HMR - Asynchronous Boundary Control
Final Report

by

G. H. J. Geneste
X. Zonneveld

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Computer Science

at Delft University of Technology,
to be defended publicly on Friday January 30, 2015 at 17:00.

Supervisors:  
Dr. ir. A. J. H. Hidders  
Dr. ir. Z. Duge

Thesis committee:  
Dr. ir. A. J. H. Hidders,  
Dr. ir. A. M. Larson,  
Dr. ir. Z. Duge,  
TU Delft  
TU Delft  
Horseshoe Meadows Ranch

An electronic version of this thesis is available at http://repository.tudelft.nl/.
Abstract

The Horseshoe Meadows Ranch has evolved to a hospitality unit that caters to horse owners who are looking for lodging and recreation at the same site as the stable that serves their horses. With access to more than 1618 hectares of open unimproved forestland, the guests of the ranch are having difficulties navigating through the wild scenic, majestic mountains and untamed rivers since many trails and single track features are omitted. Two students from the Delft University of Technology who are in the final phase of the bachelor Computer Science, are asked to develop a methodology with which the trails and features of the unimproved forestland can be mapped in a delivery system. The delivery system should enable the guests of the Horseshoe Meadows Ranch to navigate through the wilderness of Oregon on their own. This report describes the establishment of the launched prototype during 18 weeks of development. The details of the research and implementation process will be discussed throughout the report.
This document concludes the work of a Computer Science Bachelor thesis at the Delft University of Technology. The research-phase of the project was executed at the Horseshoe Meadows Ranch in Sisters, OR, USA and the implementation at the Delft University of Technology in the Netherlands. The authors of the document have developed a methodology for the Horseshoe Meadows Ranch to assist their guests in navigating through the unimproved forestland around Sisters in a more automated way.

We would like to offer our gratitude to the Horseshoe Meadows Ranch for the given opportunity to execute this project on an international level. We also like to thank our mentor A.J.H. Hidders and Bachelor Project Coordinator M.A. Larson for their positive support, input and guidance throughout the project.

G. H. J. Geneste  
X. Zonneveld  
Delft, Zuid-Holland, The Netherlands, January 2015
6.4.3 Advanced search ................................................. 25
6.4.4 Testing ............................................................ 25

A Project Analysis ......................................................... 27
B Product Vision ......................................................... 35
C SIG evaluation .......................................................... 43

Bibliography .............................................................. 45
Introduction

For this Bachelor of Science project a methodology was developed in the form of a web application that meets the requirements of the product description. The product was finished in 18 weeks, divided in two phases. The first phase was executed at the company of our client Dr. Duge where the global requirements were determined as well as detailed functional and quality requirements. A Graphical User Interface Mockup was implemented and tested accordingly and the system design was made. The second phase was carried out in the Netherlands and was comprised of implementing the application.

This document describes all the different phases of the project. Preliminary background information will be given in the introduction, this includes a brief description of the company, our client, the assignment as well as the causes that led to the project assignment.

Chapter 2 describes how the team approached the assignment and explains the used methodologies. Chapter 3 of the document goes into detail on the requirements of the assignment. This includes the orientation of our project and the requirements will be determined. Chapter 4 focuses on the product design. The system is divided in three modules and each module will be discussed separately. Chapter 5 presents the implementation plan as well as all the methodologies that are used to ensure the software quality throughout the project. The choices of methodologies will be explained along with the regularity of its execution. Chapter 6 evaluates the project in terms of process and product. The team will also present a recommendation for future work to improve the current prototype.

1.1. Company Description

The Horseshoe Meadows Ranch (hereafter: HMR) is a working horse facility, founded in 2007 by the General Manager of HMR Zeke Duge and is located in Sisters, Oregon, USA. The business has evolved to become a hospitality unit that caters to horse owners who are looking for lodging and recreation at the same site as the stable that serves their horses. Central Oregon is a destination location with four season attractions, wild scenic and majestic mountains and untamed rivers.

1.2. The Client

The client of the project is the General Manager of the Horseshoe Meadows Ranch. Dr. Duge has a Ph.D in Computer Science from Madison University and was Senior Director of Oracle Corporation, Chief Information Officer of West Marine (NASDAQ) and General Manager NPTI / Nissan USA.

1.3. Problem Description

The guests of HMR and the general public has access to more than 1618 hectares of open unimproved forestland. The company spends a considerable amount of time explaining the guests where to go by bicycle, foot, horseback or approved motorised vehicle. Additionally if the guest becomes disoriented it can be problematic to identify the area in which they are.
The currently available description of the tracks, unimproved roads and trails are the very detailed maps of the United States Geologic Service (2015). These maps are large scale, expensive and only reflect government features and hence are not user-friendly. Additionally many trails and single track features are omitted.

In response to the lack of user-friendly navigational methods through the unimproved forestland, Dr. Duge has asked the team to develop a methodology whereby the trails and features can be mapped. The development of a delivery system that will enable the user to use readily available assets to be self-reliant with regard to opportunities (e.g. routes and distances) would be of great value to the guests of the HMR as well as the general public. The system would become available to all to use without any additional costs. This would also help preserve the land for recreational use instead of it being repurposed.
2 Approach

The project team consists of two members and is responsible for the requirements analysis, implementation and delivery of the product. To ensure the team will follow a controlled process, a choice in the available design processes needs to be made. Due to the size of the project team it is necessary to apply a few adjustments to conventional methodologies. The choice of methodologies and tools will be outlined in section Methodology. Section Deliverables will give a brief description of the deliverables that have been produced during the project.

2.1. Methodology

There are several methodologies available that are applicable to the development of this project. The team chose between a selection of general known methodologies (Pezze and Young, 2008) such as the Waterfall Model, Extreme Programming, feature driven development or test driven development. Each of these methods have their own qualities, and a well founded consideration is needed before the decision is made.

2.1.1. Agile Scrum

When this project initiated, only a very generic project description was available. It became clear to the project team that more detailed specifications had to be described during the design process. These specifications were unavailable before the project started because the client was not sure about the details of the project. The client indicated that the advice of the project team would be necessary and it would not be unrealistic that some adjustments will be made during the process. Based on this information the team has decided to work with the Agile Scrum methodology (Rubin, 2012) because of its incredibly flexible design model. With this method the team would be able to respond to changes in requirements effectively and rapidly.

The most desirable size for a Scrum Development Team is three to nine members (ScrumGuides.org, 2013), therefore the team had to adapt the regular Scrum process to decrease the possibilities of potential downsides. The team decided to start with implementing a shippable product by the incremental Scrum Methodology after the research and product design was done.

This resulted in a two-phased project where the first phase was mainly focused on orientation and determining the design and scope of the project. A strong basis and more specific vision was formed because of the created documents during this phase. The documents Project Analysis and Product Vision also contain the foundation of most decisions that have been made throughout the project and are therefore very important as a reference for the next chapters.

During the second phase more conventional scrum events were executed to maintain an agile development process. The planned milestones per sprint were based on the produced product backlog and the quality of work was kept in balance with the productivity by setting deadlines for each sprint.
2.1.2. **Version Control System**

Serious software projects of this size need a Version Control System (hereafter VCS) to keep a maintainable system. A few important reasons for choosing a VCS was that the developers will be able to write code concurrently, facilitates distributed development and encourages a clean internal design. Different VCS's are available but the software team was most familiar with Subversion ([Collins-Sussman *et al.*, 2011]) and GIT ([Loeliger and McCullough, 2012](#)). GIT was preferred above Subversion because it allows version control of changes done locally. This concedes the user to revision early drafts of work without directly publishing it to other team members.

