

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
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	1
2	Motivation	2
3	Background Information & Research	3
3.1	Existing Systems	3
3.2	Platforms and Tools	5
3.2.1	Native Mobile Applications VS Responsive Web Applications	5
3.2.2	Back End Tools	6
3.2.3	Front End Tools	7
4	System Specification	8
4.1	Functional Requirements	8
4.2	Non-Functional Requirements	8
5	System Designs	10
5.1	Navigation/Control Flow Design	10
5.2	User Interface Design	12
5.2.1	Heuristic Evaluation of the Interface	15
5.3	Internal Design	17
6	Software Implementation	18
6.1	Key Implementation Decisions	18
6.1.1	Implementation Platform - Responsive Web Application	18
6.1.2	Back End Tool - The Zend Framework (PHP)	18
6.1.3	Front End Tool - Twitter's Bootstrap	18
6.2	Implementation Methodology	19
6.3	Problems Encountered	20
7	Testing of the Project	21
7.1	Functional Testing	21
7.2	Non-Functional Testing	21
7.3	User Feedback Testing	22
8	Evaluation of the Project	24
9	External Aspect	25
10	Further Work	26
11	Summary & Personal Evaluation	28
A	Usability Testing Instruction Set	31
B	Times recorded for usability tests	32

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 has risen 51% [23].

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 [18][22][19]) 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 of users to contributions content. Google’s “My Maps”, and MyScenicDrives are optimised for desktop browsing and their support for mobile devices is limited. 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[17]. 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 the 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[21].

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[16] 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 sections look at some software systems that are already attempting to solve the problem identified in this dissertation, 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 features and the worst features, 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 point, and generate the route afterwards, rather than generating the route after each point is placed. This is an interesting 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 is to build a community of users. If they can only share their routes to *other* platforms, the community will have a much smaller 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.

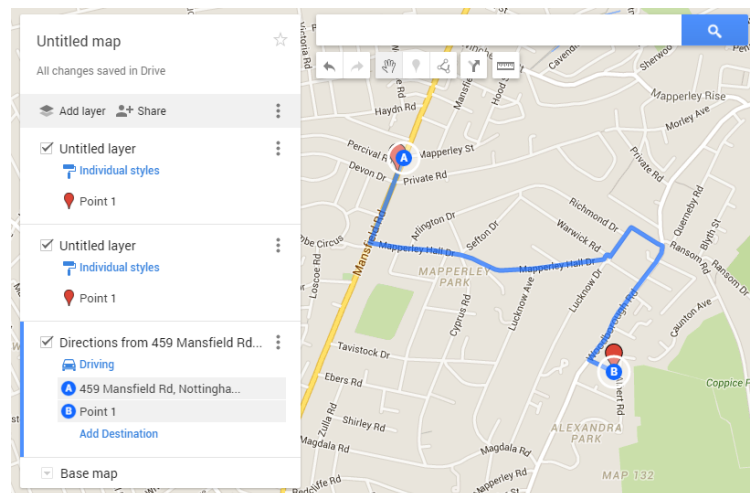


Figure 1: Google’s “My Maps” route creation/editing feature

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 that 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 point of MyScenicDrives.

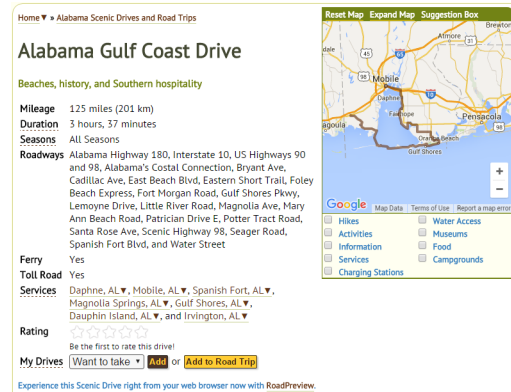
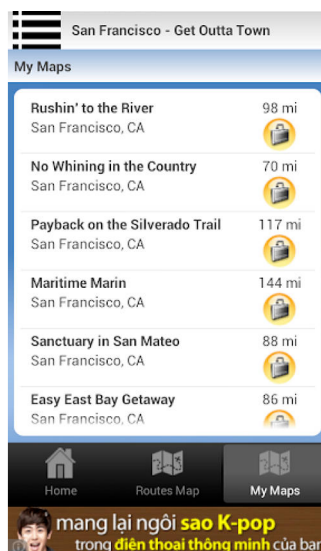


