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
cxk01u

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 [14]. 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 it's 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 affect 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% [27].

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 [22][26][23]) 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 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[21]. 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[10]. 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[25].

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[13]. 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[20] 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[12].

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.

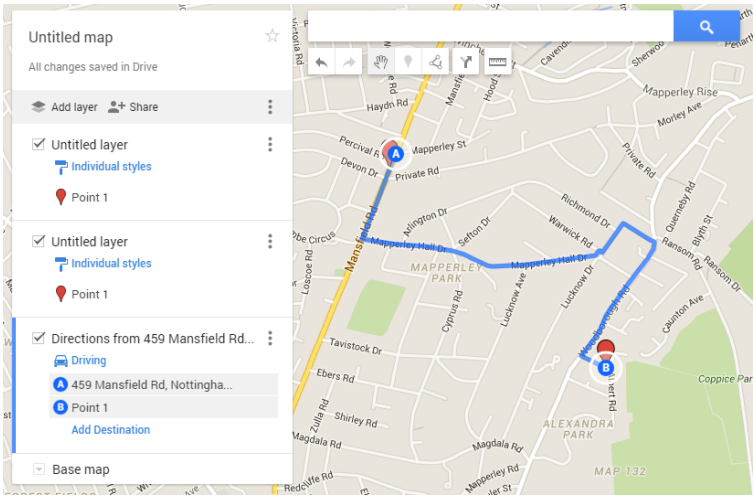


Figure 1: Google's "My Maps" route creation/editing feature

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 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.

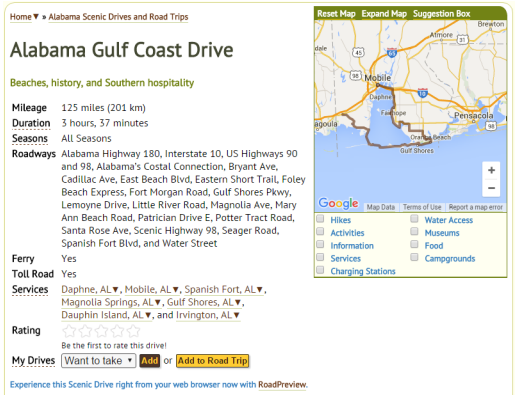


Figure 2: MyScenicDrive's route details

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 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.

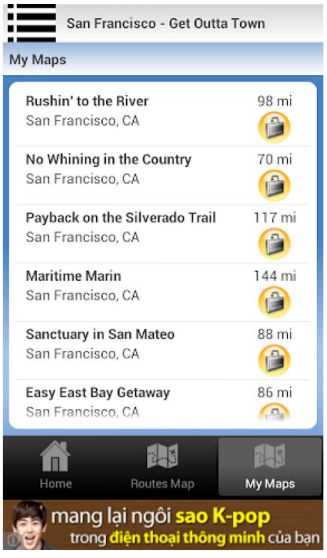


Figure 3: List of Mad Maps routes

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 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 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 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
Bootstrap[4] is a very popular front end design framework produced by Twitter. It is fully responsive and scales easily and efficient 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 help to reduce the overall size.

One of the biggest advantages of Bootstrap, it's huge user base, had proven to also be one of it's 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[9], 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.

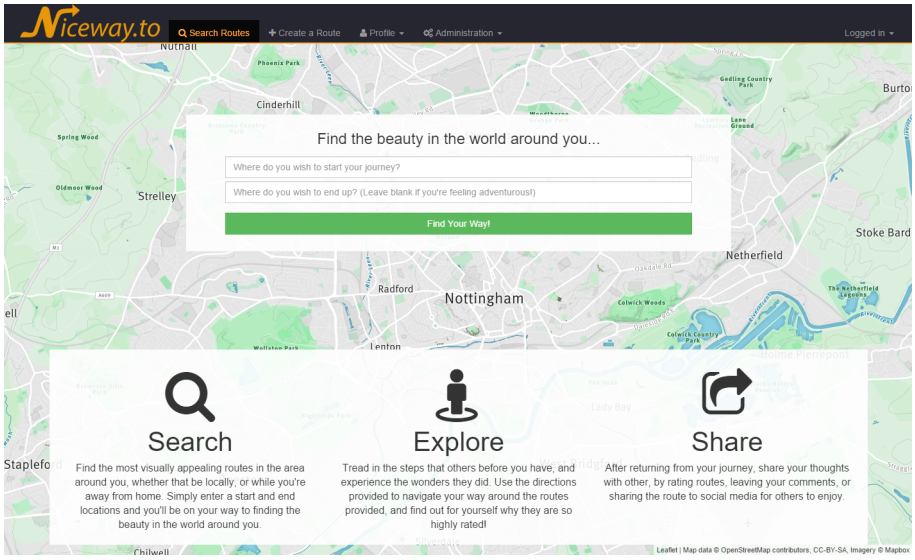
5 System Designs

In this section, all of the design aspects of 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 User Interface Design