2.2. **Deliverables**

Several deliverables have been submitted to the client and mentor throughout the project. A few deliverables served primarily as a communication method between the client and the development team to assure a common understanding. The purpose of the deliverables are described in the following two sections.

2.2.1. **Documents**

In general, agile development methods emphasise face-to-face communication over documentation ([Highsmith and Cockburn, 2001](#)). However, the team has chosen to create a set of documents. This decision is based on the following. First of all, the project was divided in two main phases with 10 weeks between those phases. In this case it is very important to be able to recall the work that was done in the past to reduce the amount of duplicated work. Secondly, according to several studies, it is even questionable if the maintainability of an agile based project is secured ([Clements *et al.*, 2003](#)). Therefore the team has decided to create a few documents which supported the team to find an answer to specific (already researched) questions and could help future developers explain certain decisions. The following official documents have been created during the project:

- **Product Vision** Explores the desired scope of the project and the requirements analysis is documented.
- **Product Analysis** Informs the reader about the research that has been done and explains the decisions that have been made beforehand.
- **Final Report** Describes the most important subjects of the project such as process, research and implementation.

2.2.2. **Other Deliverables**

During the first phase, it was advisable to deliver a General User Interface Mockup (hereafter GUI Mockup) to the client before the implementation of the project would start. This mockup would show the client a basis of the interface for the requested methodology. The team needed to develop the interface in close collaboration with the client because the GUI is very important for the success of a web-based methodology. This GUI Mockup was delivered during the first phase of the project, adjustments have been made since the first delivery of the GUI Mockup as a result of client feedback and test results. A more detailed explanation on this subject can be found in [Design](#).

Other deliverables are all software based. According to the General Guide of this project the team is required to submit the source code of the project twice to the Software Improvement Group ([SIG, 2015](#)). The submissions will both be treated as a release. This means that each release has to meet the requirements as defined in the Definition of Done, written in the product vision **Product Vision**.
As a first step in the process to develop a methodology for the client, the functional- and nonfunctional requirements and constraints for the prototype will be elucidated. It is important to translate the needs of the client into a development plan. Two different methods have been used to specify the requirements for this project. In section 3.1 the assignment is described in more detail, section 3.2 describes the most important sections of the Product Vision and the requirements are presented in section 3.3 which is a combination of the MoSCoW model and the Product Backlog.

3.1. Assignment
In this project, a web application will be realised to which the guests of the HMR and general public have access. The new application should visualise the available trails and the user needs to be able to select one of these trails for further use.

The client describes in his proposal that the extent of technological involvement will be left to the students. However, the public interface will be a joint decision of the students and the company, the operation and design will rest solely with the students.

The final product will offer the features that are described in section reference:section and should be considered a "prototype".

3.2. Product Vision
The Product Vision has been extensively covered in one of the previous deliverables of the project. The Product Vision is, according to Schwaber (2004, p68), "the vision that describes why the product is being undertaken and what the desired end state is." This section outlines clients’ expectations of the product on a very abstract level. It should not describe specific features or solutions to problems. On the basis of the Product Vision, Figure 3.1 has been put together to envision the product.

The key features of the product are determined according to the Product Vision. These features describe the product more concretely in order to retrieve specific specifications. The key features are:

- The product visualises all available trails within the selected ares.
- The product will be web-based enabling it to be used on different devices.
- The product connects neighbours who share the interest for outdoor activities by sending invitations through the application.
3.3. Product Roadmap

In order to give the client a better insight in the project a roadmap has been developed. The roadmap is made up of the MoSCoW model and Product Backlog and gives a high level specification of the targeted releases.

3.3.1. MoSCoW model

To determine the exact requirements more accurately, a prioritisation method was needed. The first prioritisation method used for this project was the MoSCoW model (Hatton, 2008). The model divides all the clients wishes, that have been transformed into basic requirements, in four different groups:

- **Must Have:** Core functionality of the system, without these features the system will not work.
- **Should Have:** Features the system needs to function properly, but could do without.
- **Could Have:** Features that will probably not make it in the system, but that would be great for a follow up project.
- **Won’t Have:** Features that will not be implemented in the system, even if there is enough time to do so.

As represented in the Product Vision, chapter three the requirements have been determined and prioritised as can be seen in Table 3.1.
### 3.3. Product Roadmap

#### MoSCoW Feature

**Must Have**
- Search for route
- Display routes in map
- Normal & Mobile layout

**Should Have**
- Save routes (unique identifier)
- Sent invitation through Social Media
- Make printout of selected route
- Display route on smart phone

**Could Have**
- System to add new routes/datapoints
- Show extra features of routes (viewpoints)
- Return current location

**Won’t Have**
- Login system
- Collection of personal data
- Fees or collection of money

<table>
<thead>
<tr>
<th>MoSCoW</th>
<th>Feature</th>
</tr>
</thead>
</table>
| **Must Have** | Search for route  
|             | Display routes in map  
|             | Normal & Mobile layout |
| **Should Have** | Save routes (unique identifier)  
|             | Sent invitation through Social Media  
|             | Make printout of selected route  
|             | Display route on smart phone |
| **Could Have** | System to add new routes/datapoints  
|             | Show extra features of routes (viewpoints)  
|             | Return current location |
| **Won’t Have** | Login system  
|             | Collection of personal data  
|             | Fees or collection of money |

Table 3.1: MoSCow model

#### 3.3.2. Product Backlog

The product backlog is another prioritisation method. The table that represents the product backlog contains detailed features which are based on the requirements of the MoSCoW model and the specified user stories **reference to productvision**. The features are ranked according to Fibonacci’s numbers $1, 2, 3, 5, ..., ((n-1)(n-2))$, where the most important feature is ranked one and the least $(n-1)+(n-2)$.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route Database</td>
<td>Create a database with several routes</td>
<td>1</td>
</tr>
<tr>
<td>Search Interface</td>
<td>Create web-interface to search for different routes</td>
<td>2</td>
</tr>
<tr>
<td>Map Interface</td>
<td>Display routes into map for navigation</td>
<td>2</td>
</tr>
<tr>
<td>Mobile interface</td>
<td>Make the current interface mobile compatible for several devices</td>
<td>3</td>
</tr>
<tr>
<td>Unique Identifier</td>
<td>Identify routes with a unique URL</td>
<td>5</td>
</tr>
<tr>
<td>Invite Interface</td>
<td>Create web-interface which enables the user to invite others</td>
<td>5</td>
</tr>
<tr>
<td>Print Route</td>
<td>Create a method that converts selected route into PDF for printout</td>
<td>8</td>
</tr>
<tr>
<td>Select Speed</td>
<td>Select speed to calculate the estimated travel time more accurately</td>
<td>8</td>
</tr>
<tr>
<td>Viewpoints</td>
<td>Show the special viewpoints per trip</td>
<td>13</td>
</tr>
<tr>
<td>Export GPS Data</td>
<td>Export GPS data to a handheld GPS device.</td>
<td>13</td>
</tr>
<tr>
<td>GPS Integration</td>
<td>Enable location tracking during navigation</td>
<td>21</td>
</tr>
<tr>
<td>Add routes</td>
<td>Create user-friendly interface to add routes</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 3.2: Product Backlog
Design

When the requirements are determined the team has to design the architecture of the methodology. According to the key features that have been set in the Product Vision, the methodology will be a web-based system. With this in mind, the team decomposed the system into three modules: the Computation Module, the Database Module, and the Graphical User Interface Module and will be discussed in the following sections.

