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 | 1,030

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 What is Scenic Route Mapping 3

3.2 Existing Systems 3

3.3 Platforms and Tools 3

3.3.1 System Back End 4

3.3.2 Front End Programming Language 4

3.3.3 Front End Design Framework 5

4 System Specification **6**

4.1 Functional Requirements 6

5 System Designs **9**

5.1 UI Design 9

5.2 Navigation/Control Flow Design 9

5.3 Internal Design 9

6 Software Implementation **10**

6.1 Key Implementation Decisions 10

6.2 Implementation Methodology 10

6.3 Detailed Description of the User Interface 10

6.4 Implementation of System Components 11

6.5 Problems Encountered 11

7 External Aspect **12**

8 Evaluation of the Project **13**

8.1 Functional Testing 13

8.2 Non-Functional Testing 13

8.3 User Feedback Testing 13

8.4 Successes and Limitations of the Project 13

9 Further Work **14**

10 Summary & Personal Evaluation **15**

A Appendix **17**

1 Introduction

- Powerful opener, similar to proposal opener
- Mention other software briefly and why it's bad
- Main objective of this project, and how it will be achieved
- Where this project will not focus, if any

2 Motivation

- Over arching motivation - **what is being solved?**
- Mention previous work for client
- Personal ties to the project?
- What is good about this project existing?
- Mention good HCI (see yr 3 diss)

3 Background Information & Research

3.1 What is Scenic Route Mapping

- Not sure about this section, just kept it because I had 'What is fuzzy logic' in my diss last year

3.2 Existing Systems

- Since routing/mapping began.. there has been many pieces of software.. here are a few

SW X

- What is it
- What is does
- Why it's good
- Why it's not

SW Y

- What is it
- What is does
- Why it's good
- Why it's not

SW Z

- What is it
- What is does
- Why it's good
- Why it's not

3.3 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...

3.3.1 System Back End

Backend X

- What is it
- What is does
- Why it's good
- Why it's not

Backend Y

- What is it
- What is does
- Why it's good
- Why it's not

Backend Z

- What is it
- What is does
- Why it's good
- Why it's not

3.3.2 Front End Programming Language

- not sure if this is overly necessary. Could condense this language + framework into one?

Front End X

- What is it
- What is does
- Why it's good
- Why it's not

Front End Y

- What is it
- What is does
- Why it's good
- Why it's not

3.3.3 Front End Design Framework

- not sure if this is overly necessary. Could condense this language + framework into one?

Front End X

- What is it
- What is does
- Why it's good
- Why it's not

Front End Y

- What is it
- What is does
- Why it's good
- Why it's not

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 the client 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 submitted 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 should be able to decide on the visibility of this route
3. The user should be able to interact socially with the route, including:
    - 3.1. Comment on public routes
    - 3.2. Recommend alternative routes
    - 3.3. Share routes to external social media websites
4. Users should be able to create an account with, for example, name, email and location
5. There should be administrative users who have extra functionality, including:
    - 5.1. Managing users
        - 5.1.1. Delete users
        - 5.1.2. Update users
        - 5.1.3. Create users
    - 5.2. Managing routes
        - 5.2.1. Delete routes
        - 5.2.2. Update routes
    - 5.3. Delete comments
    - 5.4. Make announcements
    - 5.5. Make backups of the website in a standard, compliant, free and open format
    - 5.6. De-authorize active sessions
    - 5.7. Lock the site and prevent access
6. Users should be able to export their routes to a FOSS
7. Users should be able to make a copy of other user's public routes and edit them
8. There should be a route editor component which allows the users to specify a route
9. The user should be able to log into their account and:
    - 9.1. Access and change personal information
    - 9.2. Access and edit their submitted routes

Accessibility
The proposed system is to be made available entirely on-line, allowing anyone with internet access, and a computer, to use the system. As the application will be web-based, the user is not required to download anything to start using it, which will make it more accessible for the user, and easier for them to get started using 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 relatively 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. Help will be available within the system, so the user can received help and directions, without having to leave the system and check some external documentation. The project will be developed as a fully-responsive web application, meaning mobile devices will be fully supported.

Maintainability & Documentation
The system must be exceptionally 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. The system itself will have help available to the user, so that if they are confused, they can be guided in the right direction.

Quality
As this system is to be used externally, and will be a representation of both myself, and the client, there are several quality issues that must be addressed. The system must be built so that it is robust, and works as the user expects, but it must also contain as few bugs as possible. Any bugs that are identified should be reportable to the maintainer, and be fixed as soon as possible.

The quality of the code must also be considered. In this regard, the code will be written in a modular way, and useful comments will be provided to highlight the purpose of each function, and to 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 would be fairly minimal anyway, as it is mostly loading only JavaScript and HTML5. The only true computation takes place on the server, and thus would not be a concern for the user.

The ability to load and save files potentially causes a problem, but only if the user has very little hard drive space, and attempts to save an extremely large file from the system. Unfortunately, there is nothing that can be done about this, but even very large systems 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 (developed using HTML5, CSS3, and JavaScript), it is difficult for it not to be cross platform compatible. However, there are still a few issues that may have to be dealt with, especially when looking at different browsers that the user may be using. For this reason, research into how the system performs on different browsers will be important, so that it can be assured that any user using any browser will have full access to the system, as it was meant to be.

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, compliant, 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 anything goes wrong.

5 System Designs

In this section, all of the design aspects of this system have been detailed and justified.

- containing a comprehensive description of the design chosen, how it addresses the problem, and why it is designed the way it is.

5.1 UI Design

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

5.2 Navigation/Control Flow Design

- The generally expected path for a user to take through the system + picture
- Explain how design facilitates this
- Explain navigation allowing random access

5.3 Internal Design

- Models/Controller/Views
- Languages
- Flow of data from database -> view (don't forget AJAX calls)
- See image from last year diss

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 [2]. 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[1].

  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 [3], 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?

6.3 Detailed Description of the User Interface

- In this section, each individual screen of the system has been displayed, along with a detailed explanation of why it is effective, and why it has been implemented as it has.

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

- Potentially don't need this? Or a slimmed down version

6.4 Implementation of System Components

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

6.5 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

7 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...?)

8 Evaluation of the Project

8.1 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 [1]). A complete listing of all the tests conducted, and their results, can be found in appendix **??**.

8.2 Non-Functional Testing

- Look at non-functional requirements and talk about if they were met

8.3 User Feedback Testing

- user feedback / questionnaire / focus group / test users? + their feedback

8.4 Successes and Limitations of the Project

- As a result of the test...
- x was good
- y was bad

9 Further Work

- 2-4 bigs things that I would do next time (either changing something I did, or adding/removing something)

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

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

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

A Appendix