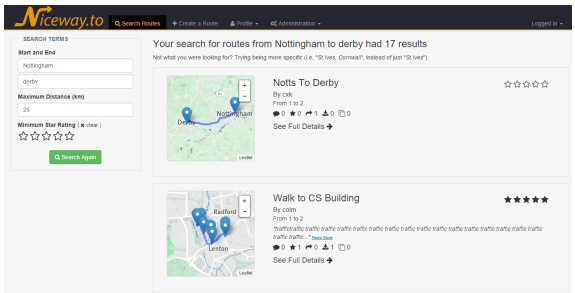
In this section, the interface for each of the main pages of the application have been displayed, along with justifications for why they were designed the way they were. Before any work was completed on the project, designs for each of these pages were produced, and shared with the client. The finale designs were then based on his feedback, and the combination of the best features from all the designs produced. These initial designs can be found in appendix A and are referenced throughout this section. s

Route Discovery Page/Landing Page



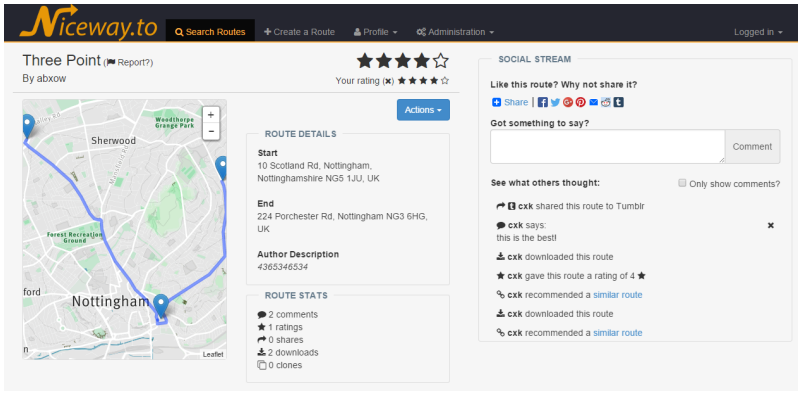
Why it is good
What was taken from each of the designs in the appendix
How it addresses the problem
Why it is designed the way it is
What differences there are to the designs

Route Listing Page



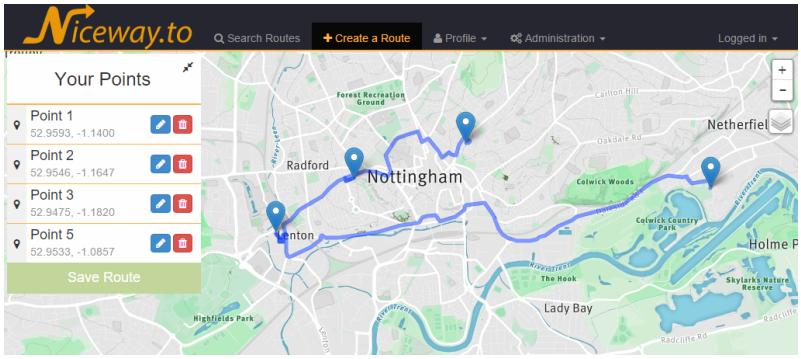
Why it is good
What was taken from each of the designs in the appendix
How it addresses the problem
Why it is designed the way it is
What differences there are to the designs