Figure 2: MyScenicDrive's route details

However, no matter how high the quality of this content is, 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 downfall 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 will 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.



The service works by having a group of “experts” compile the maps, and distribute them to the users. This is supposed to instill confidence in 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.

Figure 3: List of Mad Maps routes

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 Framework¹ 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 it is 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 Rails² 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 one 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 continuity.

The negatives of Ruby on rails is that it is difficult to find good documentation for it, and it is more resource intensive than 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.js³ 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 its 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 separate 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

Bootstrap⁴ 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, including phones, tablets, and desktops. Due to its popularity, there is a huge amount of support online, as well as extensive documentation provided by Twitter themselves. These things, combined with its 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.

One of the biggest advantages of Bootstrap, its huge user base, had proven to also be one of its biggest downfalls. Due to the quantity of websites now using Bootstrap, a large 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 Framework⁵ 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 its 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

Semantic⁶ is the final front end design framework that was investigated, and is the newest of the three. The main advantage of semantic is its 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 this system have been detailed and justified. This includes the design of the user interface itself, the navigation and expected user path through the application, 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 a 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 their avatars appearance.
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 this routes detail page, 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.

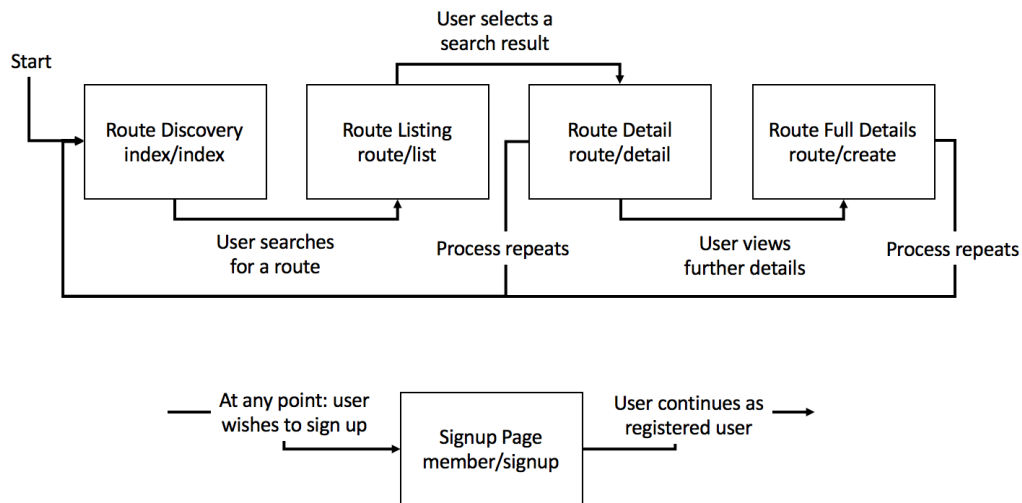


Figure 4: Expected path of non-registered users

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 bookmarked specific pages, and would need to log in to access them. After logging in, it is expected the user would spend some time on their profile page, and then using this page as the springboard into one of three main 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.

