content it, there is still a glaring problem with MyScenicDrives: the quantity of routes. This may be due to the huge amount of information required for a route to be accepted onto the site, but essentially makes the entire application useless. As an example, a search for all routes in the state of Alabama returned a single result. In addition to this, the search functionality itself is very primitive. There is no option for users to select where they wish to start or end their journey, and instead they can only pick a large geographic region. Which would not be suitable for Niceway.to, which aims to provide a way for users to find routes between specific locations.

Mad Maps, `http://madmaps.net/`

The final mapping website that was investigated was Mad Maps, which differed from the others in that it allowed users to purchase physical maps, which they could also view on their mobile phones. One useful feature of the service was the ability to download routes directly to a mobile device, so that the user's Internet connection did not become a limiting factor during a journey. However, it is difficult to ignore the biggest down fall of this service, which was the cost associated with it. All of the maps had a price associated with them, and there was no ability to preview the maps before purchasing. This blind investment could be off putting for many users, as they may be unsure if they would even enjoy the routes provided. Further to this, the mobile application both required an upfront payment to download, as well as containing adverts within it: which seems unjustified when the majority of the Mad Maps applications had very poor reviews.

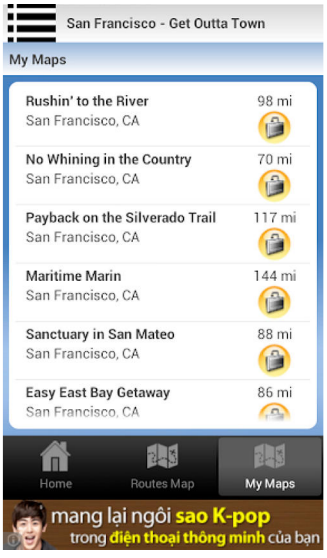


Figure 3: List of Mad Maps routes

The service works by having a group of "experts" compile the maps, and distribute them to the users. This is supposed to instil confidence of their quality, but instead takes out a huge part of travelling, which is the social experience. The only social interaction that users can have with Mad Maps is the ability to upload photos of waypoints that they have visited, but these will then only be seen by other users that have purchased the route, and would not help to foster the community that Niceway.to is striving for.

3.2 Platforms and Tools

This section introduces some potential platforms and tools that could have been used in the project, along with justifications for and against them. The system consisted of the back end and server technologies, as well as some user-friendly front end. Tools and frameworks and languages for both of these parts were evaluated to determine which would be the most appropriate. A full list of the final decisions of tools to use, including their justifications, can be found in section 6.1.

3.2.1 Native Mobile Applications VS Responsive Web Applications

Before investigating technologies to use, it was important to determine what kind of application was to be developed, as this would radically change the tools required. Ultimately, it was decided that it would be better to build Niceway.to as a responsive web-application, but the advantages and disadvantages of both approaches have been discussed below.

Native Mobile Application
Native mobile applications are applications that are downloaded onto a mobile device, and run directly on the hardware. These applications generally have greater exposure, because they are distributed through the application marketplace for the given operating system, and can be reviewed and rated by users. There are several advantages to developing a native mobile application including: the ability to implement multiple pricing models (payment for download, in-application purchases, free with advertisements, or entirely free), after the initial download they can be used without an Internet connection (as data can be downloaded to the phone and accessed later), and the native technology and hardware of the phone can be utilised to provide a better experience for the user.

However, native applications do also have some drawbacks. The most prominent is the number of different operating systems available, which mean that any application created needs to be rewritten multiple times in different languages, to ensure that it can target all devices. This is a huge investment of time and resources, especially as some platforms do not have a large user base (and therefore this effort would potentially be wasted). Native applications must also be downloaded onto the user's device, which requires a commitment from the user, and their on-going desire to keep the application on their phone. Mobile phone users can be fickle, and delete the application at any time for any number of reasons.

Responsive Web Application
A responsive web application is a website that can function both on desktop devices, and mobile devices by scaling the elements on display. They are incredibly versatile because they run in a web browser, which means that they target all possible devices without the need to rewrite the code base in several languages (some tweaks for certain browsers may be required, but this is usually a small amount of work). They are written in the default web languages of HTML, CSS and JavaScript, and the back end can be whichever language the developers are most comfortable with, which makes it very easy for developers of any skill level to work on them. Another advantage of them being hosted on a server online is that it becomes very easy to release updates, because they are instantaneous, and the users do not have to download anything.

However, the Internet is a large place, and without a centralised place to advertise the application, it is possible it will never be discovered by many potential users. Alongside this, web applications require a constant Internet connection to access them, which could be a problem for users that do not have a large data allowance on their phone.

3.2.2 Back End Tools

In this section several well known back end frameworks utilising different programming languages are discussed, with the advantages and disadvantages of each identified.

The Zend Framework (PHP)
The Zend Framework[1] is an open source MVC framework for developing web applications using PHP. It has been around for a long time, and therefore has extensive documentation, and well defined standards and guidelines. This helps all developers using Zend to be able to instantly recognise and understand what specific parts of code are doing. It is a fully object oriented framework, which means that all classes can be extended and an inheritance model can be utilised.

The drawbacks of Zend are that is it very bloated, and can be a little slow, but the reason for this is the huge number of convenience classes that are provided by default (including form validation, user authentication, and many more), which can be removed as necessary to increase performance. In addition, this modularity means that Zend can easily be combined with other libraries, so further features can easily be added.

Ruby on Rails
Ruby on Rails[2] is an open source web application framework written in Ruby that implements the MVC framework. The advantage of Ruby code is that it is very readable and mostly self-documenting, which saves developers time whilst programming. However, this can also lead to laziness, and lack of comments, with developers assuming that what they have done is obvious. The modular design of Ruby on rails results in faster development of applications and flexibility, and there is a vast collection of open source libraries (known as gems) available within the Rails community.

The negatives of Ruby on rails are that it is difficult to find good documentation for it, and it is more resource intensive that some of the other frameworks discussed, making it slower to boot, and slower to run. In addition to this, Ruby is the only language of those discussed, that I do not personally know, which would have slowed down the initial development process by a large amount, and the intricacies of the framework would most likely not have been utilised effectively.

Javascript with Node.js
Node.js[3] is a JavaScript runtime which uses an event-driven, non-blocking I/O model, and is very lightweight and efficient. The advantages of using Node.js is that it is quick and easy to set up a basic server, and it is extremely popular, which means there is a huge amount of support available. This includes Node.js' package ecosystem, *npm*, which is the largest ecosystem of open source libraries in the world. It also means that the entire code base would be written in JavaScript, meaning external developers would only be required to know one language.

However, Node.js is not without it's downfalls. These being that it does not scale well to large applications, and quickly becomes confusing and difficult to manage. Along with this, there is no separation of model and controller, which means database internals "leak" out into applications.

[1] http://framework.zend.com/
[2] http://rubyonrails.org/
[3] https://nodejs.org/en/

3.2.3 Front End Tools

After deciding that the project would be a responsive web-application, the language choice was obvious: HTML, CSS and JavaScript. However, consideration was still necessary for the framework that would be used to create the interface.

Bootstrap
Bootstrap[4] is a very popular front end design framework produced by Twitter. It is fully responsive and scales easily and efficiently to a multitude of devices, include phones, tablets, and desktops. Due to it's popularity, there is a huge amount of support online, as well as extensive documentation provided by Twitter themselves. These things, combined with it's simple class names, make it extremely easy for beginners to pick it up, which is an advantage if external developers have never used it before. It can also be customised to only load the components that are required, which helps to reduce the overall size.

However, one of the biggest advantages of Bootstrap, it's huge user base, has proven to also be one of it's biggest downfalls. Due to the quantity of websites now using Bootstrap, a large number of websites look extremely similar, which detracts users and prevents them from standing out amongst the crowd. Many developers are specifically recommending to not use Bootstrap for this very reason[6], and thus the inclusion of Bootstrap in this project would require thought as to how to make the design stand out.

Foundation
The Foundation Framework[5] is, similar to Bootstrap, a fully responsive front end design framework. Unlike Bootstrap, and most other front end design frameworks, Foundation is "mobile-first": which means that everything is responsive by default, and assumes you are developing for mobile. This is useful for mobile-only applications, but it means that applications that also target desktops require more work to implement. As a result of this mobile-first approach, the grid system in Foundation is much more fleshed out than in other frameworks, and has much more functionality. Foundation also has a collection of built in extra tools, such as *Abide*, which can be used for simple form validation.

However, Foundation provides far fewer themes than Bootstrap, and due to it's smaller user base, there is a lot less support available from the community. In addition to this, the majority of the documentation provided is in video format, which makes finding information quickly more troublesome. Finally, Foundation is not supported by certain older browsers (like Internet Explorer 8), which restricts certain users from using the site.

Semantic
Semantic[6] is the final front end design framework that was investigated, and is the newest of the three. The main advantage of semantic is it's concise HTML and how classes use syntax from natural language, which makes writing code much more user friendly. It also extends the majority of interface elements through intuitive JavaScript "phrases", which can trigger different behaviours on those elements.

The disadvantages are that it is not as well known, and therefore has a far smaller user base than either Bootstrap or Foundation, which means the only real support comes from the documentation that has been provided (which, whilst good, isn't fully comprehensive). This also means that an external developer is less likely to know how to use it, and they may find it difficult to pick up.

[4] http://getbootstrap.com/
[5] http://foundation.zurb.com/
[6] http://semantic-ui.com/

4 System Specification

In this section, the functional and non-functional requirements of the system have been outlined. These were obtained by looking through the design specification that was provided at the start of the project, and were agreed upon by both parties.

4.1 Functional Requirements

1. The user should be able to search by geographic region and discover routes for that region.
2. The user should be able to contribute routes.
    2.1. Only the creating user should be able to modify these routes
    2.2. The user can make the route private
3. The user should be able to interact socially with the route, including:
    3.1. Commenting on public routes
    3.2. Recommending similar routes
    3.3. Sharing routes to external social media websites
4. Users should be able to create an account with basic information
5. There should be administrative users who have extra functionality, including:
    5.1. Deleting users
    5.2. Updating users
    5.3. Creating users
    5.4. Deleting routes
    5.5. Deleting comments
    5.6. Making announcements
    5.7. Making backups in a standard, free and open format
    5.8. De-authorizing active sessions
    5.9. Locking the site and preventing access
6. Users should be able to export their routes
7. Users should be able to make a copy of other user's routes and edit them
8. There should be a route editor component which allows the users to construct a route
9. Users should be able to log into their account and:
    9.1. Update personal information
    9.2. Access and edit their submitted routes

4.2 Non-Functional Requirements

Accessibility
The proposed system is to be made available entirely on-line, allowing anyone with Internet access to use the system. As the application will be web-based, users are not required to download anything before using it, which will make it more accessible, and easier to get started with the application. The only potential issue with a web-based application is if the server goes down, or if the domain expires. The server going down does not, unfortunately, have a solution, but keeping the domain should be trivial.

Usability and Operability
The project should be designed in such a way that users ranging from a low level of skill, to a high level of skill should be able to use it with little prior knowledge. The project will be developed as a fully-responsive web application, meaning mobile devices will be fully supported.

Maintainability & Documentation
The system must be well documented allowing for easy maintenance by an external developer. This includes both an easy to understand code structure, as well as commented code (specifically using PHPDoc, and a similar style in the JavaScript code), which should be kept as modular as possible.

Quality
As this system is to be used externally, and will be a representation of both myself and the client, there are several quality concerns that must be addressed. The system must be built so that it is robust, and works as the user expects, and contains as few bugs as possible. Any bugs that are identified should be reportable to the maintainer, and be fixed as soon as possible. The code must also be of a high quality, and therefore will be written in a modular way, with useful comments provided to highlight the purpose of each function, and illustrate any particularly complex code.

Resource Requirements and Constraints
As this system is aimed at users with a mixture of skill levels, no assumptions can be made on the level of hardware that the users will possess. For this reason, the system will be designed to use as few resources as possible. Fortunately, due to being a web-based system, the load of the system will be fairly minimal, as it is mostly loading JavaScript and HTML5. The only true computation takes place on the server, and thus would not be a concern for the user.

The loading and saving of files potentially causes a problem, but only if the user has very little hard drive space, and attempts to save an extremely large file. This is, however, unlikely, as even very large routes will have a relatively small file size.

The final concern is Internet bandwidth which, whilst not a problem on desktops, will be a problem for mobile phones. This is why the amount of data sent to the user when they are navigating a route will be kept to a minimum, so that they do not use up their data allowance (or drain their battery).

Cross Platform Compatibility
Due to the project being a web-based system, it is difficult for it not to be cross platform compatible. The only real concern is cross-browser compatibility, so it is important that the system is tested on different browsers.

Security
Security is a concern for this project, as the users will be able to create accounts with the system, and therefore their data will need to be stored. This data will be stored in compliance with the Data Protection Act, and all passwords will be encrypted.

Disaster Recovery
The administrator should be able to take backups of the site in a standard, free and open format. They should also be able to de-authorize active sessions, and lock the website to prevent access. As far as the code base itself, the project will be stored both on the server, and in a Git repository, allowing for recovery if any problems occur.

5 System Designs

In this section, all of the design aspects of the system have been detailed and justified. This includes the design of the user interface itself, the navigation and expected user path, and the internal design of the application.

5.1 Navigation/Control Flow Design

In the final implementation of the system, there were seven main sections, which have been enumerated below with a brief explanation of what their main purpose is.

1. Route Discovery
   The landing page, which provides a large search box (allowing users to search for routes), and a description of the main functionality of the website.

2. Route Listing
   The search results page, which would list all of the results returned for a particular set of search terms, as well as some brief information about each of the routes.

3. Route Detail
   The individual route display page, which lists all the available information for a given route, as well as the social stream that users can interact with.

4. Route Creation
   The map interface that allows users to specify their own routes, as well as an interface for listing the points on the route.

5. Profile Page
   The user detail page, which allows registered users to view their own routes and their saved routes, edit their settings, and update the appearance of their avatar.