The design of the product is based on different design principles. The development team has agreed with the client which design principles are most important for the project and these will be taken into account throughout the project. The most important design principles are as follows:

- **Robustness**
  The methodology should be able to handle anomalous situations. This means that the application should act normal for as many situations as possible even if the user input is faulty.

- **Portability**
  During the requirements analysis the client requested a method that has to be available on different platforms. This means that the development team needs to take the portability design principle into account during the design and implementation of the project. It will also require specific tests to validate the application on different systems that could for example differ in screen size, platform or browser.

- **Maintainability**
  The project in general is very broad and could be extended with many extra features. The client is interested to extend the project in the future and it is possible that the size of the project will increase significantly. Therefore the team concluded that it is important to keep the system maintainable in order to be able to possibly extend and maintain or repair the design efficiently and economically.

- **Usability**
  The quality of a user’s experience is a very important success component for this project. The usability of the application depends on many different factors such as the memorability of the application, its intuitive design and ease of learning and use.

4.1. **Computation Module**

The computation model describes the process that alters the data from the given database in order to calculate the possible routes according to the users’ preconditions. The most complex part of the computation module is to determine the route selection algorithm. Before the computation of available trails can be made, an input needs to be given. The input will be given via several possible procedures like selecting a location on the map and asking for routes within a specific radius. A more detailed description of this matter is given in chapter 5. In this section the analysis of the computation of available routes and its conclusion will be given.
4.1.1. Analysis
There are many different algorithms available to find a route from A (start) to B (destination). The most common practise is to apply Dijkstra’s algorithm (Dijkstra, 1959), Bellman–Ford algorithm (Ford, 1956) (Bellman, 1958) or Floyd–Warshall algorithm (Floyd, 1962) to find the shortest path. However, looking at the problem description, the user is not necessarily looking for the shortest path. A well suited solution would be to compute a Hamiltonian Path, this solution however is a NP-complete problem as proven by Karp (1972). It is therefore in most cases impossible to compute an optimal solution within polynomial time. There exist approximation algorithms that are able to approximate the required solution for the problem. One of these algorithms is called ant colony and was first proposed by Dorigo and Gambardella (1997). The general idea of the algorithm is that a random solution will be found during the first iteration, and should be improved in the next iterations of the algorithm. The longer the algorithm runs, the more exact it approaches the optimal solution. Besides the complexity of this algorithm, it also needs to be adjusted very accurately in order to find a suitable balance between the weight of the computation and accuracy of the solution. According to the design principles the team has to focus on a robust, maintainable, usable and portable solution. The team has therefore concluded that the suggested solutions do not lie within the scope of the project. A more feasible solution to retrieve available routes is one which is less dynamic but significantly less complex.

4.1.2. Conclusion
Designing a solution to compute the available trails every time a search query is executed would make the search results more dynamic. However, since finding all possible n-tuples (routes) such that it satisfies certain relations like the users preferences, is considered a difficult problem to solve efficiently due to its NP-completeness property (Mackworth and Freuder, 1985). Thus implementing a corresponding algorithm that uses a reasonable amount of processing power without decreasing the usability of the product because of the time it takes before any result is shown, seems to be unfeasible within the given time frame. Therefore the team has decided to implement a prototype conform the design principles during this project with a computationally less complex technique. A less complex technique, that alters the data from the given database in order to calculate the possible routes according to the users preconditions, may not be as elegant as the solution the team has been looking for but suits the scope of the project better. A pre-computed approach will solve many of the current problems regarding the computational module. For this reason the team decided to pre-compute the routes locally and store the properties of these routes in the database.

4.2. Data Module
The data module is an important part of the application and has been discussed thoroughly in the previous deliverables. The sections below will inform the reader about the research that has been done but will mainly focus on the conclusions. For a more detailed explanation it is advisable to read the Project Analysis, chapter 2. The Data Module needs to represent the trails that will be visualised in the application. This requires the application to consist of a well designed database which is briefly discussed in subsection 4.2.3. The collection of the data, as well as the methodology that needs to be used to visualise the trails are represented in subsection 4.2.1 and subsection 4.2.2 respectively.

4.2.1. Collected Data
To be able to represent the trails in a database, a digital form of the trails has to be created. There are several methods available, however an extensive research to determine the most suitable method did not lie within the scope of this project. The team collected and processed a few data-samples and concluded that the Keyhole Markup Language (.kml) files fit the project well enough to continue with this format. The .kml file format is used to display geographic data and contains features such as placemarks and descriptions. The coordinates are represented in longitude (−180 ≤ longitude ≤ 180), latitude (−90 ≤ latitude ≤ 90) and optionally altitude (in meters above sea level) (Google Inc., 2014a). An automated method has been implemented in Java to extract useful information from the file. The
program iterates through the `.kml`-file to collect the information based on the node-names and directly substitutes it into SQL statements for filling in the database.

Figure 4.1: Piece of the representation of the collected GPS data (.kml file).

### 4.2.2. Route Visualisation

There are currently numerous procedures available to represent geographic data in a map. This methodology should, as requested by the client, represent the available trails in a user-friendly manner. Therefore the team has compared the most popular web map applications to decide which API is most appropriate for this project.

Further study (Steiniger and Hunter, 2013) shows that Google Maps and OpenLayers were by far the most popular and promising applications. The KML based vector data integrates more easily with the Google Maps API compared to the OpenLayers API because of the available infrastructure in Google Maps for KML integration. The OpenLayers API however, mainly stood out because of its Creative Commons licence (Creative Commons, 2015) which means that the API is free for non-commercial use. Overall the Google Maps API had a better fit with the design principles and became therefore the chosen methodology to visualise the collected data into a user friendly manner.

### 4.2.3. Database

The main functionality of the database is storing the collected data which can be retrieved by a function call for visualising the trails in the application. The data is collected by recording a GPS track while a user moves along the trail. The GPS track is a polygonal line with a single start and end point (which could be connected) and it is desirable when the visualisation of the trail is alike. In other words, the most intuitive visualisation of the trails is a graph with vertices and edges where every edge trail segment represents, and every vertex a trail junction. Each trail segment in the graph is in reality a set of connected GPS points.

The development team has carried out extensive research to determine the most suitable database model for the project. The two considered database models are the Graph Database Model (Graves et al., 1995) and the Relational Database Model (Codd, 1970).

The most important differences between these two models is that the graph model performs exceedingly better in speed for specific query computations. While it is possible to represent a graph with a relational database, one must take into account that certain complex graphs will be easier to represent in graph databases. However, the speed of executing retrieval queries of graph databases is only optimised for full text-indexing whereas the relational database is more efficient when other types of data need to be retrieved.

Another important property of the relational database compared to the graph database is the storage efficiency: The Relational Model uses in general about half the size of the corresponding Graph Model (Vicknair et al., 2010).

Therefore the development team has decided to use a relational database because it applies better to the design principles of this project since the usability of Relational Database is greater than the usability of a Graph Database due to its complexity.
4.2.4. Database Design

The current database design needs to store several elements in order to be able to visualise the trails in the application. Please refer to chapter 5 for a more detailed explanation of the implementation of the database. During the database design, several options have been considered. Although the Relational Database was more efficient from different perspectives, it is still important to develop a database which is not unnecessarily complex in structure or avoidably heavy in its size. The team has developed several schemes for the database and concluded that the scheme presented in figure Figure 4.2 maintains the best balance between complexity and size.