Route Detail Page



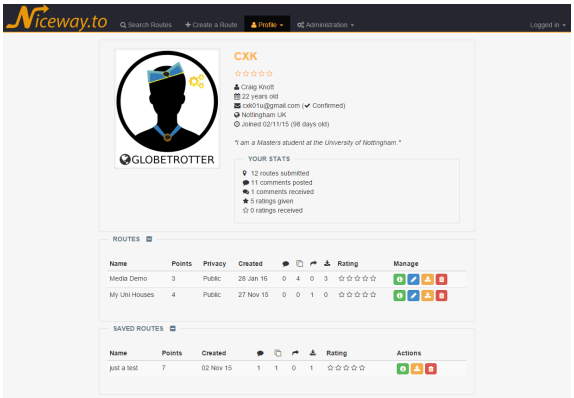
Why it is good
What was taken from each of the designs in the appendix
How it addresses the problem
Why it is designed the way it is
What differences there are to the designs

Route Creation Page



Why it is good
What was taken from each of the designs in the appendix
How it addresses the problem
Why it is designed the way it is
What differences there are to the designs main client feedback was more social - so implemented the social stream to emphasize

My Account Page



Why it is good
What was taken from each of the designs in the appendix
How it addresses the problem
Why it is designed the way it is
What differences there are to the designs

Mention the magic number seven / white space is not your enemy stuff White Space is Not Your Enemy: [8]

A The magical number seven, plus or minus two: [17] - only put a certain number of things on the screen at once

Could I have the menu please? An eye tracking study of design conventions [15] - how navigation and elements on the page generally start on the left because that's where people (in western culture at least) will look first

Colour appeal in website design within and across cultures: A multi-method evaluation [6] - how colours have been used to group actions - green is pos, red is neg

Interface design technique to simplify and declutter your interfaces [7] - icons, placeholders, controls on demand, hovers and modals

McDougall - uses are more likely to 'like' icons if they recognise them, which is why using well known icons was important, hence fontawesome/glyphicons [16]

5.1.1 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[19], and the golden rules of interface design identified by Ben Schneiderman[24]. The purpose of this section is to highlight how the interface conforms to these metrics, and why they are important within the context of the application.

These have been grouped total, and the specific heuristics being referenced as in brackets, with the initials of the author, and the number of that heuristic. - these heuristics are listed in appendix D //in the format jakobs hueristics - JN1 - abc, JN2- def, ...

Visibility and Feedback (JN1, BS3)
User should be in full control (JN3, JN7, BS4, BS7, BS8)
Consistency (JN4, BS1)
Error prevention and Recovery (JN5, JN9, BS5, BS6)
Cognitive Load (JN6, BS8)
Aesthetic and minimalist design (JN10)
Match between system and the real world (JN2)

5.2 Navigation/Control Flow Design

In the final design of the system, there were x main sections, which have been listed below:

1. Route discovery - what it is
2. Route listing - what it is

3. Route detail - what it is
4. Route creation - what it is
5. Profile page - what it is
6. Admin page - what it is
7. Login/Signup - what it is

when desigining the website, it's important to think about how users will travel through the site, depending on what their goals are. For niceway.to, there will be two main user groups: those with accounts (exports - activity and regularly contribute), and those without accounts (travellers, most using the search), whose paths would be different. the general path for non use,as shown in figure y is landing page, results, then detail page. This is why they are designed as they are: on search, they want to search, so box is very visible, like google, then results - clearly laid out results with ability to re-search if mistakes, and then detail page, display everything about the route, but ability to look deeper if user so wishes

the general path for the expert user would be different. generally would involve logging in and checking social interaction on their owns, and routes that they follow, as well as: general path for expert - login in, look at my account, check out new social interactions on their routes, look at friends routes, create a new route, update skins. diagram shows this. how does design facilitate this?

of course some users may not follow this path, and it is not the only bath. user has freedom to traverse to essentially any page at any point (excluding logged in only pages, and the search results to details page). important for promoting freedom and a sense of control within the system. The user is free to decide how they wish to travel through the system, and this means they are free to traverse forward or backward through the system to make any alterations, or fix any mistakes they may have made.

5.3 Internal Design

using a model view controller design pattern because of x, y, z advantages. figure 10 shows this interaction. when a page is requested, we hit the controller, this processes stuff, makes some db calls using the models (which abstract the db), then we pass the data to the view and display this. this means the view never talks directly to the data. to make pages interactive / updatable, we use ajax calls to request more data or specific actins (defined in the contorller). at this point, it was decided to use php with the zend framework (reasons for which are described in section 6.1)which helps to enforce this model

diagram of db, model, controller, view, ajax, client and server, base controller subclasses and modelfactory subclass

6 Software Implementation

In this section, the actual implementation of the software has been detailed, including: what tools were used in the implementation, how the software was implemented, and any issues that were encountered during the implementation process.