6. Admin Page
   The administrator only page which gives them access to various administrative tools, as well as the ability to look at and handle reported content.

7. Login/Signup Pages
   Pages that provided short forms allowing users to either sign up or log in to the system.

When designing a website, it's important to think about how the users of that website will traverse it, depending on their goals. For Niceway.to, there are two main user groups to consider: those with accounts, and those without accounts. Both of these groups will use the site in very different ways: those users without accounts will use the site simply for the searching of routes, but those *with* accounts will be much more involved in the social aspects of the site and sharing their own content.

The path that users *without* accounts would usually take, as shown in figure 4, would start with the landing page. They would then follow a rather linear path through the system, as they are only given the choice to advance to the next page in the sequence (or back if they so wish). This starts with searching for a route, selecting a route from the search results, viewing the detail page for this route, and then potentially looking at the route in "full-view" (where we see the route displayed in a map filling the entire screen). At any point in this sequence, the user is free to jump to the sign up page, a link to which is located in the top right hand corner of every page.

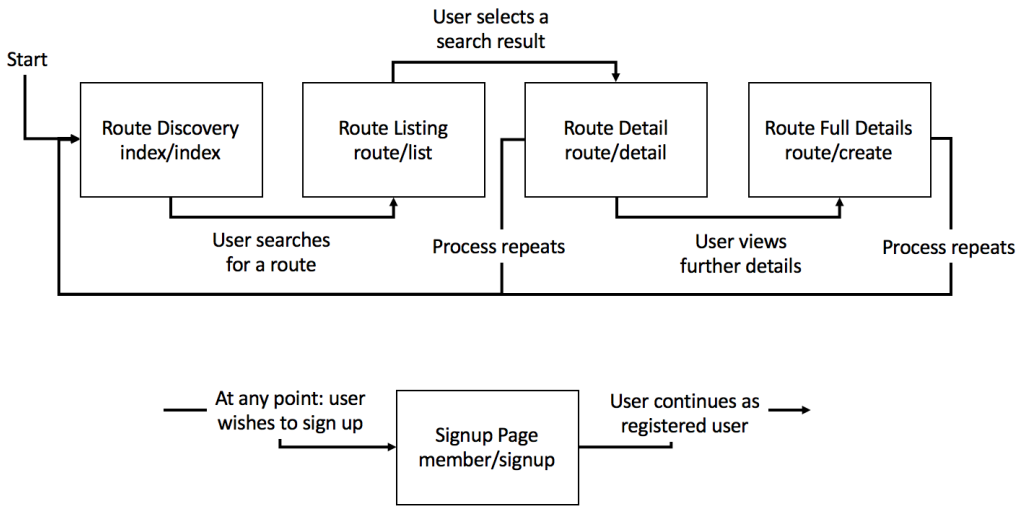


Figure 4: Expected path of non-registered users

For a registered user, we can expect a much less linear path, due to the larger quantity of actions they have available to them, this path is shown in figure 5. Unlike the non-registered users, there's no requirement for these users to start on the landing page, as it can be assumed that they would have book marked specific pages, and would log in to access them. After logging in, it is expected the user would spend some time on their profile page, and then use this page as the springboard to one of three main available actions: looking at their own routes or routes they are interested in, creating a new route, or following the same route search flow as non-users would.

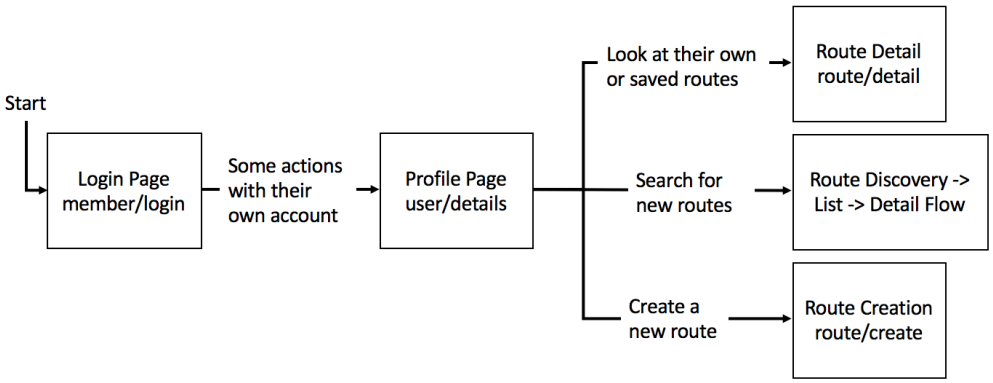


Figure 5: Expected path of registered users

Of course, this is just the generally expected path, and users are in no way restricted to what they can do (with the exception of accessing the search results page without having performed a search). The user can access any page of the site from any other page, through use of the navigation bar present on every page (shown in figure 6), which provides links to the search, creation, profile, and administrator sections of the website. This navigation helps to promote freedom and a sense of control within the system. This means that even when users make mistakes, they should feel like they know how to go back and rectify these mistakes, rather than being stuck in a state they do not wish to be in.

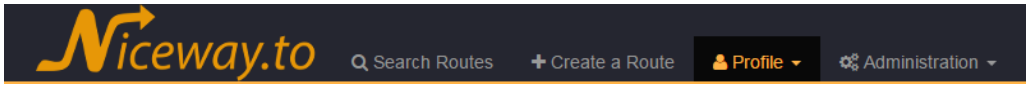
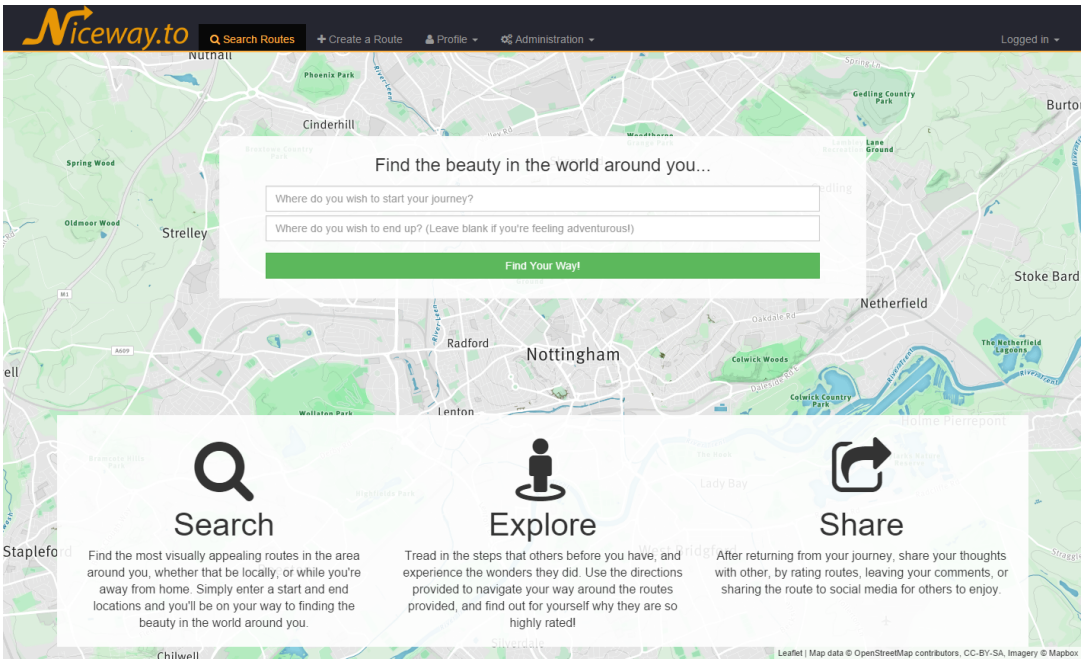


Figure 6: The navigation bar displayed to registered users

5.2 User Interface Design

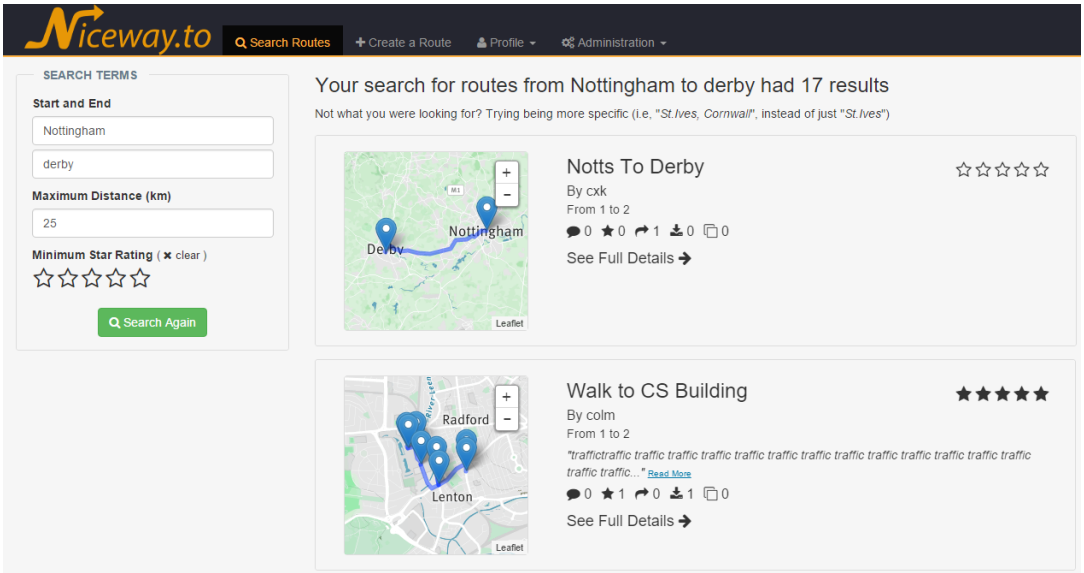
In this section, the interfaces for each of the main pages of the application have been displayed, along with justifications for why they were designed the way they were. Designs for each of these pages were produced at the beginning of the project and shared with the client. The design presents here are based on his feedback, and a combination of the best features from all the other designs. The initial designs can be found in appendix A.

Route Discovery Page/Landing Page



The design of the landing page was as simple as possible, with a large search box in the centre of the page (similar to Google), and a small infographic describing the purpose of the website. The prominent search box promotes the main functionality of the website, and makes it extremely easy for users to get started on their journey, without them being distracted by superfluous extra details. The page aims to look attractive to the user, so they are more compelled to use the site, rather than being put off, and taking their interest elsewhere.

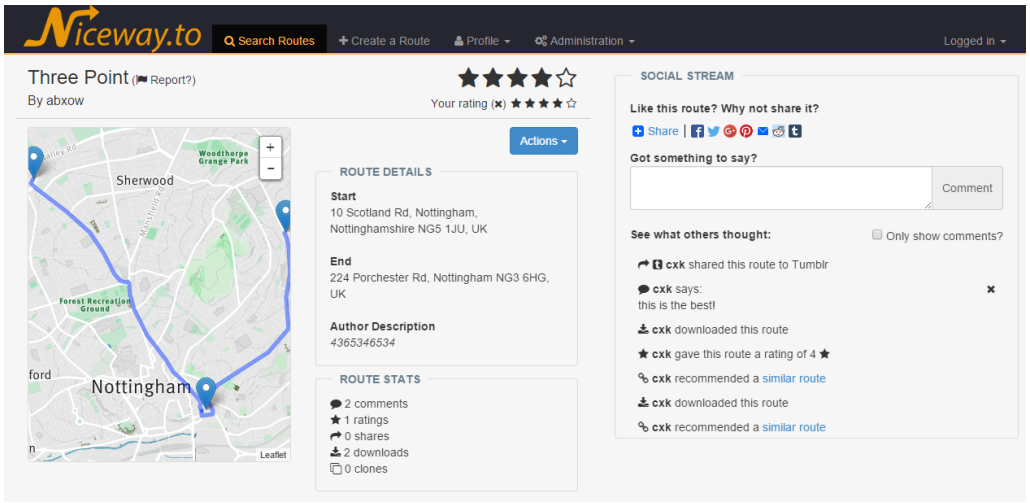
Route Listing Page



The route listing page was used to describe the key details of the routes that matched the user's search terms. During the design stage, two main approaches were considered: displaying the routes on a map, or displaying them individually. The problem with the former approach would be when there were a large number of results returned, which would cause the map to look cluttered, and it would be difficult for users to distinguish between routes. The listing approach allows each route to be considered individually, as well as being put in some kind of order (which is where the rating system is utilised).

A form on the left hand side (as well as the large title next to it) is present so that users can see the terms they searched for, and make any amendments necessary if there were mistakes (it is placed on the left because western cultures are more likely to look at the top left of a page first[12], which allows users to identify their errors earlier). It also allows them to further refine their search, with a minimum star rating, and maximum distance from the entered points. The key points of each route are displayed on this page, so users can make a decision as to which ones they which to investigate further. This includes the name and description, as well as social ranking, such as the rating and number of social interactions made on that route. The idea being that a user should very quickly be able to decide if a route is worth visiting or not, so they do not find they are wasting their time looking at poor quality routes.