![Database schema diagram](image)

The database scheme is constructed by representing the data in a more abstract form, this form is based on the properties of a regular graph that could represent the collected data. A regular graph contains edges and nodes, and a path is a collection of edges from two nodes within the graph. The data that is extracted from the .kml-file is a set of GPS-points and is stored in the gps-table. A row should be inserted in the edge-table when two GPS-points are connected. The complete path within a trail is represented as a set of edges.

The current database scheme contains more properties compared to a regular graph to avoid data-loss since the data has to be transformed back to an XML-scheme to visualise the routes properly. Another property is the trail-table. The trail-table adds an extra dimension to the graph, and represents the possible trails as a whole by combining several paths. To be able to retrieve a trail that has previously been selected and saved by the user, the saved_trails table is created. The two tables poi and poi_translation have been added to the database since the client suggested that it could be interesting to have points of interest that contain additional information about the trails like beautiful overviews or waterfalls. All these tables together form the designed database and contains the required information to represent trails as requested.
4.3. Graphical User Interface Module

One of the key elements to improve the usability of the application is via the Graphical User Interface: Users may spend a longer time at the website, revisit or even recommend the website when the user satisfaction is high and this will therefore contribute to the success of the project.

An empirical study is conducted to find the motivational components which could improve the satisfaction factor. The conclusions drawn from this study have been collected and resulted in an initial GUI design and is presented in subsection 4.3.1.

The purpose of the empirical study was primarily to establish an initial design and through a variety of techniques improvements have been made to this initial design. These techniques tested the user experience and the outcome will be presented in subsection 4.3.2 along with the changes that resulted from these test techniques.

The work is concluded in subsection 4.3.3, where the final design will be presented as well.

4.3.1. Initial Design

The user interface must provide access to the functions and features of an application. These functions and features have been deduced from the requirements analysis and especially from the user stories\(^1\). A list of user requirements are the result of gathering user information via these user stories and enable the developers to specify the buttons and forms that will be needed for a functional GUI.

Despite the many potential functionalities that could be added to the project, it is important to note that a lot of different selection criteria in a user interface may overwhelm the user (Miller, 1956a). Another important aspect that could increase the usability of the GUI is the addition of structure and can be achieved by grouping similar elements together (Williams, 2008). The initial design should therefore be simple and structured in order to cultivate repeat visits through an attractive interface.

Since users typically scan quickly for relevant information, the application has to emphasise its main functionality by centralising that feature. With these conclusions the initial design as expressed in Figure 4.3 was composed.

\[\text{Figure 4.3: First design of the user interface}\]

\(^1\)The user stories can be found in Appendix B of the Product Vision
4. Design

This initial design centralises the map representation because this is the main feature where the trails will be visualised. The buttons to share a selected route (print, download or share with social media) are grouped together on the upper right corner of the application to indicate that these functions are alike. The basic search functions are grouped together to categorise the similar features. To ensure the user will not be overwhelmed but also has enough options when it needs to find very specific routes, the advanced search-button is created. Extra options will appear when this button is clicked by the user.

This design is the first step to establish a user friendly website that is easy to use and understand. To improve the initial design, an iterative process has been carried out where the users experience was tested during every iteration.

4.3.2. User Experience

The design is evaluated by measuring the user experience, which results in an improvement of the application after each iteration. The user experience can be measured by conducting usability tests which in essence gathers the opinion of potential users. Since opinions differ per individual it is recommended to combine qualitative feedback with quantitative data to identify the average opinion. To gather quantitative data, several Click Tests and Five Second Tests have been conducted. To collect qualitative feedback multiple three minute verbal and screen capture responses have been recorded of numerous random users that are asked to “think out loud” their first impression and general opinion of the website. All the feedback was received from non-associated test participants varying in age, experience and gender.

Clicktest

A Click Test is a test where the test participant is asked a question on how he or she would perform a specific action. The user will answer by clicking on a general mockup of the interface and the click-actions are recorded. The result of each test is represented in a heat map, a graphical representation of the recorded clicks from the test participant. The tests have been conducted at www.UsabilityHub.com. All the services of this website are paid by credits, and these credits can be bought or obtained by performing tests of other users.

Figure 4.4: The result of a clicktest represented in a heatmap

Figure Figure 4.4 shows the result of a clicktest that has been performed during a iteration to improve the usability of the application.

One of the most remarkable tests results was retrieved when the mockup of the initial design was tested. This test had over 20 participants with an average age of 25-29 years old and 90% of the participants originated from the United States of America. The participants received the following question: “Where would you click to select an area and find a route there?”.
The remarkable outcome of this test was that more than 20% of the users seemed to be confused by the location of the clear map button. This button was located between the Find Route and Advanced Search. With this design, 20% of the users didn’t click the correct buttons to find a route and the average response time was over 40 seconds.

To improve the design, the clear map button was relocated and the same test was conducted for a second time. By comparing the results of the two tests, the team could conclude that the relocation of the clear map has improved the design a lot since the average response duration was decreased to 26 seconds and 100% of the participants clicked the correct areas.

Five Second Test
Another method to gather quantitative data about the user’s experience of the interface, is by conducting a Five Second Test. The Five Second Test shows a mockup of the design to the test participant for just five seconds. After the five seconds is up, the tester is asked a series of questions. This test was conducted every time a change was made to the design. The results of the tests indicated whether the quality of the design decreased or increased after the change. Every time the test was carried out, 30 responses were retrieved and the following questions have been asked to different participants after every design change:

- What do you think this page was about?
- What was the most prominent element on this page?
- Rate the quality of this page between 1 (worst) and 5 (best).

After the last iteration of the user experience test, the average result of the quality of the page was a solid four. 28 out of 30 users were able to describe the functionality of the web page by looking at the page for just five seconds and the most prominent element on the page was in 80% of the cases the map. No changes in the interface have been made after these results and therefore the results are still applicable to the current design.

User Test
To perform qualitative tests, several user tests have been conducted. A user test is great for examining how people interact with the application. Each task a user performs is followed by a related question to gather analytical data to support the study.

To reduce the interference of the developer as much as possible, since the developer could influence the test which results in a less effective test, the team has decided to use the service of the website usertesting.com. This website provides a five minute review of the website that will be recorded and sent by email. The first three reviews per month are free, if the user wishes to conduct more tests or likes to ask application-specific questions then the paid service could be used.

The tests results have been very useful every time the results were received. The recorded responses of the test participant gave big clues as to how people interact with the site and which areas cause people to pause, reflect or question different aspects of the site. Each test the same four questions have been asked:

- What is your first impression of this web page? What is this page for?
- What is the first thing you would like to do on this page?
- What stood out to you on this website?
- What, if anything, frustrated you about the site?

From the first few test results it became clear that the map module was overrepresented in the interface. Since the interface was tested on several devices by different users, it showed that on many devices the map was represented in such a way that it would cover almost 90% of the window. This resulted in a very unintuitive user interface because it was unclear to the user that a search-form and buttons existed underneath the map-module and people did not notice the scroll function of the website since scrolling would change the zoom-factor of the map.
The solution to this problem was by simply changing the size of the map in a way that it was still the center of the application but not overrepresented so users would be able to notice the other features of the website as well. Other improvements made to the design included the relocation of the print, share and download buttons. Instead of having the buttons always on the upper right corner of the website, they will now only appear when a trail result is being returned. In most of the test results, the users felt comfortable navigating through the website. The first impression of the web page often was very positive and the test participants were able to determine the functionality of the website. Another result that stood out was that the application’s behaviour was almost never unexpected. Therefore not only the usability of the website but the level of robustness of the interface was acceptable as well.