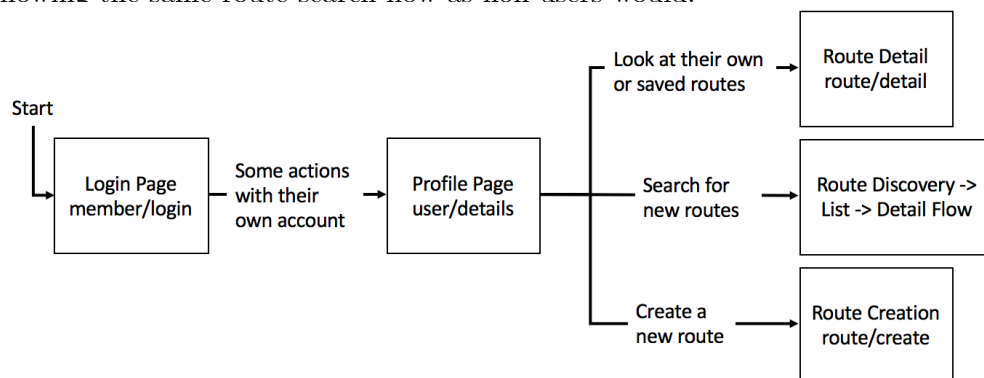


Figure 5: Expected path of registered users

Of course, this is just the generally expected path, and users are in no way restricted to what they can do (with the acceptable 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.

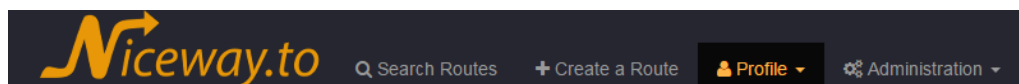
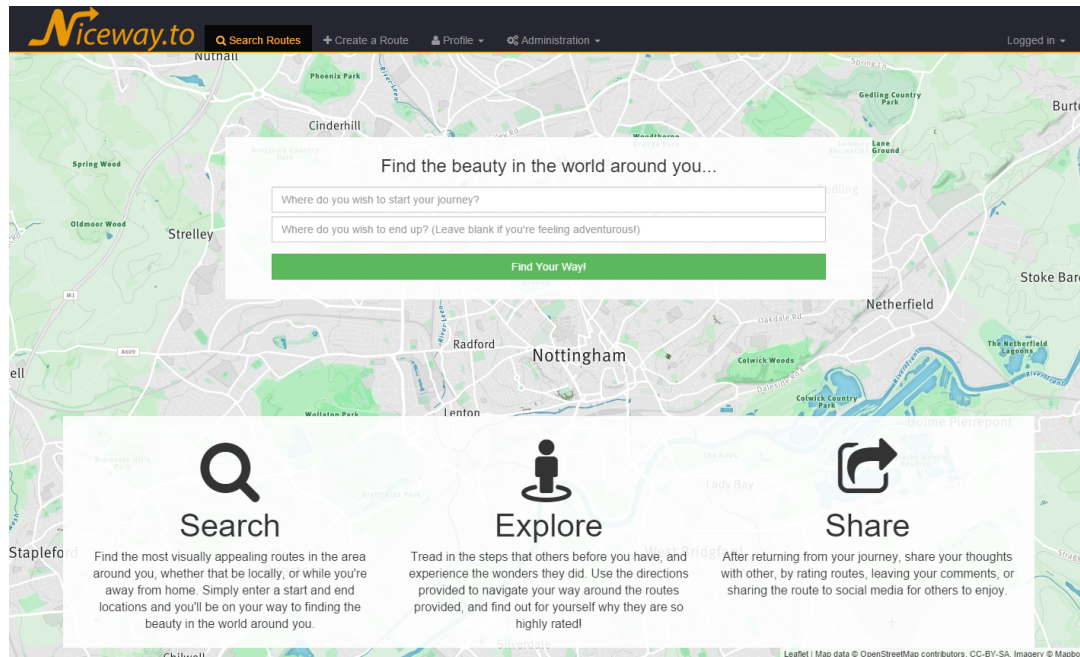