Route Detail Page

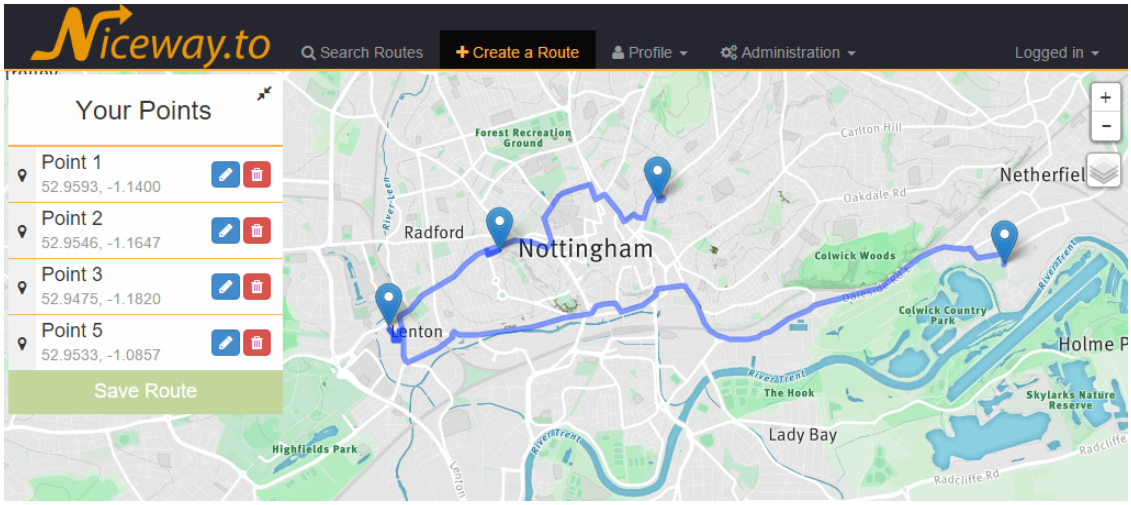


The route detail page was the page used to list all information about one specific route - for this reason, there was a lot of potential for this page to become cluttered. To combat this, the page is clearly divided into two main segments, which are then also further divided. These main sections were the route details on the left, and the route social stream on the right. This clear separation and hierarchical structure to the page made it very easy for users to locate specific information, by slowly narrowing down their search to different areas of the page.

The details section of the page contained a map of the route, key information about it (both administrative information like its name, and practical information like where it starts), as well as social statistics, like the number of comments and shares (allowing users to see a numeric representation of the popularity of the routes). This meant that users could garner all the information they needed to actually experience the route, as well as see the route they would be following. Being on the left, it is likely users will look at this section first, which means they can quickly determine if they have navigated to the wrong route.

On the right was the social stream, which displayed an amalgamation of all user social interaction with the route. This included comments, shares, downloads, and all other social interactions available. The purpose of this was to give a complete picture of the all interaction with the route, so that users could fully immerse themselves in it, and see what the community thought. It is very prominently displayed, so that more users are likely to notice it, and are more likely to interact with it (a point that my client stressed was extremely important). The social stream was visible both to users that were logged in, and those that weren't. This meant that registered users could easily share their opinions and interact with the route, and non registered users were more convinced to register, so they can could join in with the discussion.

Route Creation Page

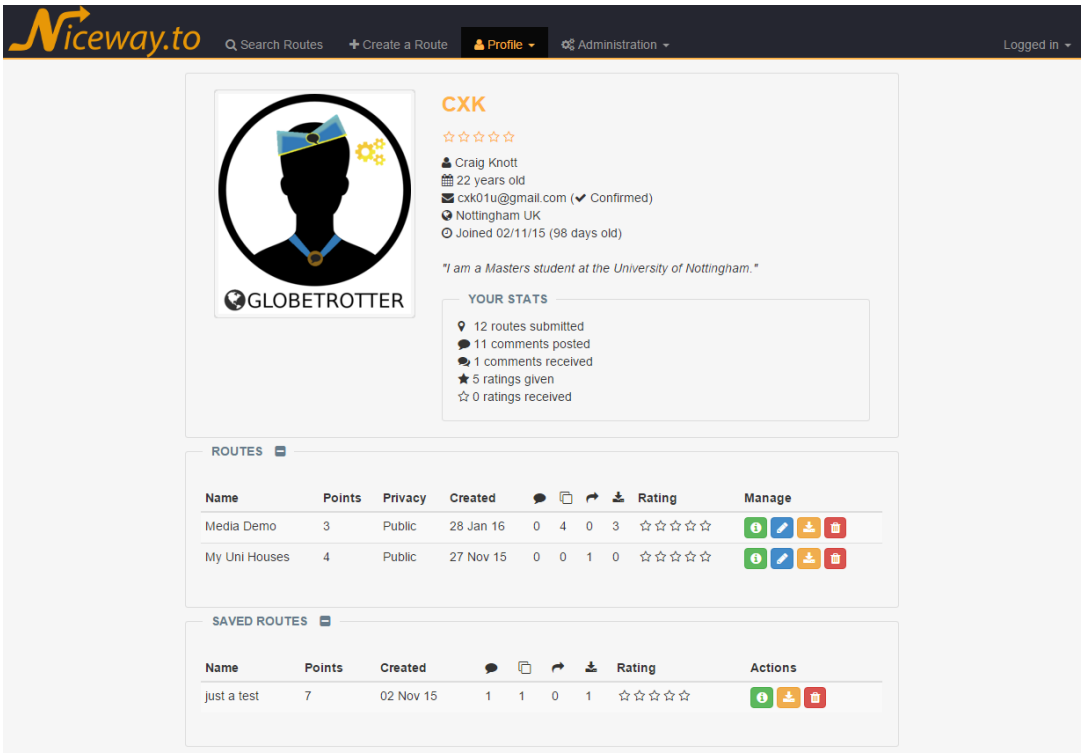


The route creation page was the page that would be used to actually construct and edit routes. It has a very simple interface that consists of a large map and a list of points on that map. This meant that the user could focus on the primary objective of the page, which was the creation of their route, without being distracted by other, useless interface elements. The list on the left served as a tool for users to track what they had done, easily jump to specific points, and update them. This list could also be collapsed, to take up even less space, to further reduce clutter and allow users to see their maps unobscured.

To add points, users only had to click on the map, which meant that using the page was very simple and intuitive. As points are added to the map, a route is drawn between them, during which time a popup is displayed so the user is aware of this. The advantage of drawing the map after each point, is that the user can see how their route is forming, and can make adjustments on the fly, rather than getting to the end, generating the route and realising they'd made a huge mistake.

When the user is done with their route, they can click the save button to bring up a modal window, allowing them to enter in the name and description of the route. Having this in a modal window helps to further eliminate clutter on the page, as it is only displayed when the user specifically requests it.

Profile Page



The final major page of the website is the user profile page, which displays information about a specific user, a list of their routes, and a list of their saved routes. The purpose of this page was to allow users to express themselves to other members of the community, and have somewhere to call "home". It displays their public and private details, like their customisable profile image, email, account age, and location, as well as statistical information, like the number of comments their routes have received. This meant that other users could quickly build up an idea of what this user is like, and how they interact with the application.

This page also displayed all of the routes that the user had created, which served a different purpose depending on whether you were viewing your own account page or another user. If viewing your own account page, it served as a central hub to administer and manage all the routes that you had produced, by listing them all, and providing functionality to view them, edit them, or delete them. If viewing another user's profile all of their public routes would be on show. From here, users can view, clone, or download the routes, but not actually make any changes to them. This meant that users could look at the profile of a content producer they enjoyed the work of, and look at other work they had produced.

5.2.1 Heuristic Evaluation of the Interface

In this section, the user interface is evaluated using several key metrics. These include the general purpose usability heuristics identified by Jakob Nielsen[14], and the golden rules of interface design identified by Ben Schneiderman[19]. These have been grouped together, with the specific heuristics being labelled in brackets (JN for Jakob, and BS for Ben), a full listing of all the heuristics referenced in this section can be found in appendix B.

Visibility and Feedback (JN1, BS3)
Users should always be aware of the impact of their actions. This is why any user interaction will have some visible feedback to the user, which will usually either be the updating of the interface (if something is added or removed), or popup modal windows informing users of what has happened, or what is about to happen.

User should be in full control (JN3, JN7, BS4, BS7, BS8)
Users should always be in charge of their experience of the website. This is why they are given full freedom to navigate anywhere within the system (using the navigation bar), and to decide what actions they wish to perform. At no point is the user forced to do anything they have not explicitly requested.

Consistency (JN4, BS1)
Consistency is important within a system so users can become accustomed to how things work, and what they mean. This is especially important when users are accessing new features for the first time, because they can use their previous knowledge to help them understand the new functionality. This is why colours and icons will be used heavily throughout Niceway.to. Green colours will be used to resemble positive actions (like adding or accepting), and red will be used for negative (deleting or cancelling). Icons for different aspects of the system (like cloning routes, or downloading routes) will be implemented and kept consistent so that users will associate specific icons with specific actions, and they will easily be able to locate and achieve their goals.

Error prevention and Recovery (JN5, JN9, BS5, BS6)
It is important that users do not feel punished for making mistakes, and do not get trapped in error states, unable to return. For this reason, extensive error prevention will be implemented, including validation on all forms, confirmations on "dangerous" actions, and useful error messages being provided when mistakes are made. The language of these messages is extremely important, as to not make the user feel like they have done something wrong. Instead, users should be instructed on how to resolve the problems they have encountered.

Reduce Cognitive Load (JN6, JN10, BS8)
A simplistic design is vital, otherwise users will be encumbered by the sheer quantity of content (it has been shown that humans can only truly cope with between 5 and 9 things at a time[13]). If there is too much on the screen at once, users will be confused and unable to focus. This is why only relevant and useful data will be displayed to the user, and interfaces will be as minimalistic as possible. Another way to reduce the cognitive load on the user is to remember data for them. This principle will be applied on the search results page, where the search terms are displayed back to the user, allowing them to be dropped from the user's short term memory.

Match between system and the real world (JN2)
The system should use language and imagery that is common and well known, rather than specific to the application. This helps the user's general understanding of the system, as they can relate it to other systems they have used. One example of this in practice is the use of the word "clone", for the forking of routes. Forking is a term well known in the field of computer science, but many average users would not understand it, hence this simplification. This is also the reason that icons are used throughout the system, as many users will recognise them and will understand the system better with their inclusion.

5.3 Internal Design

As well as the front end, the inner workings of the back end of the system also needed some design. At this point during the project, the Zend framework had been picked as the back end framework (for reasons discussed in section 6.1), which enforces a Model-View-Controller design pattern. The advantages of the MVC framework are that it is simple to understand and use, provides separation of the accessing, processing and display of data, and is extremely modular. The diagram in figure 7 shows the interaction that users would have with the front end and back end of the system. It begins with a user on some web-browser making a request for a specific page. The browser then talks to the Controller (firstly the base controller, and then the specific controller for that page), which usually requests some data from the database. This access is abstracted, and is facilitated through the Model and various factories of the system, which actually access the data, and returns the results as a PHP object. The controller then passes this data on the View, which displays the user interface and is served to the client for them to interact with.

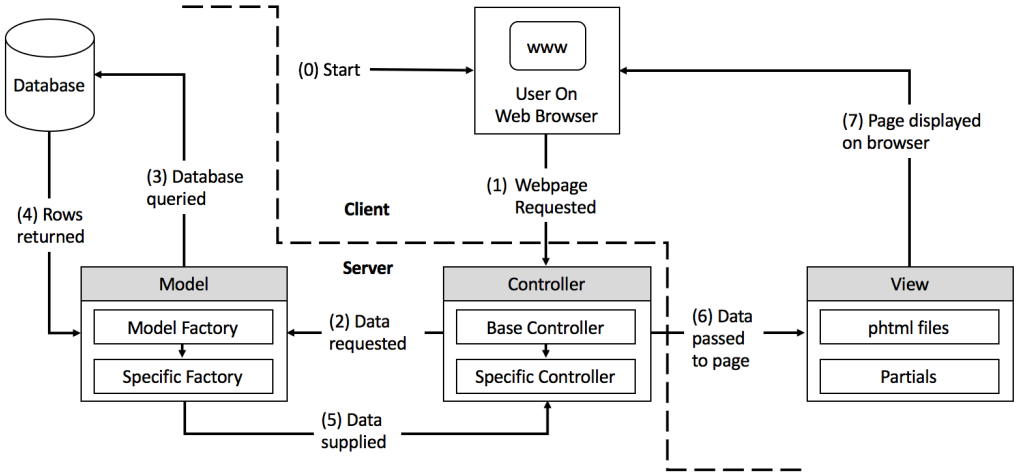


Figure 7: The internal design of Niceway.to

6 Software Implementation

6.1 Key Implementation Decisions

In this section, the tools used for the implementation for this project have been detailed, and the reasons for their adoption have been justified.

6.1.1 Implementation Platform - Responsive Web Application

Responsive web applications have many advantages that make them perfect for this project. Firstly, they work both on desktop devices, and on mobile devices. This means that the application can be built one, and it will work (almost automatically) on both platforms, and therefore the user is not restricted to using only one. Further to this, a web application is ran through a web browser, and therefore there is no need to compensate for different operating systems or environments, as the application should theoretically work on them all. Another reason for picking this medium is that they are written in the default web languages, and the backend can be any language desired, meaning that external developers are much more likely to be able to pick it up and begin working on it (rather than them being required to have specific knowledge on how to create Android or iOS applications). The final major advantage is that they do not need to be downloaded, which means the user doesn't need to make a commitment to the application, and updates can be released instantaneously, and with ease.

6.1.2 Back End Tool - The Zend Framework (PHP)

The back end tool of choice was the Zend Framework, which is written in PHP. This framework was picked because of its maturity, which includes its extensive documentation, online support, and well defined standards. This means an external developer can easily interpret the code produced, and can find help online if necessary. The Zend Framework also defines a very strict standard in how code should be organised and managed which is very intuitive to pick up, and helps split up large projects into small segments which are much more manageable. In addition to this, the development of a Zend application can be heavily sped up with the use of its many convenience classes and extra features, such as form generation and validation, and authentication. This also has the advantage that it is well documented by Zend themselves, and help is easily accessible, which will help future developers to understand and debug the code.

6.1.3 Front End Tool - Twitter's Bootstrap

Twitter's Bootstrap is extremely well known and extremely well documented. This makes it very simple for new users to understand how it works, and how to start using it themselves. It effortlessly creates aesthetically pleasing websites, with little work required on the part of the developer, which makes it perfect for a project like this. In addition to this, it automatically resizes to different screen sizes and display resolutions, which is a necessity for a responsive web application.

One important factor that was taken into consideration when designing the site was how to make it stand out from other websites developed with Bootstrap. The main step was to use a flat design with special orange highlights to give the site a unique defining feature and colour scheme, which helps to make it distinguishable from other Bootstrap websites. In addition to this, other libraries were also used, like the jquery-confirm[7] plugin, instead of some of Bootstrap's default features (like modals), to further help it stand out.