4.3.3. Final Design

The final design of the application is based on the initial design and evolved in the final design due to every improvement that was made per iteration. This design was evaluated by combining qualitative feedback with quantitative data. The combination of the test results helped the team to develop the design as it is today.

With this design, the user is able to find trails by clicking on the map or by entering an address in the search form. When the search result returns at least one trail, the user may select a trail and is able to share this via social media or to download or print the trail. When a user receives an URL with an unique identifier, the website will show the user this trail specifically. The functionality of the website is represented in a state diagram Figure 4.5. Depending on the actions a user could possibly carry out, the transition from one state to another will be made.

![Figure 4.5: A statediagram, representing all the possible functionalities based on the current state of the website.](image-url)
During the implementation phase the team started to follow the scrum-process more strictly. The product backlog and roadmap, became more important and have been used as a guidance for the implementation process. The Definition of Done (Schwaber, 2004) was used to check whether the work has been completed when the implementation of a feature, at the end of a sprint or release was done. After every implementation the feature, sprint or release had to meet its requirements and all the tests needed to pass before it was marked as completed.

This chapter describes the most important topics of the implementation process and informs the reader about the different tests that have been conducted during or after the implementation of a feature.

5.1. Implementation Plan

Having minimal experience with web development the team decided to start with completing the basic Google Maps Javascript API v3 (Google Developers, 2014) tutorials. From here the application has been extended to two separate systems communicating with each other using Asynchronous calls and callbacks.

5.1.1. Framework

Building a successful responsive website depends on many different factors. Nowadays there are several frameworks available to improve the quality of responsive websites. These frameworks contain methods to handle or support general functionalities of a website.

The team decided to use a few frameworks which allows to concentrate on the application itself rather than worrying about several functionalities that otherwise had to be written by the developers themselves. Existing frameworks enable the website to handle multiple browsers and mobile platforms dynamically. The downside of using a framework is that it could contain too many (unused) features which results in an unnecessarily high redundancy. As mobile platforms generally have less processing power it is important to keep in mind that the used frameworks have to be small in size and preferably usable without flash enabled.

Skel

A very important requirement of this project was its portability. The client requested an application which should be usable on different devices and browsers. In combination with the design principle robustness it would be very time consuming to reach this goal by implementing a dynamic and responsive website without the use of a framework, for this reason the development team chose to use Skel. It contains many needed features to achieve both robustness and portability.

"Skel is a lightweight, low-level framework for building responsive sites and web apps“ (AJ (n33), 2014). It provides a flexible CSS grid system, based on a box model, to facilitate responsiveness and compatibility for all modern desktop browsers and mobile platforms. The Skel framework normalises the browser environment when breakpoints are defined, the breakpoints set viewport conditions and apply different framework options for different screen sizes. These elements have been the foundation of the current responsive web design.
jQuery

“jQuery is a fast, small, and feature-rich JavaScript library” (jQuery Foundation et al., 2014). Its compatibility with all modern desktop browsers and ease of use made the development team to choose for this framework. jQuery was used for this project to apply manipulations to the DOM Documents as well as event handling for different purposes. The documentation of this framework was accessible and very extensive so the team was able to find and use the needed methods without to much trouble.

5.1.2. Implementation

The implementation of the system has been split up in three separate layers. In essence there exist only two layers; the Visualisation Layer does not contain many functionalities but will produce the Graphical User Interface. The Data Processing Layer only consist of PHP-code and processes the data that is stored at the server side. These two layers however, do not function well without the existence of a Connection Layer in between. The Connection Layer primarily connects the server-side with the client-side using JavaScript functions. The Connection Layer will also processes some parts of the data without using the defined functions in the Data Processing Layer.

A simplified visualisation of the different layers can be found in Figure 5.1. A more detailed description on the different layers is given in the following sections.

![Figure 5.1: Simplified flowchart for a typical route search.](image)

**Data Processing Layer**

The PHP scripts maintain the server-side of the application. Each call to the server is handled and answered with a XML-file containing the requested information. In order to process the request correctly the script decides between three different actions: search, save or load.

Saving a trail means that the trail identifier will be saved in the database along with a randomly generated sequence of characters, this sequence will be used in the future to load the saved trail. When the trail has been saved, it is ready to be loaded from the database afterwards.

While searching for trails, the script decides which trails lies within the search radius by checking whether the starting point of a trail is located within this search radius. When a trail satisfies this...
constraint, it will be loaded from the database. To retrieve a certain trail, all paths belonging that trail have to be loaded. The collection of these paths that represent a trail are saved in the database as an array of path_id's. When the information is obtained, the GPS coordinates can be fetched by loading the edges between the coordinates. Each retrieved data point will be processed consecutively into an XML-file, so there is no need to load the complete graph first and traverse it again to save everything to the XML-file. As soon as the XML-file is finished the Javascript callback function retrieves it.

Connection Layer
The Javascript functions are used to handle user interaction with the front end of the application. On the one hand there are functions that immediately respond to interactions, like placing a tree on the map or opening the advanced search section. On the other hand there are functions that are used to pass requests from the user to the server and parse the response from the server to visualise it on the Visualisation Layer.

Using an asynchronous call to the server enables the page to respond to user input while the server processes the request and computes a response in the form of a XML-file. As soon as the XML-file is finished, the callback function retrieves it and the XML will be processed. In order to be able to visualise the trails on the map, every GPS coordinate has to be stored in a polygonal line. Furthermore for every trail in the XML-file a corresponding information box below the map is added to the page. This box contains some extra information about the trail and enables the user to highlight the trail by clicking on the box.

Visualisation Layer
The Visualisation Layer represents the front end of the application. This layer is the interface for the users to communicate with the application and will visualise the trails within this design. There is one general style sheet that defines the layout of the application and three additional style sheets that alter specific parts of the web page for different screen sizes, it also contains a style sheet for the printing functionality which deactivates most visual objects and only displays the map in order for it to be printed.

5.2. Test Plan
Different types of testing methodologies and tools have been used for testing the application. Conventional application software testing is well defined with a set of methods and practised by the development team throughout the Bachelor of Computer Science. Many of these methods however, may not work with web based applications because of its dynamic nature and uncertainties. Therefore the team had to change the strategy they were familiar with, and needed to choose a set of different methodologies to ensure the quality of the application.

Continuous integration testing is applied during the implementation of the project as described in subsection 5.2.1. Several unit tests have been written for specific critical parts of the system to address misbehaviour after any committed change as discussed in subsection 5.2.2. Furthermore other quality tests have been carried out like performance tests, compatibility tests and validation to measure the quality of the service that needs to be delivered.

5.2.1. Continuous Integration Testing
As integration test, the team used an open-source project called Selenium for in-browser testing. This framework was chosen because the tool could create test cases for a wide range of applications, where similar open source tools like Canoo WebTest and HttpUnit were found to be insufficient as they could not handle most instances of recorded scripts.

Selenium automates browsers by performing a set of predefined tasks. After every committed code change, all the integration tests of Selenium will be executed to make sure that the commit did not break any of the application’s features nor the tests themselves. The Selenium test suite would produce an error for every unexpected form of behaviour of the application. When a test case produces an error, the tester can easily recognise the origin of the produced error and should be able to find certain faults or bugs faster, as can be seen in Figure 5.2. This allows the developers to refactor the front end code without fear of breaking the previously passing criteria of the application.
5.2.2. Unit Tests

Solid unit tests and well tested code will prevent future changes from breaking functionality. Ultimately, the unit tests will help to identify failures in the logic behind the system and could improve the quality of the code. During the development process the unit testsuite will be extended as new units will be added to test this logic in different segments.