Figure 6: The navigation bar displayed to registered users

5.2 User Interface Design

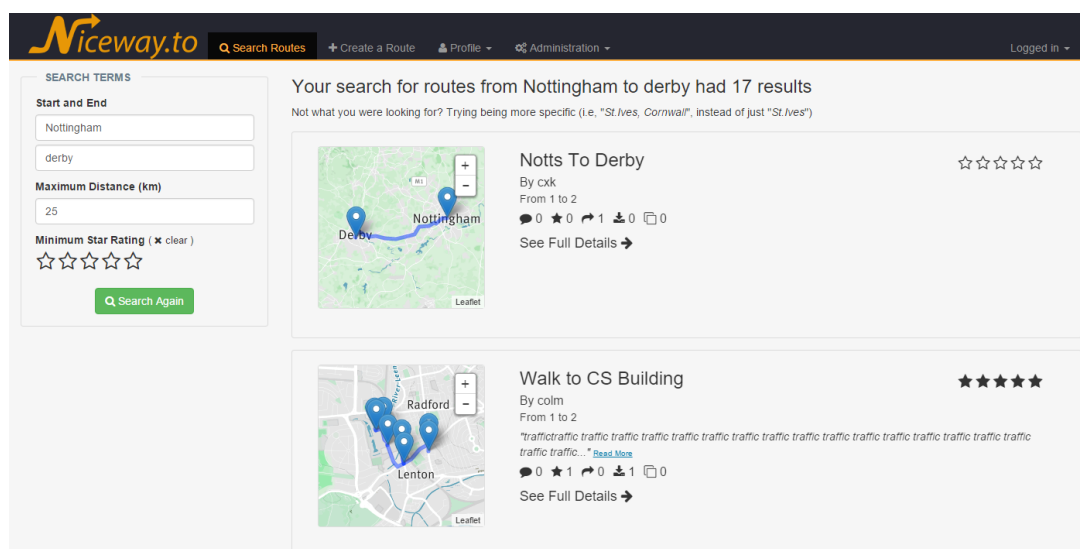
In this section, the interface for each of the main pages of the application have been displayed, along with justifications for why they were designed the way they were. Designed for each of these pages were produced at the beginning of the project and shared with the client. The design presents here 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 ??.

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's website), 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.

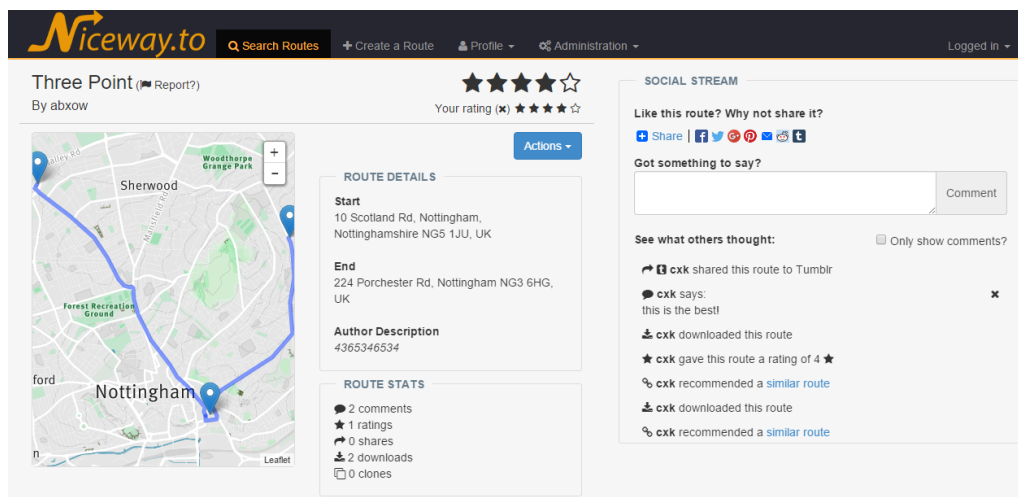
Route Listing Page



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 user 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, number of social interactions and a summary of comments 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