[7] `http://craftpip.github.io/jquery-confirm/`

6.2 Implementation Methodology

For a project of this size, it was a necessity to define some standards for how work would be completed. I decided to use an agile approach, utilising a Kanban board, and to work with the tips laid out by Henrik Kniberg[8]. The main reason for this is that it allowed for quick pivoting of focus (when bugs arose), I had a lot of experience with it already (having used it for my last large project) and found I worked well under its regime. At the beginning of the project, the list of requirements was generated from the specification sent by the client. This list was then broken down into different core components of the site (route creation, route discovery, etc), and then each individual task broken down into subtasks. Each of these subtasks was then put on a digital Kanban board, using a service called Trello[8] (combined with the extension Scrum for Trello[9] to add extra scrum functionality, like time estimations), to keep track of their progress, and for scheduling. The main advantage of splitting each task into many smaller subtasks, is that it meant it was easy to make progress on the project every day, as these tasks did not (individually) take a long time to complete. It also meant that no tasks were ever forgotten about, and bugs could easily be recorded and resolved. It has been shown that it is easier to reach goals if they have been explicitly written and expressed[23], which is why this Kanban method was so successful, especially as this could be shared with my project supervisor.

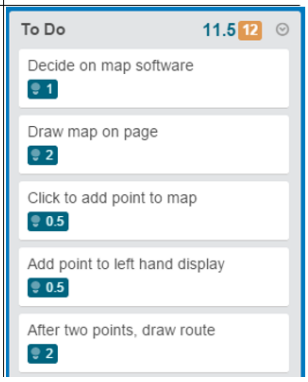
Requirement	Subtasks	Trello Representation
FR.8 There should be a route editor component which allows the users to construct a route	• Drawing of map • Ability to add points • Ability to connect points • Ability to edit points • Ability to save map • Load previous map	

Figure 8: Process for breaking down tasks, adding extra granularity with each step

At the beginning of each week (where a week started on the day of project supervision meetings) the total list of tasks to be completed was evaluated. Based on priorities, pre-requisites, and the Gantt chart produced for the project proposal (displayed in appendix C), a group of tasks would be selected for completion that week. Each task was given an estimated time of completion which would be used in conjunction with the number of work hours completed in the previous week to decide how many tasks should be picked. This meant that a constant stream of work was completed every week, and that the project did not fall behind. Having weekly meetings was extremely useful, as they were a chance to course-correct if things weren't going to plan, to receive feedback on the progress of the project, and to get an outside perspective on how new features looked and behaved.

[8] `http://www.trello.com`
[9] `https://q42.com/projects`

6.3 Problems Encountered

Assuming that a large project could be completed without any problems or issues along the way would be foolish, and this project was certainly no different. It is, therefore, extremely important to identify what these problems are, what the causes were, and how these can be prevented in the future. The biggest problem faced during the implementation of Niceway.to, was the lack of a proper testing framework. As new features were added, they were individually tested, and some minor integration testing was carried out with other parts of the system they may have had an impact on. After this was done, the feature was assumed to be bug free, and progress on the project continued. However, on several occasions, a new feature would be added, tested briefly, and marked as complete, only to have it be the root cause of some problem further down the line, due to some unforeseen interaction with another component. This meant that a lot of time was dedicated to going back to previous features and fixing them, instead of pushing the project forward. The solution to this would be to implement unit testing on every aspect of the system. This would mean, after the addition of any features, a set of tests could be run to ensure that all components of the system are still functioning as expected. Considering how well integrated unit tests are with the Zend framework, their lack of inclusion is this project is even more worrying. The main reason for this is how some work on the project was prototyped before its official commencement. This prototyped code did not implement testing, and was later directly included in the project. This meant that the speed of the initial stages of the project were accelerated, but meant that unit tests were left out, and with such a large code base now, it would troublesome to implement them at this point.

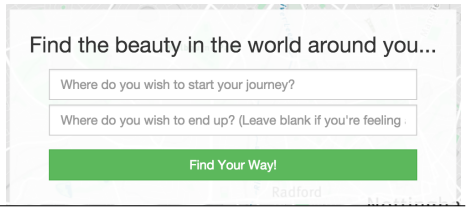
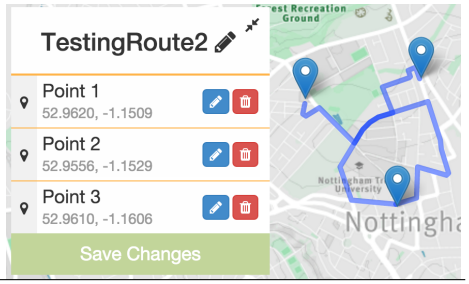
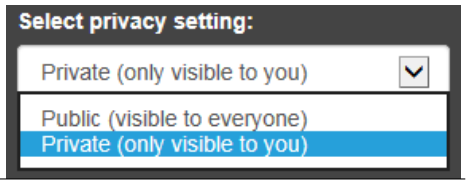
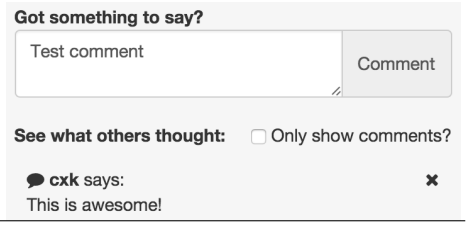
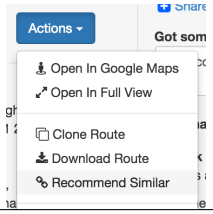
The next major problem encountered was the lack of proper research into an adequate routing service. When implementing the functionality to draw a route between two points, not much research was conducted into which routing service would be the best, and instead the easiest to implement was selected. This initial routing service was the MapBox default routing service, which was extremely easy to implement, because MapBox was being used to provide the map. After implementing the functionality to draw these routes, it became apparent that the service had a limit to the number of points it could route at any one time (a maximum of twelve), and that the service was unreliable (at times it would be unable to provide any route at all). This meant that a new service needed to be selected (the Leaflet Routing Machine), and that a lot of the functionality for routing needed to be rewritten with this new library. This wasted a lot of time, as work that was previous deemed complete needed to be completed a second time. This could have been resolved by expanding the research conducted at the beginning of the project, and being more thorough in which software was investigated (instead of just back end and front end tools, it would have been beneficial to look at libraries that were required as well).

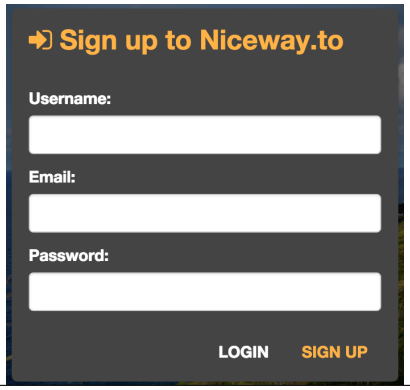
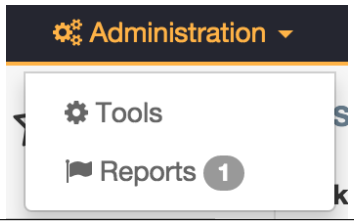
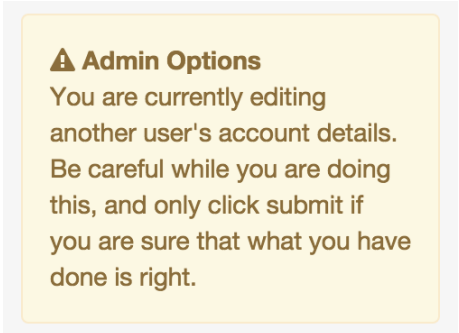
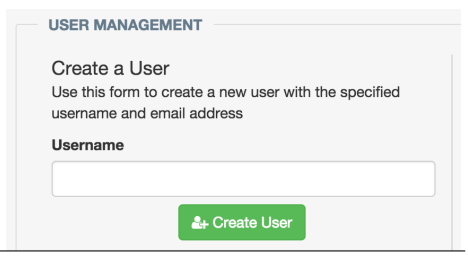
The third major problem was lack of attention to detail in the creation of the requirements. At the beginning of the project, the client provided a specification for the project, which included all features he required. This was then converted into a requirements specification, with functional and non-functional requirements. However, during this conversion, some of the intricate details of the initial specification were lost. This meant that, half way through the project, it was discovered that there was a large amount of functionality missing. Not only did this add extra pressure to the project, but also meant that many previous features had to be revisited and revised, which meant a lot of time had been wasted in the implementation procedure. In future, this can be avoided by paying much closer attention to the initial specification, and ensuring that all details are extracted as necessary, and getting the client to sign off on these extracted requirements.

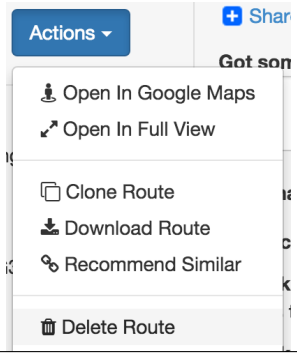
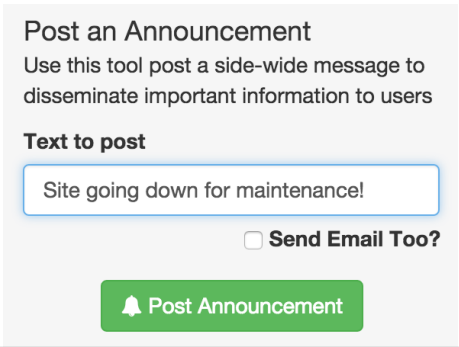
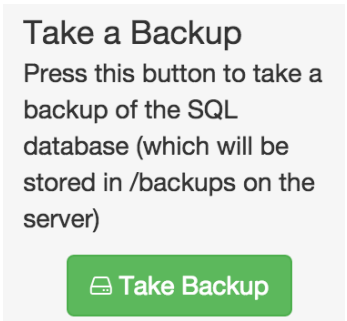
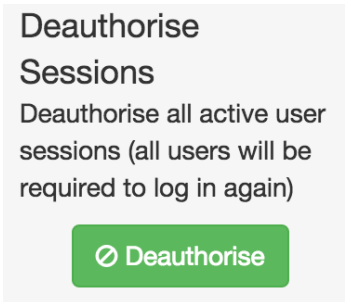
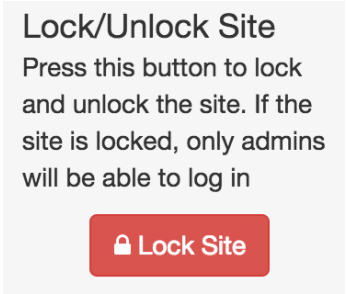
7 Testing of the Project

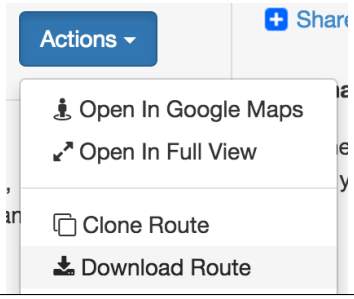
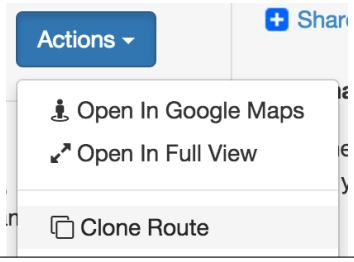
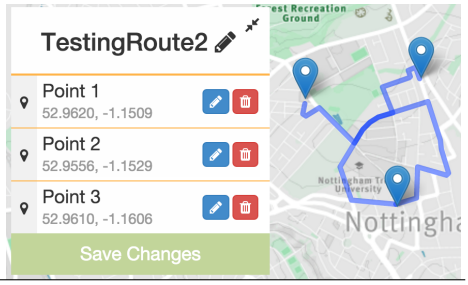
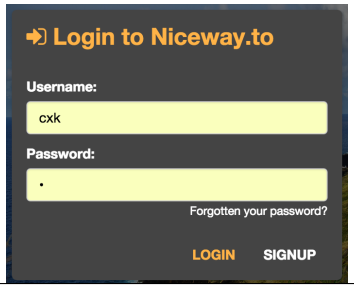
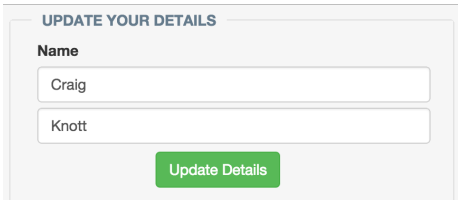
7.1 Functional Testing

In this section, each of the functional requirements laid out in section 4.1 have been evaluated and tested, to ensure that the system meets them. Due to these tests coming directly from the requirements, it was easy to determine if each test was successful or not, based on whether or not the feature had been included. In cases where the test is marked as "passed", a screen shot of evidence has been provided. This testing was only interested in whether or not specific features were present, and the inner workings of these features was irrelevant, for this reason, these can be considered black box tests[4] (testing where only the input and output can be observed).

FR#	Test	Result	Proof
1.	Users can search by geographical region	Passed	
2.	Users can contribute routes	Passed	See 2.1 and 2.2
2.1.	The creator of a route (only) can modify it	Passed	
2.2	Users can make their routes private	Passed	
3.	Users can interact socially with routes	Passed	See 3.1, 3.2 and 3.3
3.1.	Users can comment on routes	Passed	
3.2.	Users can recommend routes similar to the current route	Passed	

FR#	Test	Result	Proof
3.3.	Users can share routes externally	Passed	
4.	Users can create accounts	Passed	
5.	There are administrative users	Passed	
5.1.	Administrators can delete users	Passed	
5.2.	Administrators can update users	Passed	
5.3.	Administrators can create users	Passed	

FR#	Test	Result	Proof
5.4.	Administrators can delete routes	Passed	
5.5.	Administrators can delete user comments made on routes	Passed	
5.6.	Administrators can post messages that will be displayed on every page	Passed	
5.7.	Administrators can make backups	Passed	
5.8.	Administrators can close and deauthorize active sessions	Passed	
5.9.	Administrators can lock the site	Passed	

6.	Users can export/download their routes	Passed	
7.	Users can make a copy of another user's route	Passed	
8.	There is a route creation/editing component	Passed	
9.	Users can log into their account	Passed	
9.1.	Users can edit their personal information	Passed	
9.2.	Users can access and edit their routes	Passed	

7.2 Non-Functional Testing

In this section, each of the non-functional requirements defined in section 4.2 have been evaluated, and steps taken to ensure they were adhered to have been listed.