- Screenshots of initial designs + justifications
- Screenshots of final designs + justifications + reasons for changes
- Screenshots of actual final system + reasons for changes

6.1 Key Implementation Decisions

- From the background research section, list all the technology I chose to use and why

6.2 Implementation Methodology

- To help manage the implementation of such a large piece of software, the adoption of some methodology was necessary. It was decided that the best methodology would be an agile one, with heavy use of Kanban, using the tips laid out by Henrik Kniberg [11]. In order to accomplish this, at the beginning of the implementation stage, after the requirements specification had been detailed, the entire project was split into user stories. Each of these stories detailed a specific action that a user of the system would be able to accomplish, along with how long it should take to implement, how important it was, and a way of testing its completion. These stories were then organised onto a digital Kanban Board, using a service called Trello[7].

  Each week, a set of tasks would be selected to be worked on for that week. The amount of tasks selected would be dependent on how much was completed, on average, in the weeks before, so that reasonable estimates could be made (obviously excluding the first few weeks). This ensured a decent portion of work was being completed per week, and that progress was constant. During the week, tasks would be selected from the available pool, prioritising those that were prerequisites of others, or had a high importance, and would then be worked on until completion. After the completion of a task, a new task would be selected, and work would begin on this. This was an extremely effective method of managing the implementation, as any small tasks that were necessary could be added to the board, and there was an assurance they would eventually be completed, and nothing would be overlooked. It has also been shown that it is much easier to reach goals if they have been written down [28], which a Kanban Board was the perfect tool for.

  Also mention weekly meetings with max and use of Gantt chart. Also mention this is what I did at work and in my last diss and found it the best way to work for me?

full example of taking a requirement, making story, splitting into sub tasks (in table or screenshot format)

[7] `http://www.trello.com`

6.3 Implementation of System Components

- Potentially don't need this
- Do last, look at last year's diss

6.4 Problems Encountered

- Look through problem log document and pick out key things, especially those with lessons
- What happened / what this affected / how the project was affected / what I would do differently / why it happened
- problem: lack of unit tests meant that the addition of features would break others. especially silly with the support Zend has for unit tests

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

8 Evaluation of the Project

8.1 User Feedback Testing

An important part of evaluating the usability and accessibility of any software system is to have real world users attempt to actually use it - The usability engineering life cycle [18]

- user feedback / questionnaire / focus group / test users? + their feedback
- the aim of these tests is to find **usability problems**
- RITE method for discovering bugs

8.2 Successes and Limitations of the Project

- As a result of the test...
- +
- +
- +
- - creation page was lacking in features
- - routing unreliable at times, because of using free service
- -

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

- 2-4 bigs things that I would do next time (either changing something I did, or adding/removing something)
- increased scenicness between points (atm it connect points with the shortest ath, future shoudl do most scenic ;))

11 Summary & Personal Evaluation

- Personally, I feel as thought 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: 01/10/2015.

[2] MADMAPs website. `http://www.madmaps.net/`. Accessed: 01/10/2015.

[3] My Scenic Drives website. `https://www.myscenicdrives.com/`. Accessed: 01/10/2015.

[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] Dianne Cyr, Milena Head, and Hector Larios. Colour appeal in website design within and across cultures: A multi-method evaluation. *International journal of human-computer studies*, 68(1):1–21, 2010.

[7] Webdesigner Depot. Interface design technique to simplify and declutter your interfaces, 2014.

[8] Kim Golombisky and Rebecca Hagen. *White Space is Not Your Enemy: A Beginner's Guide to Communicating Visually Through Graphic, Web & Multimedia Design*. CRC Press, 2013.

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

[10] Matthew Joint. Road rage. 1995.

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] Siné JP McDougall and Irene Reppa. Why do i like it? the relationships between icon characteristics, user performance and aesthetic appeal. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 52, pages 1257–1261. SAGE Publications, 2008.

[17] 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.

[18] Jakob Nielsen. The usability engineering life cycle. *Computer*, 25(3):12–22, 1992.

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

[20] 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.

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

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

[23] 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.

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

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

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

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

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

[29] Ping Zhang and Gisela M Von Dran. Satisfiers and dissatisfiers: A two-factor model for website design and evaluation. *Journal of the American society for information science*, 51(14):1253–1268, 2000.

A Appendix