Of course, writing high quality unit tests comes at the expense of investing time and a balance needs to be found between a sufficient high code-coverage against the investment of time. Since each language will require a separate library in order to test the code, the team has decided to unit test the most critical layer of the system that will not reveal potential errors easily by itself. Therefore the team will unit test the different segments of the Data Processing Layer which only contains PHP-code. The SimpleTest PHP unit tester-framework has been used to test this layer in detail.

A notable result from these tests is that the whole Data Processing Layer has been redesigned to make it more clear which parts were important to unit-test. Shorter methods are created and have been a great improvement for the code-quality. With this development the code became much more maintainable and will have an added value in the future when the system will be extended.

5.2.3. Performance Tests

Performance testing can be viewed as Black Box-testing, a test which focuses on the system with no knowledge of the program code. This test type reflects the behaviour of the complete website in terms of load time and connection methods. The main objectives of the performance test is to find any scalability issues, find the maximum of concurrent users that can be supported and finding possible bottlenecks.

The development team has used a free service, called tools.pingdom.com to test the performance of the website and retrieved very interesting diagrams, suggestions and the amount of produced errors from different perspectives as a result of the test.

The test performed on the 21st of January (Figure 5.3) did not show any surprising results. Pingdom.com indicated that the website performed 98% faster than all other tested websites with 111 requests and a load time of 394ms. The performance grade could be improved since it scored 83/100, based on 12 different elements. Seven elements received a grade between 90 and 100, out of 100. The lowest score from this test (27/100) was based on the numerous calls to the Google Maps API and could not be improved unfortunately.

One of the improvements that has been made as a result of these tests, was the order in which certain resources where included. The order of resource includes has been changed to improve the load time of the website. With this change, minor elements of the website are now loaded after the main components have been loaded successfully.
5.2. Test Plan

5.2.4. Compatibility Tests

This non-functional test type tests several components and is conducted on the application to evaluate its compatibility with various other objects. The application should be compatible with different browsers and screen sizes without affecting the functionality of the website. Different tools have been used to test the compatibility of the product. The behaviour of the website is tested with different screen sizes as well as different browser platforms. These tests have to show that the application is capable of being used on different devices without showing unexpected behaviour. Cross browser tests have been conducted by using the service of browsershots.org. This service enables the tester to select several browsers and it will upload screenshots to show how the website appears on these browsers. 117 different browsers have been tested and the majority displayed the website correctly. The browser that showed the website in such a way that it became unusable was Dillo 3.0.2. According to browser statistics less than 2% of the internet users uses another browser than Chrome, Internet Explorer, Firefox, Safari or Opera (W3Schools, 2015). Therefore the team has decided to ignore the misbehaviour because the amount of people that are affected by this misbehaviour is negligible.

Cross device tests have been conducted by using http://quirktools.com/screenfly/. Screenfly shows how the website would appear on different devices according to its screen size. Major changes have been made according to the test results since the website did not turn out to be usable on different devices. The developers observed that a usable design on a desktop-sized screen does not imply a usable design for mobile devices even after scaling its interface. Therefore the buttons have been repositioned for mobile devices to maintain an intuitive interface design even for different devices.
Discussion and Recommendation

For this Bachelor Project a responsive web application has been developed in 18 weeks, divided in two phases. The solution features a technique that is capable of visualising trails that have been collected by gathering GPS data. The proof of concept of this solution is designed as a user friendly and robust web application which allows users to find trails within a selected area. Additionally the user is able to select one of the trails and could potentially share the result by social media or download the trail for future use. These features help the visitors of the Horseshoe Meadow Ranch to navigate through the unimproved forestland without the need of guidance by other techniques.

6.1. Limitations

The final product is considered a prototype and contains several limitations that have not been addressed during the project because of limited time or knowledge. The database is currently filled with test data on which the system has been built. Although additional data has been added to the database to assure the system would not show any unexpected behaviour, the system still needs to be provided with more data. The collection of the data is very time-consuming and did not lie within the scope of this project. As a result of the scarcity of realistic data, proper naming of different trails as well as the additional points of interest are not added to the database. Although the feature of naming trails properly as well as displaying the points of interest have already been implemented, the naming of the trails are currently not meaningful and the points of interest are missing since the data does not provide this information yet. These limitations are acknowledged by the development team and have not been solved during project since the current data was satisfiable enough for the development of the prototype.

6.2. SIG evaluation

During the implementation phase the code was sent to the Software Improvement Group twice. The Software Improvement Group evaluates the code quality by reviewing the source code and assess the submission by taking several important code quality related elements into account. After every submission the Software Improvement Group graded the code measured in for example the Unit Size, Unit Complexity and Component Balance. The assessment has been beneficial for the project because the potential risks could be identified early, suggestions have been made to improve the maintainability and a higher code quality could be achieved as well.

The first submission \(^1\) was rated with four out of five stars, a score that is above average. The fifth star was not achieved because of a lower score on the Unit size, Unit complexity and Component balance. According to the detailed feedback that was received from the Software Improvement Group the team refactored the code in order to improve the Unit Size, Unit Complexity and Component Balance. Initially

\(^1\) The full SIG evaluation (in dutch) can be found in Appendix C.
searching and loading trails was primarily handled by one single method containing different functionalities. This was one of the main reasons that the Unit Size and Unit Complexity scored lower compared to the other components. The team has split this main method into several smaller methods which resulted in better maintainable code.

The code structure in general has changed as well. Libraries have been moved to one folder and the different modules have been divided over multiple files. Comments have been added in a general layout to indicate which module a certain method belongs to, along with a short description of its functionality. The system is divided in different components to keep the code maintainable when the system will be extended in the future.

The second submission was rated with the same grade although the Software Improvement Group did acknowledged a few improvements. The Unit Size and Unit Complexity improved significantly and the change in file structure made the project more descriptive according to the feedback of the second submission.

### 6.3. Process evaluation

The team has launched a successful prototype within 18 weeks while experiencing the process of developing a project from scratch on an international level. The process led the development team to this result and is considered as a great achievement because the members of the project team have learnt a lot on many different aspects. The process did not solely consist of successes but mistakes have been made as well. The development team underestimated the amount of work that was required to finish the project. During the implementation phase the development team planned the implementation of new feature almost every two days. The tight schedule and many deadlines (once a week) in combination with the size of the team created a dangerous pitfall which could lead to losing control of the process. To avoid this disaster the team repeatedly scheduled extra hours in to be able to reach the deadlines. The team experienced that the amount of work that could be done with two members is not exactly half of what four members would be able to achieve.

The members also experienced that a team could spend too much time on research during the orientation phase. When exhaustive research has been conducted the team had difficulties in making decisions that should not take as much time as it sometimes did. Fortunately the orientation phase was planned to be carried out over 7 weeks and it was therefore not a huge problem that it took more time to make these decisions. For future implementation projects however, the members will advise to spent less time on research since more research does not necessarily improves the quality of the decision equally.

The team enjoyed working on this project a lot. The experience of executing this project for a client from the United States of America was very exciting. The team considered it as the ultimate test to execute the project in a foreign country, without the familiar environment the team was used to work in. In an unknown environment the members were able to the apply the skills they learnt during the Bachelor of Computer Science.

The team experienced cultural differences in communicating as well as in the workplace. These additional experiences of the Bachelor Project have been of great value that made the members of the team gain in confidence on international level and on executing similar projects in an unknown environment.