Accessibility
To ensure the service was accessible for the largest number of people, it was developed as a fully responsive web application, rather than a native mobile application. This meant that any users with an Internet connection could access the site, regardless of device. As a website, the project relied on the server to allow users to continually access it, for which there were two main concerns: the domain expiring, and the server going down. The solution to the first of these was setting up reminders to renew the domain when necessary, and the solution to the second would be to set up backup servers (although that was not within the scope of this project).

Usability and Operability
To ensure that all users, regardless of skill level, could use the system effectively, it was designed to be as leading and simple as possible. This meant having large buttons to draw the eye to certain key tasks, decorating pages with well known icons (so users could easily determine what the functionality of each item should be), ensuring that simple language was used throughout, and the interface free of clutter, or at least presented in an organised way. As mentioned above, the application was developed as a fully responsive web application, utilising Twitter's Bootstrap framework, therefore allowing users of any device to access the website.

Maintainability & Documentation
This requirement was partially met simply through the use of the Zend Framework, which enforces a Model View Controller design pattern. This meant that it was already modular, and the form was separate from the function. This, along with the extensive documentation and standards that have been produced for the Zend Framework, meant that understanding different segments of the code is simple.

In addition, every function and class has been commented with PHPDoc (or a JavaScript equivalent for the JavaScript code), and therefore functionality of each individual block of code is immediately obvious, and an external developer should be able to understand what each block does, and how to use it.

Quality
To ensure the software was of the highest quality, the standards defined above were strictly adhered to, including the use of Zend standards, and extensive commenting of code. To ensure that the quality of the product was equally as high, all new functionality was tested vigorously to ensure that it was free of bugs, and that it did not introduce bugs to other parts of the software. Further to this, the RITE method for identifying usability issues (which is discussed further in section 7.3) was employed with 15 individuals, to identify any potential problems with the interface.

Resource Requirements and Constraints
To ensure that the system was not a burden on the user's hardware, it was designed to do as much computation and calculation on the server side as possible. This meant that the server was responsible for doing the majority of the work, and the client would be spared. Even after page loads and the user did some interaction, an AJAX call would be made, which would get results, do computation, and then only have to quickly update the DOM of the page to reflect these changes.

The downloadable files (of routes) produced by the website were saved in JSON format, so that the signal-to-noise ratio (the amount of data used to describe the data itself) was kept to a minimum, meaning the size of the file was reduced. Even extremely large routes saved in this format would have minuscule file sizes (as a test, a route with 50 points was created, which only had a file size of 6KB).

The final consideration was the amount of data being sent to the user, and keeping this to a minimum. For this reason, all libraries in the project were minified (both JavaScript and CSS), to reduce the total data expenditure. After the project is released fully, the JavaScript and CSS of each page can also be minified to further reduce the size of each web page.

Cross Platform Compatibility
Due to the project being a website, it was inherently cross-platform compatible, because it is rendered within a web browser rather than a specific application or operating system. However, there are many intricacies within each web browser, and therefore it was necessary to test the core functionality of the application in each of the main web browsers: Google Chrome, Internet Explorer, Mozilla Firefox, and Apple Safari. Despite some graphical differences due to how each browser renders page differently, the core functionality worked across each of these browsers (which served to increase the size of the user base that could be targeted).

Security
As users were signing up for accounts for the system, they could provide some personally identifiable information. For this reason, both access to the web server, and access to the database running on the server required a correct user name and password. In addition to this, the passwords for all users were stored as an encrypted MD5 hash, to ensure that no member of the Niceway.to team could access a user's account unlawfully.

Some information that users can enter is also displayed on their profile page. For this reason, a disclaimer is included on the settings page, so that users can see what pieces of information will be publicly visible. This included: user name, age, length of membership, public routes, and location, but explicitly *not* email address. This meant that the user had the freedom to choose which pieces of information they shared, and which they would not.

Disaster Recovery
There were several methods employed to aid in disaster prevention and recovery for Niceway.to. Other than having access to the server, and the ability to use the UNIX system to run whatever commands were necessary, the administrators of the site also had several tools available within the application that they could use. The first of these was the ability to take backups of the site, which would save a copy of a mysqldump of the database to a directory on the web server, allowing the site to be restored to a previous state if necessary. The second tool was the ability to de-authorize all active sessions, meaning that any users would be required to log in again, before continuing to use the site (useful if hackers had infiltrated user accounts). The final tool was the ability to lock the site, which would prevent any users that were not administrative users from logging into the site, instead presenting them with a message informing them that Niceway.to was down for maintenance.

7.3 User Feedback Testing

Usability testing is the process of getting actual users to use the system, with these users performing a set of tasks whilst being observed. The time taken to complete the various tasks, and any comments or criticisms the user had were recorded, as well as any bugs the user encountered. The purpose of these tests was to discover usability problems in the interface, and what could be made simpler.

For the tests a sample of 15 users were selected, with a skill level ranging from low, to very high, and ages ranging from 21 to 46. The reason for this broad range was that users of different skill levels, and different ages, use and understand computers differently, and therefore what is considered intuitive for one user is not necessarily considered the same by others. This meant that the number of usability issues and simplifications that could be identified was greatly increased. The five tasks that the users were expected to complete are given below, and a full list of instruction can be found in appendix D.

1. Search for a route, and view its details
2. Create an account for the system
3. Comment, rate and download a route
4. Create a route
5. Navigate back to their route, and make edits

These five tasks represented the five core tasks that users of Niceway.to would be expected to engage in on a regular basis (except signing up, which would only occur once, but was a barrier for entry so it was important it was short and simple). This meant it was vital that they were easy to understand, and easy to complete by users of all skills levels. It was for this reason that the instructions for the tests were as vague as possible, as not to lead the users to the correct answers. Some of these tasks were purposefully extremely simple and short, so that users could really focus on the specific areas of the system, and therefore identify more issues.

The RITE method (Rapid Iterative Testing and Evaluation) was employed during the usability tests to quickly iterate on user feedback to fix issues within the system. This meant that, after each user completed the set of tasks, their feedback would be implemented, and any bugs they discovered would be fixed before the start of the next test. Each participant would therefore be looking at a slightly different product, and they would all be able to identify unique issues instead of all being focused on the same issues.

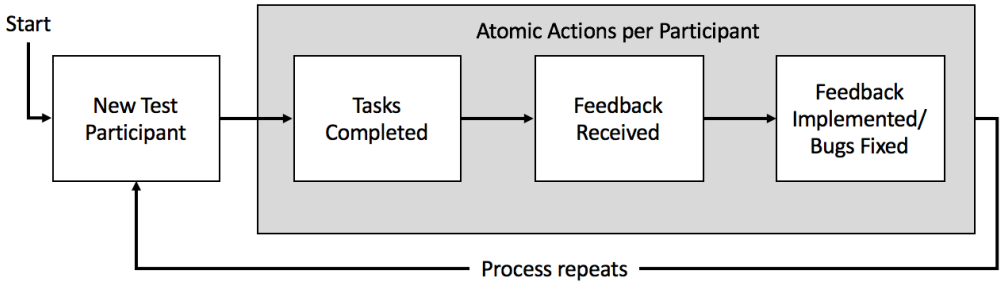


Figure 9: The employed RITE method life cycle

The time taken for each participants to complete the tasks can be found in appendix E, with a summary presented below, and an discussion of these results forthwith.

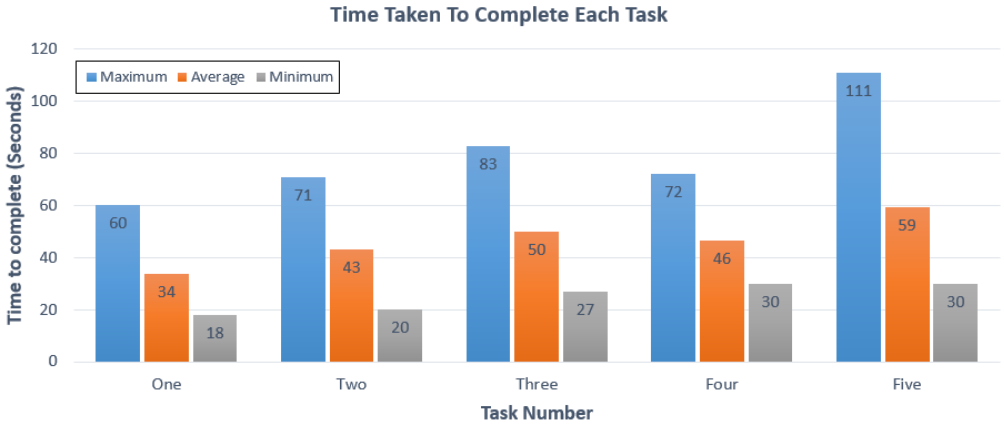


Figure 10: Summarised results from user testing

From these results it would not be far-fetched to claimed that each of these tasks were simple to achieve. They could all be completed in under two minutes by users of all skills levels, with an average completion time of just under a minute. Some expert users could even complete the tasks in under thirty seconds, which is promising, and is a figure that many users would eventually be able to replicate, as their usage of the site increased. A brief analysis of each of the tasks is given below.

Task One - Searching for Routes
This task was completed fastest of them all, which in part is due to users starting on the page they needed to use, but also that the functionality was easy to locate. This was because of the large search box in the middle of the page, which was eye catching and intuitive for users to understand. The main limiting factor in this test was not the software, but rather how quickly the user could type the search terms, which was the cause for the differing times. On the search results page, most users were almost instantly able to find the link to the details page, and complete the test unhindered. One user attempted to click on the title of the route, which he believed to be the "more intuitive thing to do", and thus this functionality was added after that user completed their testing.

Task Two - Creating an Account
Creating an account required the user to enter a user name, their email, and a password. This was therefore an extremely simple task, that most users performed with little hindrance. The main factors that drove the time up for this task were the waiting for the confirmation email (which varied based on the participants Internet connection speed), and errors in the form. The main error being users entering passwords that were too short (less than six characters), and their accounts being rejected as a result. This meant that users has to completed this step again, and spend some time trying to understand the problem. None of these, however, were points of discussion, as most users accepted this as standard, and didn't feel like it was a usability problem.

One issue that was not addressed in this process was what happens when a user attempted to sign up with a user name that already exists in the system. This is because there were so few accounts created, that the odds of two users picking the same were minuscule. If a user did happen to pick the same user name as someone else, it would likely result in a highly inflated time for this task, because the user would then be required to think of another unique user name, which can be difficult, as most users tend to use the same one for most websites.

Task Three - Interaction with a Route
The third task was used to determine how easy it was to experience the social elements of the system, which was an extremely important part of the system in regards to building a community around the website. In this task, users were required to search for a route and open it (building on knowledge they acquired in the first task), then perform three social interactions: commenting on the route, giving the route a rating, and downloading the route. Of these three, the only one that users struggled with was the downloading of the route, as many users could not find this option. It was clear that most users expected the download button to be somewhere in plain site, which was due to them being unaware of the other interactions they could have with the route, and thus assumed the download functionality be available just like commenting on routing. After the users discovered the action button, they quickly found the download feature, as well as a host of other features, which would then be able to use next time they were on the site.

Task Four - Creating a Route
It is worth pointing out that, despite the results for this test being very positive, there are not indicative of how a real user would use the route creation page. In the tests, users were simply asked to construct a route with three points, and save it. A real user would spend much longer in positioning these points, ensuring the route between them was good, and adding rich content, like titles, descriptions and media to the points. In the tests, most users clicked on three places at random, and hit the save button. It may have been beneficial to enforce the users to look for specific locations, which would have better reflected real users.

Regardless of this, a lot of useful feedback was gathered from these tests. The most prominent (which could have easily been missed as a result of the format of the test), was the difficultly of navigating large portions of the map at once. This was only discovered when one user that was not local to Nottingham, decided to create a route started at his house in Guildford. This required him to scroll the map for a considerable time to find this location. As a result of this feedback, the ability to search for specific locations, and centre the map on them, was added to the route creation page. This meant that users could search for a location and instantly jump to it, which was useful for when they wanted to create a route that started far away from their initial location.

Task Five - Editing your Route
The fifth task was the task that, on average, took the longest to complete, and had the greatest range of results (with a difference of 81 seconds between the fastest and slowest times). This is most probably caused by the users requiring some level of intuition to finish the task. In two of the previous tasks, the user was asked to use the search functionality, but in this task they were told to specifically avoid it. This meant the user needed to think about the system and utilise tools they had already used (the navigation bar), but apply them to a new problem - finding where their routes were saved.

After users determined that their routes were stored in the “My Profile” area of the site, locating the ability to edit them was very simple. The synonimity of a pencil icon, and the ability to edit is very strong, and many users (after locating the specific route), jumped to the edit page very shortly after. Perhaps changing the testing instructions to use the word “modify” or “change”, rather than “edit”, would have had an impact here.

An interesting thing to point out in task five, which was a merit of the RITE method, was how the editing of titles changed as a result of the tests conducted. One of the first users commented that having to click "Save" before being able to change the title was cumbersome. He suggested putting an edit icon next to the title, and allowing users to change this value - which would be more intuitive. This feedback was then implemented and it was observed that every single participant utilised this new feature, and thus sped up the time it took to complete that particular task

Issues Identified
Some of the main issues that were identified during the usability tests, and were then resolved in between testing stages, have been listed below. The advantage of resolving these issues in between tests is that subsequent participants were able to detect unique and novel issues, rather than each user identifying the same set of issues.