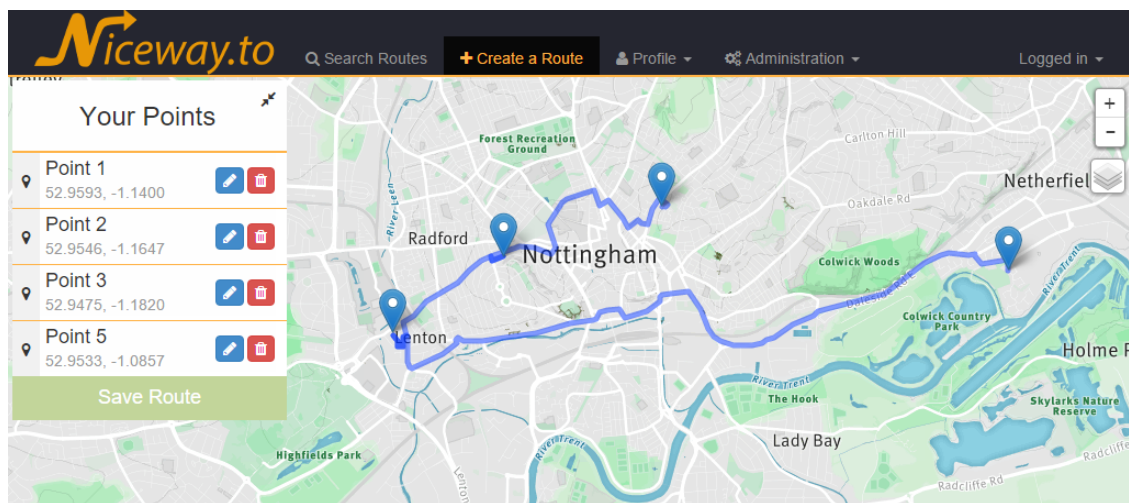


The route detail page was the page used to list all information about one specific route - for this reason, there was a lot of potential for this page to become cluttered. To combat this, the page is clearly divided into two main segments, which are then also further divided. These main sections were 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 it's 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 map, 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 obscured.

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 hit 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 screenshot shows the Niceway.to user profile page for user CXK. The header includes the Niceway.to logo, search and create route buttons, and a navigation menu with Profile and Administration. The profile section features a custom profile image, the user's name (CXK), a 5-star rating, and personal details: Craig Knott, 22 years old, email cck01u@gmail.com (Confirmed), Nottingham UK, and joined 02/11/15 (98 days old). A bio states: "I am a Masters student at the University of Nottingham." Below this is a "YOUR STATS" box showing: 12 routes submitted, 11 comments posted, 1 comments received, 5 ratings given, and 0 ratings received. The "ROUTES" section displays a table of routes created by the user, and the "SAVED ROUTES" section displays a table of saved routes.

Name	Points	Privacy	Created	Comments	Clones	Downloads	Rating	Manage
Media Demo	3	Public	28 Jan 16	0	4	0	3 ☆☆☆☆☆	Info Edit Download Delete
My Uni Houses	4	Public	27 Nov 15	0	0	1	0 ☆☆☆☆☆	Info Edit Download Delete

Name	Points	Created	Comments	Clones	Downloads	Rating	Actions
just a test	7	02 Nov 15	1	1	0	1 ☆☆☆☆☆	Info Edit Download Delete

The final major page for the website was 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 customisation profile image, email account age, and location, and well as statistical information, like the number of comments their routes had received. This meant that other users could quickly build up an idea of what this person 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 person 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[15], and the golden rules of interface design identified by Ben Sschneiderman[20]. 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 ??.

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 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 is displayed to the user, and interfaces are kept as minimalistic as possible. Another way to reduce the cognitive load on the user is to remember data for them. This is the principle applied on the search results page, where the search terms are displayed to the user, which allows them to drop them from their 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.