### 6.4. Future work

Like every other project this project had its limitations. Although all the “must-have”s and “should-have”s from the MoSCoW model have been implemented, and future work can always be carried out to implement one of the extra functionalities represented in the MoSCoW model as “could-have”s. Some very promising features have not been implemented as they fell outside the scope of the project. These features however are not impossible and the development team recommends future developers to look into these features.
6.4. Future work

6.4.1. Administrator Panel
Adding new GPS-coordinates and paths to the application currently needs to be done by using a simple Java program and importing the generate file directly into the database. Both the Java program and the way the file has to be imported is not user friendly add all and should be changed. Making an administrator panel to take care of adding new data would help a lot in expanding the current data set. Another very useful addition would be the possibility to edit current trails. Sometimes the data shows strange spikes in the measurements, these spikes could be removed to get a smoother and more accurate trail. This feature could also get more exciting when not only administrators are allowed to add or change trails but the users as well.

6.4.2. Trail selection
In the current prototype a static trail representation is used. In order to improve the trail suggestions a dynamic solution should be implemented. Initial tests with Ant Colony Optimisation showed very promising results and could certainly improve the quality of proposed trails. Not only will it enable the search for trails with a specific length it also opens up the search for trails with specific viewpoints or scenery.

6.4.3. Advanced search
The current available search options are limited to trails and viewpoints within a specified radius. In the future adding options to search for trails with a specific length or starting and end point would be a big asset to potential users according to the feedback that has been received from the users during the execution of the user tests. Combining the advanced search options with a more dynamic search algorithm would enable users to freely roam the mapped environment in the way they would like to according to the conducted user tests.

6.4.4. Testing
During the project the development team decided to only unit test the most crucial PHP functions. Due to the lack of experience in web testing frameworks the team had a difficult time to extend the test suite with other modules. To guarantee a correct functioning of the application and to improve the test coverage unit tests for the Javascript functions and other PHP functions should be written.
Project Analysis

Introduction
This document describes the initial design of the Horseshoe Meadows Ranch (HMR) Application. The document focuses on the data analysis, the User Interface design and especially the computation of routes based on the route selection criteria.

The system is designed with regards to a set of design principles. The team has agreed with the client that the following elements are important and will guide the design process:

• **Robustness**
  Systems behave correctly if they operate successful in their environment. A system is considered robust, if “small” violations of the environment in worst case result in “small” deviations from correct behaviour.

• **Portability**
  The portability of a system is the usability of the same software in different environments. The software should be able to run on different devices with different screen sizes. Therefore the application should be portable to multiple devices.

• **Maintainability**
  The application may be extended with extra features or will maintain a bigger scope of routes in the future. Designing for maintainability means the degree to which the design can be maintained or repaired efficiently and economically.

• **Usability**
  The success of this project also depends on the usability of the application. The usability of an application refers to the quality of a user’s experience when interacting with the application and depends on a combination of factors including: memorability, subjective satisfaction, intuitive design and ease of learning and use.

Throughout this document these design elements have been taken into account. The design elements form the basis of the system and interface.
Data Analysis
The application has to visualize routes which have to be digitalized. The team has to collect the GPS-data of the available routes and analyze the data to be able to develop the correct system for the application. This chapter describes the process of collecting the data, the possible representations of the data and the database (structure and capacity).

Collected Data
The team collected a few data samples using two different methods: a handheld GPS-system (Garmin, 2004) and the android application My Tracks (Google Inc., 2014b). Both applications use different files, and both will be analyzed.

My Tracks
The My Tracks application allows the user to download the saved data of a trip. Each trip is downloaded from the device as a Zipped Keyhole Markup Language (.kmz) file. The .kmz is the zipped representation of a Keyhole Markup Language (.kml) file which is XML based. A standard browser should be able to open the .kml file in order to show the collected data correctly, an example of the .kml file is represented in Figure A.1.

![Figure A.1: Piece of the representation of the collected GPS data (.kml file).](image)

The .kmz is a file format used to display geographic data. The file contains features such as placemark, descriptions, ground overlays, paths and polygons. The coordinates are represented in longitude \((-180 \leq \text{longitude} \leq 180)\), latitude \((-90 \leq \text{latitude} \leq 90)\) and optionally altitude (in meters above sea level) (Google Inc., 2014a).

Garmin
The team has also used a handheld GPS system to collect the GPS data while using the Garmin Communicator Plugin which supports transferring the collected GPS data. This GPS file is downloaded as a GPS Exchange Format (GPX) which is a XML data format for the interchange of GPS data. This file can contain very detailed data such as GPS signal strength. An example of the .gpx file is represented in Figure A.2.

![Figure A.2](image)

As the .gpx usually contains more data than the .kml files, it should be no problem to convert .gpx files into the more lightweight .kml format.

Route Visualization
Route visualization in a web application is done by a web map application. There are several Application Programming Interfaces (APIs) available, the most popular ones are Google Maps, Bing Maps and OpenLayers (Steiniger and Hunter, 2013). These applications provide a mapping API that uses javascript in order to display 2D data in a map. A comparison has been made between Google Maps API and OpenLayers API to determine which API fits best in this project.

The .kml vectordata is easily overlayed using the Google Maps application, because of the integrated interface of Google Maps. When using OpenLayers, there exist no such infrastructure that integrates
the .kml vectordata (Batty et al., 2010). A workaround is possible to integrate .kml files with OpenLayers but it does not work as smoothly as it does with Google Maps.

It is important to note that the Google Maps API is not an open source API. This means in this case that the API license gives the API Provider the right to use the user provided data in any way they see fit (see Google API Terms Google Inc., 2013, sec. 11). On the other hand the OpenLayers API is free for non-commercial use under a Creative Commons license.

OpenLayers has a wide range of functions, but due to its size it is not ideal for mobile applications (Robin Lovelace, 2014). The Google Maps API lacks the flexibility of open source but does not consume as much space as OpenLayers does. When taking these observations into account, the development team has decided to use the Google Maps API for the application. The Google Maps API works smoothly with the collected .kml data and is suitable for mobile applications. The downside of using this API is less important compared to the benefits.

Database

The data is collected by recording the GPS coördinations from the trail as a GPS track. Each GPS track is a sequence of locations represented in latitude (ltd) and longitude (lon). A GPS track is a polygonal line with a single start and end point. The most intuitive visualization of the trails is a graph with vertices and edges where each vertex represents a trail junction and each edge represents a trail segment (which is a collection of GPS points). The development team will investigate what database model fits best and will design an initial database model.

Database Model

Since the initial design of the database describes a graph, it may be straight forward to implement the system in a graph database. A more common model is the relational database. Both models have been analysed to determine which model is best to use for this project.

Graph Database Model

A graph database model is a data and/or schema that is represented by graphs or by data-structures generalizing the notion of graphs (Graves et al., 1995).

The graph database model has several advantages, one of them is the natural modeling. The visible structure shows nodes and edges, where the nodes are entities with information and related information about the entities are represented in arcs (directed connections).

Another advantage is the speed of query computing. It appears that the graph database performs much better at large scale databases, compared to a MySQL relational database model (Vicknair et al., 2010).

Relational Database

The relational database management system has become a primary data storage structure. It can store data in different ways where the elements are stored in relational tables (Codd, 1970). It is used for simple record-type data which is handled by the unified language SQL. The advantage of the relational database model is that its size is in general about half the size of the corresponding graph database (Vicknair et al., 2010). Another advantage is that it is used worldwide for both commercial
and academic applications. Therefore most database hosts provide relational database management systems such as MySQL.