Page	**Comments Made**	**Resolution**
Landing	"Why can't I press enter to search?"	Allowed users to press the return key to submit their search results on the landing page.
Landing	"I accidentally searched without a starting point and got loads of errors"	Prevented users from submitting their search if no start point was provided.
Listing	"I thought that clicking on the title would have been the more intuitive thing to do"	Allowed users to click on the title of a search result to view full details
Listing	"After I clicked on a route, I clicked the back button and there was an error about having to resubmit a form and I couldn't get my original search results back"	Changed form submission mechanics so users could press the back button to access their original search results
Listing	"Some of the route were really long and didn't fit in the map box"	The maximum zoom limit for routes on the listing and detail page was removed
Creation	"It was strange that I couldn't rename my route unless I opened the save popup, especially considering how the title is displayed on the page already"	An edit icon was added to the title, which users could click on to modify their route title, without having to open the save dialogue.
Creation	"I had to scroll and pan for a very long time to get the map to centre over my house in Guildford, because it started in Nottingham"	Added a search box on the route creation page that would take the entered location and centre the map on this point
Creation	"I was worried that clicking 'Save Route' would create a new, duplicate, route"	Changed button to read 'Save Changes' instead, when on the edit page.
Creation	"I accidentally deleted one of my points, and the 'Loading Map' thing stayed forever"	Modified behaviour of the loading icon to account for when points are deleted

8 Evaluation of the Project

At the beginning of the project, the three main aims of Niceway.to were stated, and it seems fitting to now use these aims to evaluate whether or not Niceway.to was a success. These aims were to build a community of travel enthusiasts, improve the travelling experience of those that use Niceway.to, and allow for users of all skill groups to access it. Unfortunately, the first two of these cannot be evaluated at this point, due to the system having no user base. The system can, however, be evaluated as to how well it facilitates the interaction necessary for these to be achieved at a later date.

The usability testing that was conducted was fairly conclusive in proving that the system was easy to use, for users of all skill groups. Of all the main actions that are available in Niceway.to, all of them could be completed by a brand new user to the system, with minimal help, in a short period of time, and this time would be lessened the more they used to site. This ease of use stems from the simple and descriptive language used throughout the site, the use of icons to depict the functionality of different features, and the use of errors messages that explained how to fix problems, rather than simply telling the user they had caused one. This meant that used could focus more on the content of the site, and were unrestricted and uninhibited by the medium in which it was represented.

One of the key successes of the project is the work conducted to facilitate the growth of a community around Niceway.to. There were several different approaches that were taken, to appeal to a wider audience, and attract more users. The first of these was the social stream present on every route. This was a list of all social interactions performed by all users on that route. This meant that other users could see what other users felt about this route, and could contribute their own opinions too. This was the key selling point of Niceway.to, and allowed users to communicate with one another, and feel like part of a community, where all members are working towards a common goal. The other method employed was the introduction of the customisation user profile picture, that allowed users to show off achievements and milestones they had reached. This targeted users that enjoyed the collecting of items and required them to be active members of the community in order to do so.

another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good another reason the software is good

As with most software, there are certain areas of this project that could be improved upon, if the project was to be worked on again, or more time was granted. The first of these was not in the application itself, but rather how it was implemented. Using the Zend framework, there is a huge amount of support for unit tests, and test driven development. However, in this project, these were not utilised, and a standard approach of testing each feature as it was implemented was used. This worked well for the beginning of the project, and meant that a lot of progress could be made very rapidly. However, as the project progressed, new features would begin to change older features in unexpected ways, and undetected bugs would begin to emerge. When these bugs were later found,

time was required to revisited previously "complete" areas of the project, to change them to work with the new features, and remove these bugs. This meant that a lot of time was wasted redoing work that was already complete. This also means that when external developers take over the project, they will find it harder to track down bugs and solve them, due to the lack of an automated testing framework.

As all routes on Niceway.to are used contributed, the tool for creating these routes needed to be simple to understand and use. The route creation page currently has support for users to add, edit, and delete points on a map, centre the map on specific locations, and join points with a route. This is the bare minimum functionality that one would expect, but it quickly becomes apparent that further functionality is required for such an important area of the website. This includes things such as being able to reorder points once a route has been constructed (at the moment, one wrong points requires other points to be moved or deleted), a better interface for modifying the details of individual points (at the moment, only a simple pop up is provided, which is inadequate for points with lots of detail), and one more! oh boy don't forget to fill this in otherwise that would be super embarrassing!. This lack of features could quickly become a limiting factor of the site, and may deter contributors, therefore making it something that should be addressed.

The final key limitation of the project, present on several pages on the application, is the routing software implemented. For the most part, the route provides accurate paths between points, in a relatively short period of time. However, for unknown reasons, there are occasions where parts of routes will simply not load, or take an extremely long time. This is probably due to this being a freely provided service, and the large number of requests being made by Niceway.to. This could be resolved by implementing a new routing service, potentially a premium service, which would provide more accurate, and more reliable results. Alternative (or in addition to this), the results of the routing could be cached on the server, to reduce the number of requests being made for routes, and allowing pages, and routes, to load much faster.

9 External Aspect

The client for this project is Matthew Pike, and his start up company, based in China. He was looking for a developer to meet the requirements for building the first version of the main software for this new company, that software being Niceway.to. Last year, Matthew reached out to a pair of developers with this same purpose, but he was disappointed with the quality of the project, and its lack of completion. For this reason, he decided that he would scrap the original work, and begin from scratch, with a new developer.

Communication with the client was initially facilitated through the supervisor for this project, Max L Wilson, who has a personal relationship with Matthew. After this point, a semi-frequent level of email communication was maintained by the client and I, at key points in the project. This was due to the client being based in China, and thus conflicting locality and time zones making face to face meetings, or even video calls, impossible to schedule. The table below summarises the main communication that took place throughout the course of the project, at the different stages, between the client and I.

Date	**Items Discussed**
Initial stages	
1st Oct 15	Initial specification received from client
8th Oct 15	Design specification with functional and non-functional requirements, initial designs, and market research provided to client
13th Oct 15	Setting up of web host for server for Niceway.to
15th Oct 15	Project proposal shared with client
Development stage	
9th Nov 15	Report of progress: ability for users to create accounts and log into the system, as well as near completion of the route creation page
29th Jan 16	Report of progress: completion of the route search page, route detail page and the user profile page.
Project Completion	
23rd Feb 16	Report of progress: Completion of coding

As far as fulfilling the original intentions of the client, I feel like the project did this well. Almost all of the communication between my client and I was positive, and he was pleased with the speed and quality of the progress being made. As can be seen in section 7 all functional and non-functional requirements of the system were adhered to, and there were even extra features included, that made the application easier to use for both the users, and the administrators. The usability tests also concurred that the system was easy to use, and users of all skills levels were able to understand and access the key areas and functionality of the site. From these tests it is clear to see that the website is ready for adoption by real users, which was an important aspect of the project for my client.

The "customers" of the project were defined by my client as belonging in one of two main categories, both of which are catered to differently. The first group is the "traveller", who has minimal interaction with the site, and benefits mostly from the content provided by other users, therefore their journey through the search process should be as simple as possible. The second group is the "experts" group, who are active contributors to the website, are actively engaged with the site, and their passion is travelling. For these users, the content creation and sharing systems are most important. As a final note, there is no general demographic for the application, and instead it is developed to be accessible by all users, regardless of any identifiable characteristic.

10 Further Work

The system that was produced adhered to all of the functional requirements that the client laid out at the beginning of the project, but that does not mean that all potential functionality has been implemented, or even that the system is truly complete. In this section some key areas of functionality have been identified and expanded upon, and it provides a starting point for the maintainers of the software to focus on.

The first of these areas is the routing system, specifically how the route between two points is calculated. Currently, the user is the main factor influencing how scenic a route is. It is expected they will select several key picturesque locations, and the drive between them will also be pleasant. However, this will not always be the case, and a good route could be spoiled with the transitions between two points, especially if the distance between the points is large. The reason for this is that the routing system provides the shortest path between any two selected points, rather than the most visually appealing. In future iterations of the project, it would be beneficial if some data mining was performed alongside the routing system, so that multiple routes could be generated, their pleasantness evaluated, and the best looking journey be picked. This would improve the quality of all the routes in the system, as well as further helping to achieve the main goal of the application: to improve the user's driving experience.

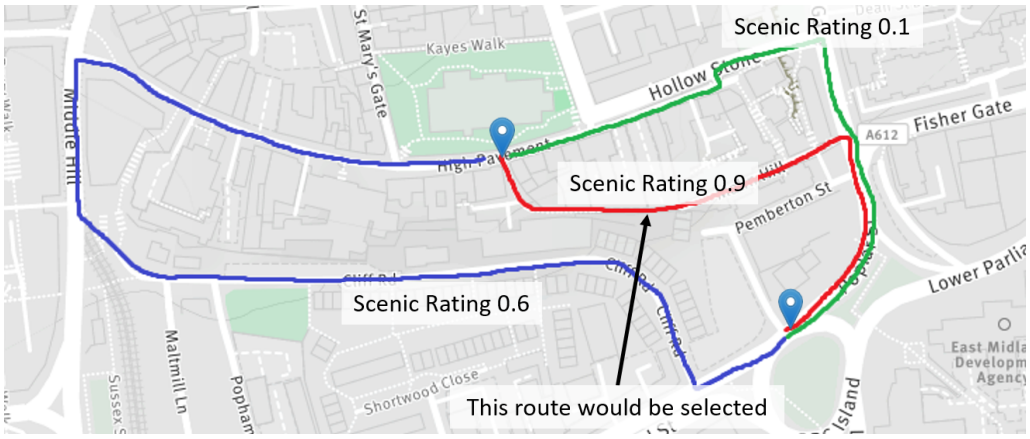


Figure 11: Route recommendation based on some scenic rating

Whilst on the topic of the route creation page, there are also other aspects which could be improved upon, by modifying the user interface, and adding more functionality. The first of these would be increasing the number of interactions possible, including the ability to change the order of points in the route, or the ability to add points in between other points. This would allow users to fix mistakes more easily, instead of having to delete segments of their route and reconstruct them.

The other change to the route creation process would be to increase the diversity of routes, by allowing the addition of walking and cycling segments. This would allow users to add more to their routes, increase their quality, and improve how scenic they were. This would also be useful for expanding the user base to users that don't necessarily have cars, but still wish to experience peer recommended scenic routes.

The next area of future expansion would be the social interaction system. As it stands, the system provides support for users to have six interactions with routes: commenting, rating, sharing, downloading, cloning, and recommending similar routes. This is fine for users that don't use the site frequently, or don't visit a lot of routes, but for dedicated and recurrent users, it quickly becomes apparent there are some issues with managing their social interactions. There is no central place where a user can view all the interactions that other users have had with them. This means that if a user wants to keep a track of this, they have to consistently check their emails, or visit each of their routes individually, which becomes much more difficult the more routes and interactions the user has. The solution to this problem is to implement some form of notification centre, where a user can see all the notifications they have received, similar in functionality to the notification system employed by Facebook. This would allow for users to review all of their notifications in a centralised place, so that they do not miss any, and do not need to check multiple locations for their notifications (which would not scale well as the number of sources of notifications increases). This could then be further expanded with the ability to follow routes and other users of the system, so that notifications could be received for these as well.

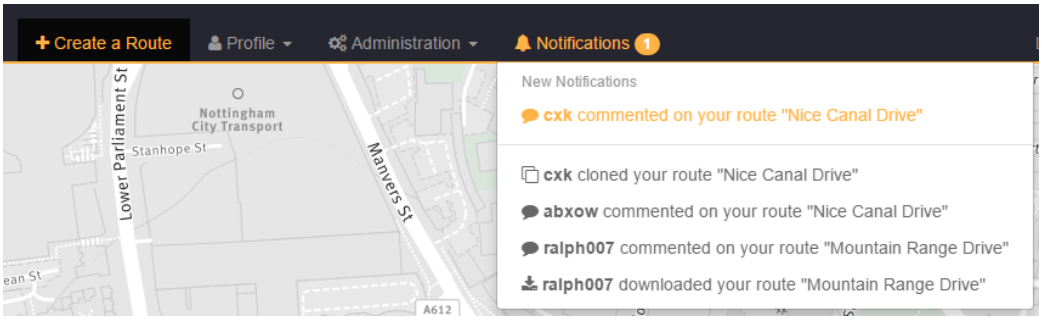


Figure 12: Mock up of the notification system, displaying all social interactions received

The final area of further work is also an improvement on social interactions within the site, specifically the implementation of a site wide messaging system. This would allow for users to interact with one another on a much more personal level than simply commenting on each others routes. This deeper social interaction could become one of the main driving features for users to returning to the site. It provides the ability for users to talk to those that provide quality content, friendships to form, and for groups with similar interests to develop. It would also mean that communication between members would be contained within the Niceway.to ecosystem, rather than users communicating on other social platforms, and potentially missing out on what their friends are doing on Niceway.to. This freedom to communicate with other users would open up a lot of potential for Niceway.to and would increase the chance of users returning to the site on a regular basis (to check for any new messages).

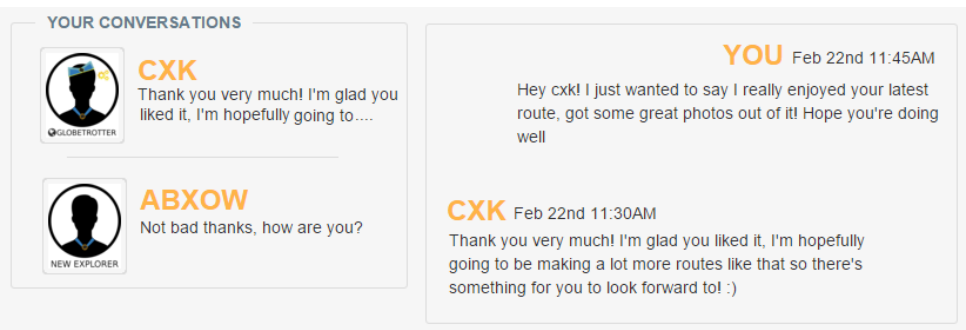


Figure 13: Mock up of the chat system

11 Summary & Personal Evaluation

Despite some ups and downs through the lifetime of the project, I would categorically claim it was a success. This comes from the meeting of all of the functional and non-functional requirements, and ample user feedback claiming that the website is aesthetically pleasing and simple to use (regardless of user skill level).