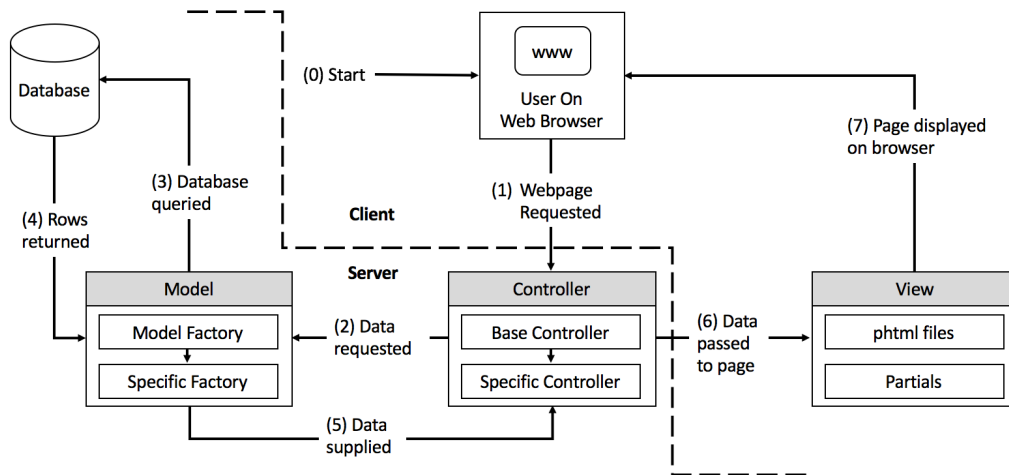


Figure 7: The internal design of Niceway.to

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 reason for their adoption has 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 of these platforms. 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 speed of developing 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 screensizes 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-confirm⁷ 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 previous dissertation) 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 Trello⁸ (combined with the extension Scrum for Trello⁹ 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 (this usually led to more productivity, because the development environment had been set up to complete that task, so it made sense to complete some more tasks). 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[24], which is why this Kanban method was so successfully, especially as this could be shared with my project supervisor.

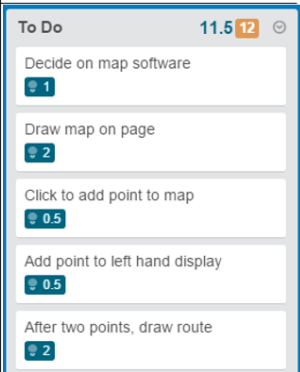
Requirement	Subtasks	Trello Representation
FR.8 There should be a route editor component which allows the users to construct a route	<ul style="list-style-type: none"> • Drawing of map • Ability to add points • Ability to connect points • Ability to edit points • Ability to save map • Load previous map 	

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 ??), 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 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 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 time, 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 in 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, but was then 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 be 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 previously 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 which 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 in turn, to ensure the system meets them. Knowledge of the inner workings of the system is not actually necessary to understand these tests, as they simply check whether functionality is present, and are not concerned as to how the system actually implements it (this is known as black box testing [4]). A complete listing of all the tests conducted, and their results, can be found in appendix ??.

some interesting thing to point out... ?

7.2 Non-Functional Testing

- Look at non-functional requirements and talk about if they were met
- Just like last year's diss

Accessibility

What this met? If so, **how**?

Usability and Operability

What this met? If so, **how**?

Maintainability & Documentation

What this met? If so, **how**?

Quality

What this met? If so, **how**?

Resource Requirements and Constraints

What this met? If so, **how**?

Cross Platform Compatibility

What this met? If so, **how**?

Security

What this met? If so, **how**?

Disaster Recovery

What this met? If so, **how**?

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, with 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 A.