Replacing the title held by my dissertation, this software was the largest single piece of software I had ever constructed by myself, with a code base of over 12,000 lines, split over almost 80 files. I used several methods of planning and time management (many of which were things that I learnt worked well from my dissertation), which meant that I was often ahead of my initial project time plan (which was well received by my client, and meant I had more time to revise the project from feedback received). This included the Gantt Chart, for the "big picture" planning of large sections of work, and Trello for the day-to-day, granular tasks. I feel that one of the main reasons I worked so efficiently on this project is that it was fun to work on, as I enjoy both the languages that were used (PHP and JavaScript), and the platform utilised (responsive web applications).

As elaborated in section 6.3, one of the key flaws in the project was the lack of unit tests being utilised. This caused huge issues in the project when new features were being added, with many unseen bugs being introduced. Time was then required later in the project (when these bugs were eventually discovered) to go back and fix these issues. The reason for not using unit tests was due to my eagerness to get started on the project, causing me to write sloppy code during the prototyping stage, that did not have unit tests. Unfortunately, this prototype code was carried forward into the actual implementation of the project, and therefore also this cavalier attitude towards unit tests. This meant that the project could progress at an accelerated rated, but the drawback was the introduction of these bugs that would cause issues later.

If given the opportunity to restart this project from scratch, there are several changes I would implement to the development process. The first, and potentially most obvious, would be the use of unit tests for all aspects of the project, as this would have made integration and regression tests much simpler, and allowed me to identify new bugs quicker. In addition to this, I would have spent more time, and gone more in depth, during the research portion of the project. In this section, several pieces of software, as well as different tools I could use were evaluated. However, only three only pieces of software and only front end and back end tools were looked at. I feel as though the project would have benefited from looking at a greater range of other software systems (to determine what is good about them, and what things to avoid), and well as looking at more different kinds of tools to use (for instance: which mapping service to use, which routing service to use, and what JavaScript libraries would be useful). This would have meant that these choices didn't need to be rushed later in the project, and I could have picked the right tool for the job, in each of these cases.

References

[1] Google Maps, My Maps website. `https://www.google.com/maps/d`. Accessed: 01/02/2016.

[2] MADMAPs website. `http://www.madmaps.net/`. Accessed: 01/02/2016.

[3] My Scenic Drives website. `https://www.myscenicdrives.com/`. Accessed: 01/02/2016.

[4] Boris Beizer. *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc., 1995.

[5] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou. Discovering popular routes from trajectories. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 900–911. IEEE, 2011.

[6] Joshua Gross. Please stop using twitter bootstrap, 2013.

[7] Matthew Joint. Road rage. 1995.

[8] Henrik Kniberg. Scrum and xp from the trenches. *Lulu. com*, 2007.

[9] Jonathan Lazar, Adam Jones, and Ben Shneiderman. Workplace user frustration with computers: An exploratory investigation of the causes and severity. *Behaviour & Information Technology*, 25(03):239–251, 2006.

[10] Kimberly Ling, Gerard Beenen, Pamela Ludford, Xiaoqing Wang, Klarissa Chang, Xin Li, Dan Cosley, Dan Frankowski, Loren Terveen, Al Mamunur Rashid, et al. Using social psychology to motivate contributions to online communities. *Journal of Computer-Mediated Communication*, 10(4):00–00, 2005.

[11] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 352–361. ACM, 2009.

[12] John D McCarthy, M Angela Sasse, and Jens Riegelsberger. Could i have the menu please? an eye tracking study of design conventions. In *People and computers XVI-IDesigning for society*, pages 401–414. Springer, 2004.

[13] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[14] Jakob Nielsen. 10 usability heuristics for user interface design. *Fremont: Nielsen Norman Group.[Consult. 20 maio 2014]. Disponível na Internet*, 1995.

[15] Heather L O'Brien and Elaine G Toms. What is user engagement? a conceptual framework for defining user engagement with technology. *Journal of the American Society for Information Science and Technology*, 59(6):938–955, 2008.

[16] Ofcom. The communications market report, published 6th august 2015. Ofcom, 2015.

[17] Fernando S Peregrino, David Tomás, Paul Clough, and Fernando Llopis. Mapping routes of sentiments.

[18] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. In *Proceedings of the 25th ACM conference on Hypertext and social media*, pages 116–125. ACM, 2014.

[19] Shneiderman Ben Shneiderman and Catherine Plaisant. Designing the user interface 4 th edition. *ed: Pearson Addison Wesley, USA*, 2005.

[20] Pierre Thiffault and Jacques Bergeron. Monotony of road environment and driver fatigue: a simulator study. *Accident Analysis & Prevention*, 35(3):381–391, 2003.

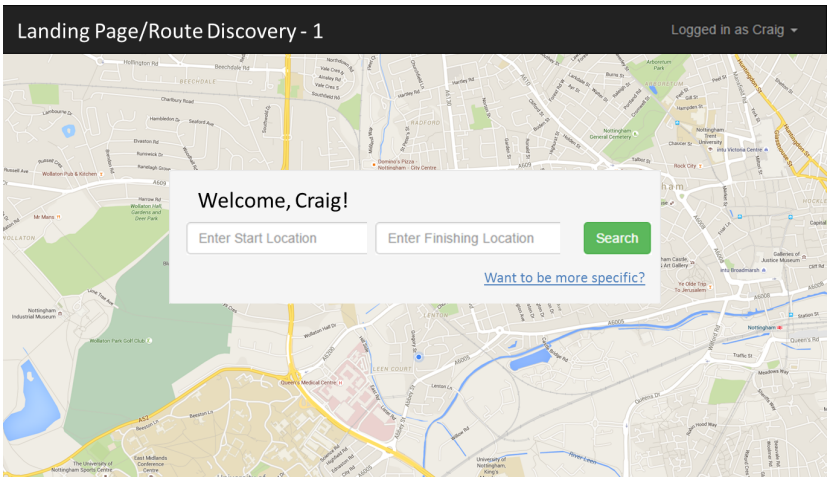
[21] Steven Van Canneyt, Steven Schockaert, Olivier Van Laere, and Bart Dhoedt. Time-dependent recommendation of tourist attractions using flickr.

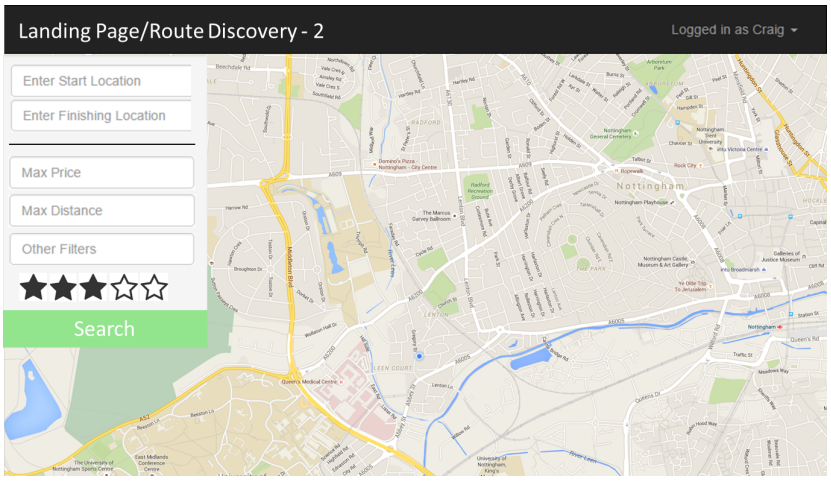
[22] Jason Vest, Warren Cohen, and Mike Tharp. Road rage (usa). *US News and World Report*, 1997.

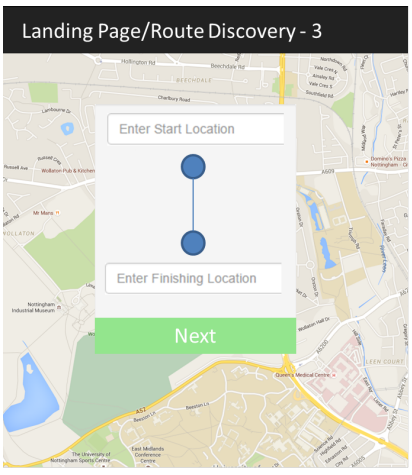
[23] Susan B Wilson and Michael S Dobson. *Goal Setting: How to Create an Action Plan and Achieve Your Goals*. Publisher - Amacom, 2008.