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

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 with 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. This meant that each participant would be looking at a slightly different product, and they would all be able to identify unique issues instead of being focused on the same issue.

diagram showing user being tested – > feedback – > iterate / change product – > repeat

Figure 9: The employed RITE method life cycle

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

Bar chart of min/max/average time taken to complete each task against one another. (on seperate plots)

Figure 10: Summarised results from user testing

While the numbers themselves are not necessarily important, the range of these values is. This is the different between the fastest user, and the slowest user. This range shows how difficult or simple the task is. If a task has a large range, it means the skill gap is a major factor in how to use the feature. This means that either the task is difficult, or there are shortcuts that advanced users utilised to complete the task quicker. [What did I do as a result of this?](#). Specifically talk about each of the tasks and why the results were as they were.

The 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 will be able to detect unique and novel issues, rather than each user identifying the same set of issues.

Issue Identified

Could not press 'Return' to search on the search page

Couldn't click on title on search results page

ask max ethics for this type of testing

8 Evaluation of the Project

Successes and Limitations of the Project

- As a result of the test...
- + get proof that it is easy to use for morons, and claim this is a success (due to handling of errors and intuitive design)
- + modular design and adherence to design patterns with well commented code
- +
- - creation page was lacking in features
- - routing unreliable at times, because of using free service
- - lack of unit tests, especially considering zend and external devs

9 External Aspect

- Very similar to proposal
- as well as explicitly addresses how your project fulfilled (or not) its original intentions with regard to its ‘external aspect’.
- maybe put this section AFTER evaluation? (or just combine the two...?)

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 provide 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, allowing the system to pick the best looking journey. 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.

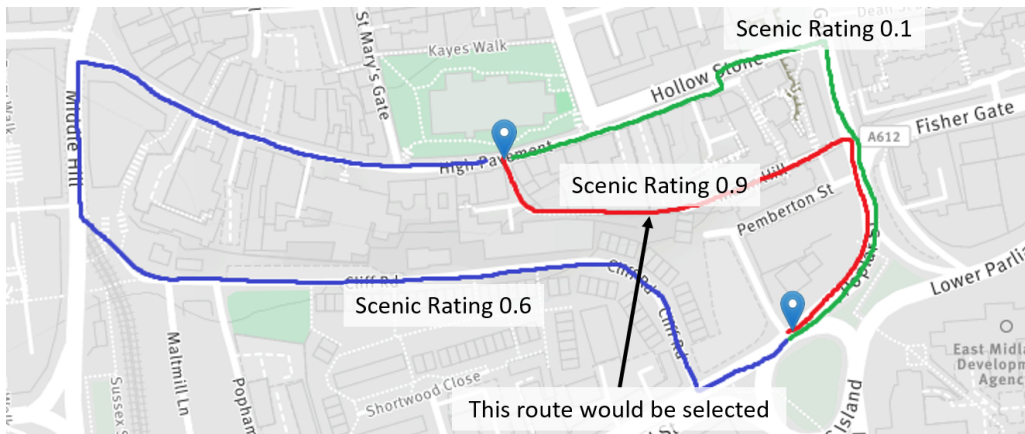


Figure 11: Route recommendation based on some scenic rating

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 one 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, their routes, 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.

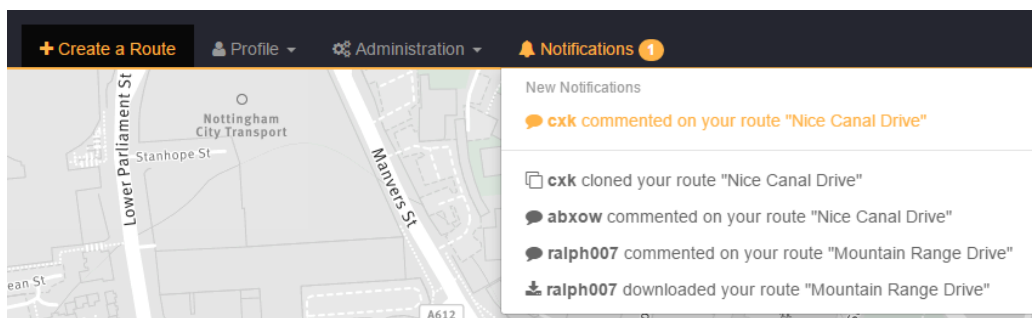


Figure 12: Mock up of the notification system, displaying all social interactions received

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 interact could become one of the main driving features for users to returning to the site. It provides the ability for users to talk to users 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's happening with their friends 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).