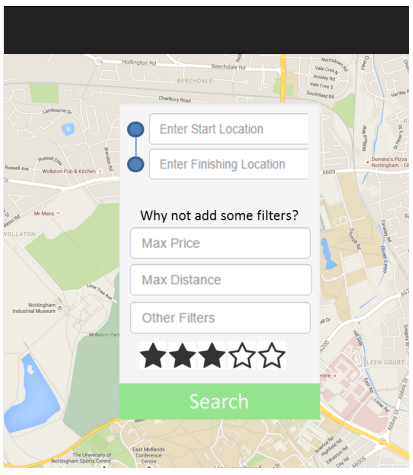
A Initial System Designs

A.1 Route Discovery/Landing Page

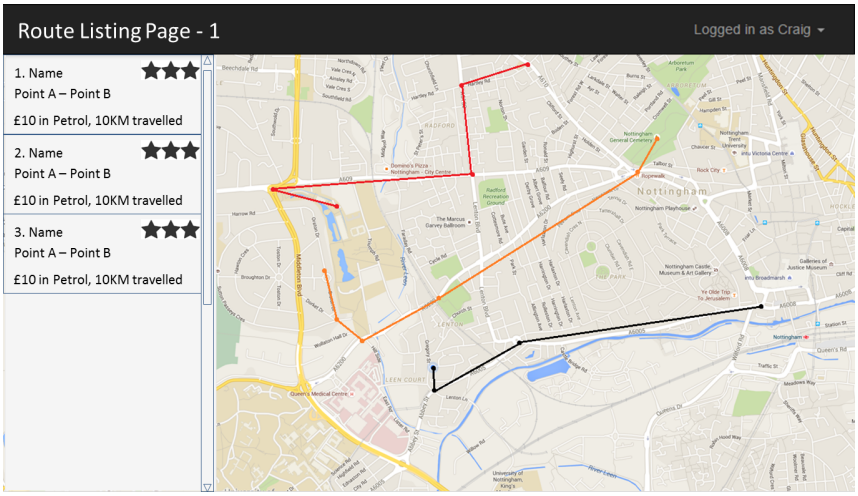


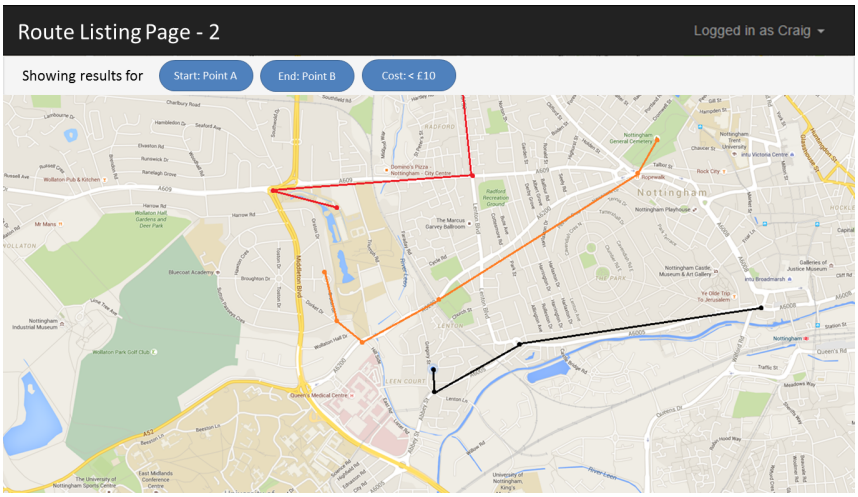


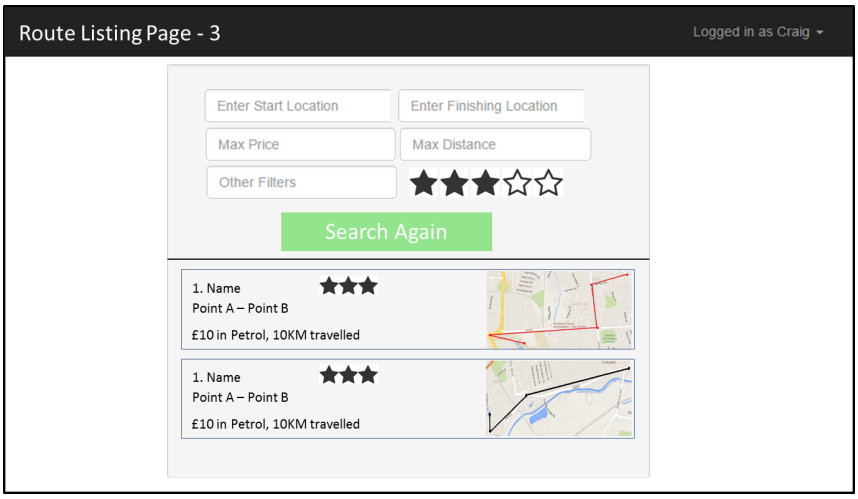




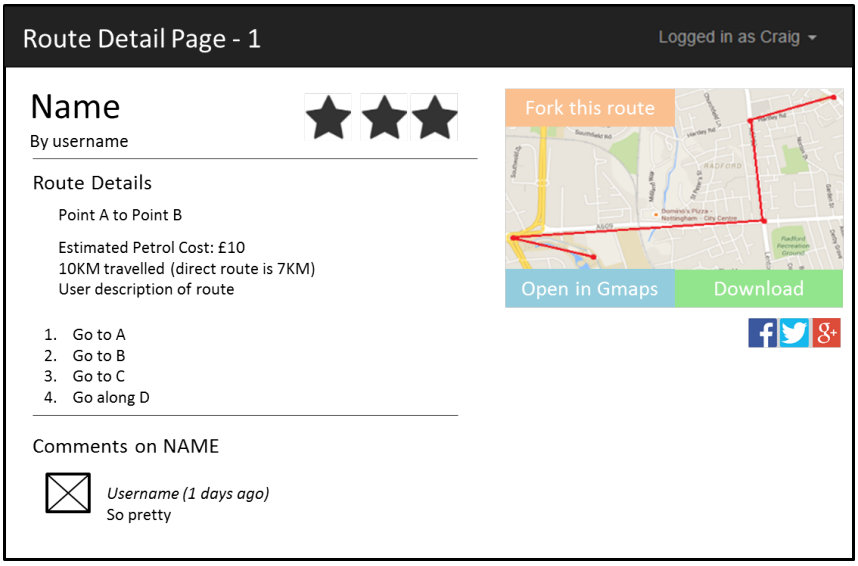
A.2 Route Listing Page

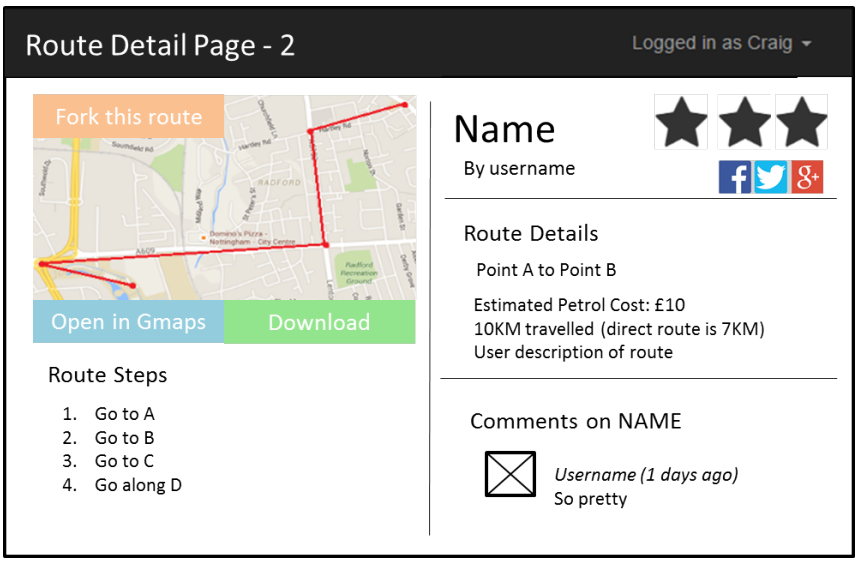


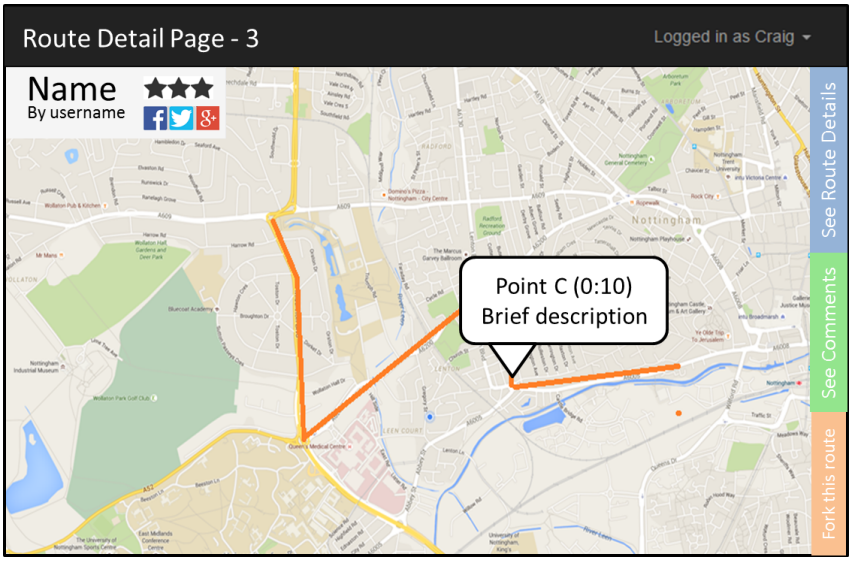




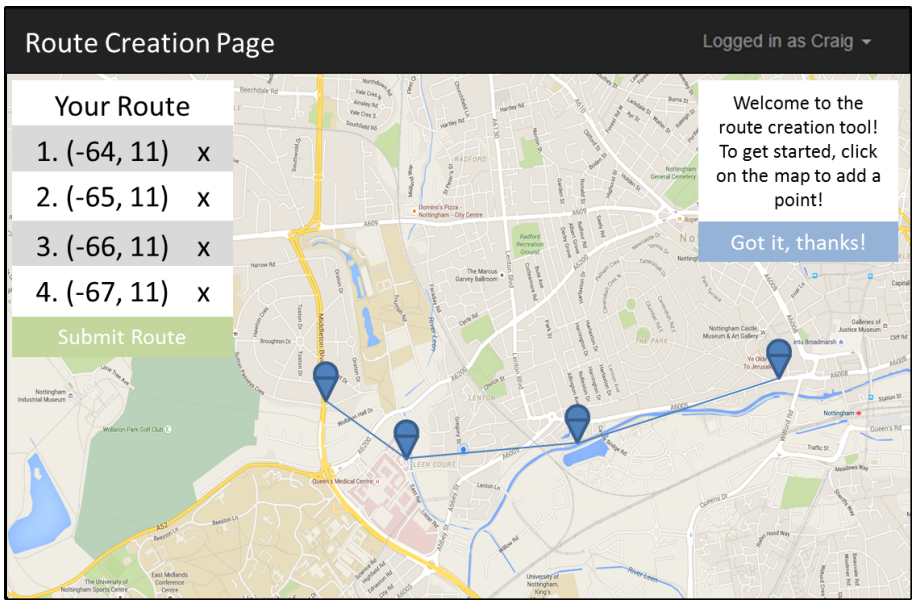
A.3 Route Detail Page



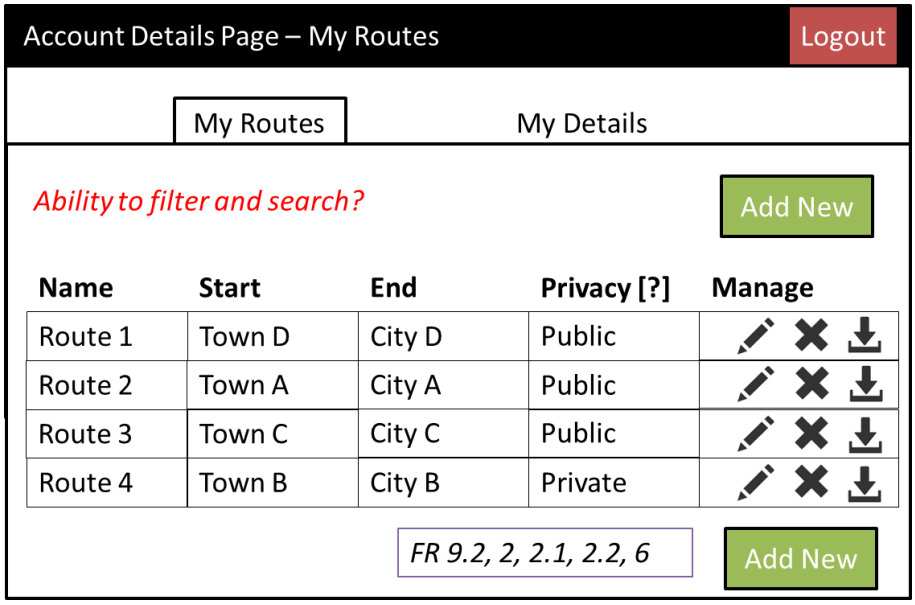




A.4 Route Creation Page



A.5 Profile Page



B Interface Design Heuristics

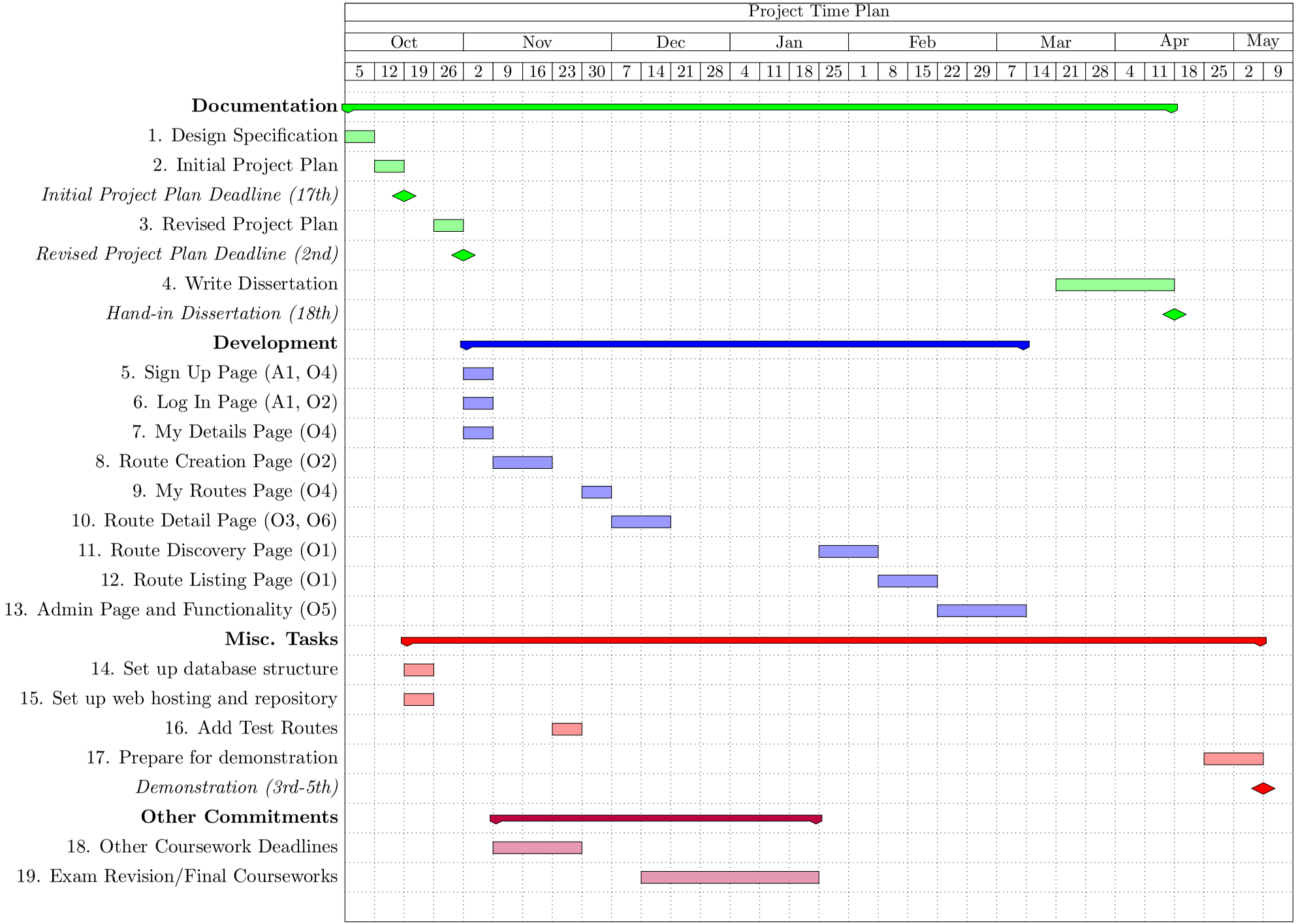
B.1 Jakob Nielsen's Usability Heuristics

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

B.2 Ben Schneiderman's Golden Rules

1. Strive for consistency.
2. Enable frequent users to use shortcuts.
3. Offer informative feedback.
4. Design dialog to yield closure.
5. Offer simple error handling.
6. Permit easy reversal of actions.
7. Support internal locus of control.
8. Reduce short-term memory load.

C Project Gantt Chart



D Usability Testing Instruction Set

The purpose of these tests is for you to experience the five main features of Niceway.to so that their ease can be evaluated. Follow the instructions below to complete each of the tasks, during which you will be timed. Please make a note, or comment on anything you find difficult or unintuitive, or any bugs you find.

Task One - Searching for routes

1. Navigate to http://www.niceway.to
2. Search for a route from Nottingham to Derby
3. Navigate to the details page of one of these routes

Task Two - Creating an account

1. Navigate to http://www.niceway.to
2. Sign up for an account
3. Ensure you get an email confirmation

Task Three - Interaction with a route

1. Navigate to http://www.niceway.to
2. Search for a route from Nottingham to Derby
3. Navigate to the details page of one of these routes
4. Leave a comment on the route
5. Give this route a rating
6. Download this route

Task Four - Creating a route

1. Navigate to http://www.niceway.to
2. Go to the route creation page
3. Create a route with three points
4. Save this route and give it a name

Task Five - Editing your route

1. Navigate to http://www.niceway.to
2. Without using the search feature, find the route you just made
3. Open your route for editing and change the title
4. Save your route

E Times recorded for usability tests

The tasks users were asked to perform, with the times taken listed in the table below, were as follows:

1. Search for a route, and view it's details
2. Create an account for the system
3. Comment, rate and download a route
4. Create a route
5. Navigate back to their route, and make edits

	Completion Time in Seconds				
Participant	Task 1	Task 2	Task 3	Task 4	Task 5
1	30	25	42	36	60
2	22	47	58	30	30
3	20	37	37	36	45
4	40	51	78	72	65
5	22	20	36	35	45
6	44	35	60	51	75
7	25	42	40	43	44
8	59	71	78	49	111
9	18	36	27	48	36
10	22	43	31	48	40
11	60	62	83	56	52
12	37	54	47	44	67
13	32	46	45	46	72
14	28	26	36	44	56
15	47	49	52	58	91