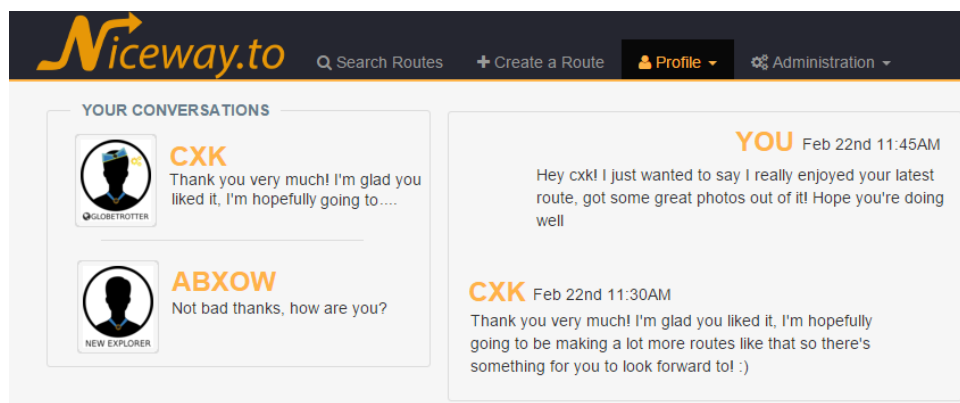


Figure 13: Mock up of the chat system

11 Summary & Personal Evaluation

- Personally, I feel as though the project was a /success|failure/
- I felt as though I /failed to rise|rose/ to the challenge
- One of the areas I feel as though was weaker within the project
- If I could work on this project again

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. The usability engineering life cycle. *Computer*, 25(3):12–22, 1992.
- [15] Jakob Nielsen. 10 usability heuristics for user interface design. *Fremont: Nielsen Norman Group.[Consult. 20 maio 2014]*. Disponível na Internet, 1995.
- [16] 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.
- [17] Ofcom. The communications market report, published 6th august 2015. Ofcom, 2015.
- [18] Fernando S Peregrino, David Tomás, Paul Clough, and Fernando Llopis. Mapping routes of sentiments.

- [19] 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.
- [20] Shneiderman Ben Shneiderman and Catherine Plaisant. Designing the user interface 4 th edition. ed: *Pearson Addison Wesley, USA*, 2005.
- [21] Pierre Thiffault and Jacques Bergeron. Monotony of road environment and driver fatigue: a simulator study. *Accident Analysis & Prevention*, 35(3):381–391, 2003.
- [22] Steven Van Canneyt, Steven Schockaert, Olivier Van Laere, and Bart Dhoedt. Time-dependent recommendation of tourist attractions using flickr.
- [23] Jason Vest, Warren Cohen, and Mike Tharp. Road rage (usa). *US News and World Report*, 1997.
- [24] Susan B Wilson and Michael S Dobson. *Goal Setting: How to Create an Action Plan and Achieve Your Goals*. Publisher - Amacom, 2008.

A 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

B 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

Participant	Completion Time in Seconds				
	Task 1	Task 2	Task 3	Task 4	Task 5
1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5
4	1	2	3	4	5
5	1	2	3	4	5
6	1	2	3	4	5
7	1	2	3	4	5
8	1	2	3	4	5
9	1	2	3	4	5
10	10	2	3	4	5
11	10	2	3	4	5
12	10	2	3	4	5
13	10	2	3	4	5
14	10	2	3	4	5
15	10	20	30	40	500