Conclusion
The relational database model focus on the data and its attributes whereas the graph database model represents the data and their relations in a graph. Most important difference between these models is that the graph model is able to determine relationships among data such as paths where the relational model has the ability to store graph data but not with as many extra features.

The speed of executing retrieval queries of graph databases is only optimized for full-text indexing and is less efficient of retrieval of other types of data.

The support for relational database models is very extensive, the data storage is more efficient and the graph database will perform only slightly better during the execution of non-full-text retrieval queries. Therefore the development team has decided to use a relational database because it applies better to the design principles of this project.

Initial Design
There are many different ways to store the data in the database, even when decided that the data will be stored in a relational database. This database will basically contain the junctions of the trails, represented as vertices and a collection of \((lat, lon)\) pairs that represents the trail segments between the junctions. The initial design contains 4 tables, where one contains all the points of interests with GPSid, description and the type of point of interest (such as junction or viewpoint). The POIid refers to the POI_translation table that translates the id to the type of POI. The other two tables will be filled with GPS coordinates (in the GPS table) and edges from junction to junction.

The relational database is also displayed in figure Figure A.3.

![Figure A.3: Referential integrity constraints displayed on the relational database schema](image-url)
User Interface Analysis

Websites are self-service products, where the user is on his own while using a website because it does not have a manual to read or direct client support. Therefore the success of a Website depends on its usability (Flavian et al., 2006).

According to the requirements given in the Product Vision, the product has several requirements. Besides the requirements it is also important to maintain an user interface that is designed well. The design is done in close collaboration with the client. The presented user interface (Figure A.5) is based on the wishes of the client and some psychological aspects regarding its usability. Taking these elements into account, the first design is presented in this chapter.

Requirements

The client wishes that the application can find routes according to the route preference criteria such as starting and end point or length of route. The application should be designed so that it can be used on devices with different screen sizes. Furthermore, to provide additional context to facilitate a user’s route choice, the suggested route could show the total length of the route, the total elevation gain and possible viewpoints. The application may also provide to output computed routes as printer-friendly files, as well as coordinates for upload to a GPS device and some way to show the selected route by using a unique url that can be attached to the invite sent via social media.

Usability

Website Usability “is a quality attribute that assesses how easy the user interfaces are to use” (Nielsen and Norman, 2000).

The usability of a website includes consistency of information and the ease of getting the website to do what the user intends it to do. A route planner should have the ability to choose between different route-selection criteria. However, a user interface with lots of different selection criteria may overwhelm human cognitive capacities in information processing (Miller, 1956b).

![Figure A.4: Grouping elements according to the relative distances between objects.](image)

The usability of the user interface will increase if a human being is able to find a structure in the interface. According to the principle of Proximity (Williams, 2008), which explains that the relative distance between objects in a display affects the perception of whether and how the objects are organized into subgroups. In Figure A.4, the human brain groups elements together according to the distances between the stars. To group elements on a web interface, the designer should work on the relative distances between elements.

When users navigate trough a website, typically they will not read every word on the web page. Users will scan quickly for relevant information. The designer has to structure the website well enough to give the reader the information he is looking for because poor information design can disrupt reading (Johnsen, 2010). Therefore the webdesigner needs to choose an easy to read typeface and a font that is big enough in size.

Thus the design of this application should be simple, grouped and clear.

First Design

The center of the design should be the map representation. The required buttons for the application are grouped together, for example: the three buttons to export the route (to PDF, GPS system or social media invitation) are grouped.

The system will be designed to find routes by selecting locations or by searching for routes in a specific area. The locations can be selected via interactive point-and-click identification on the map.
options to find a route are located below the map, and an extra menu will appear when the user hits the *Advanced Search* button. This extra menu will give the user the possibility to select additional information for finding a route such as length of the route, total elevation and possible viewpoints. A rough representation of the user interface is given in Figure A.5.

![Figure A.5: First design of main user interface](image-url)
Computation Analysis

This chapter describes the research for the computation of the application. After the data collection, the application needs to be able to display several routes according to the users preconditions within a reasonable amount of time. This requires the system to compute the routes efficiently. There are several studies available that explore the possibilities within the computation of shortest path algorithms.

This system is not necessarily looking for the shortest path, the application needs to find a route between origin and destination, but the route also needs to have approximately the same length as the user has given as input.

Route Representation from Google Maps

The system is designed to find optimised routes between origins and destinations. The locations can be selected via interactive point-and-click identification on the Google Map. The longitude and latitude coordinates will be derived from Google Map’s geocoding service. A Structured Query Language (SQL) will be used to find the nodes nearest to the user’s origin and destination in the route network database.

Route Selection Algorithm

To plan a route from A (start) to B (destination) the most common practice is to find the shortest possible path using a shortest path algorithm like Dijkstra’s algorithm (Dijkstra, 1959), Bellman–Ford algorithm (Ford, 1956) (Bellman, 1958) and Floyd–Warshall algorithm (Floyd, 1962). Unfortunately shortest path algorithms are not applicable to this project because most users are not interested in the shortest path. This means that the shortest path algorithms will most likely propose a trip that does not satisfy all the users requirements. Another problem with shortest path algorithms is that they only give one optimal solution whereas the application preferably should give multiple suggestions of routes, enabling the user to explore different parts of the forest land.

The k-shortest path algorithm (Eppstein, 1998) is an extension of the shortest path algorithm. The k-shortest path algorithm finds the shortest path, as well as the next \( k - 1 \) shortest paths. The algorithm basically iterates the general shortest-path algorithm \( k \) times and returns k-shortest paths with or without loops (loopless k-shortest path).

A better suited solution would be a Hamiltonian path through (a part of) the graph, in case of a cycle a solution would be the well known Travelling Salesperson Problem. The only downside with computing a Hamiltonian path is that it is an NP-complete problem as proved by Karp (1972). In most cases it is therefore impossible to compute a solution within polynomial time. An approximation algorithm can be used to approximate the solution to this problem.

Ant Colony Optimisation (ACO) as first proposed by Dorigo and Gambardella (1997) is a possibility that utilises a simulation of ants traversing the graph. Every iteration a predefined number of virtual ants explore the graph to find the shortest cycle. Every next node is picked based on a combination of the distance to that node and the amount of pheromone on the edge to that node. After each ant completed it’s cycle an amount of pheromone is deposited on each cycle inversely proportional to the length of the cycle the ant found. This way the shortest cycle gets the most pheromones, which makes it more likely that ants pick that cycle.

Instead of using the regular ACO solution to find the shortest cycle it might be possible to change the amount of deposited pheromones based on a fixed length \( k \) making a cycle of length \( k \) more desirable and thus resulting in the ants approximating a cycle of length \( k \).
Conclusion
The exact solution for this problem is most likely within the NP problem domain. This means that the solution is not computable within polynomial time and therefore not suited for this project. The studied approximation algorithms are not as accurate as the development team would like them to be, and some algorithms may also perform better on the basis of several important factors such as database design, computation limits etc. Since one of the design goals states that the product should be robust, the development team has decided to first implement a system with a simple and easy route visualization. If the system works and is stable, the development team will try to implement a suited algorithm for this specific problem. Because the team implements the system in this order, it is possible to find a better suited algorithm as more restrictions might become clear to the team and product owner making the problem possibly easier to solve.
Product Vision

Introduction
The Product Vision describes the desired state of a product in the future. The principle is to create a statement that describes the target customers, the customers needs and how these needs will be met. The vision details further planning (design) activities and may divert from the vision because the future might bring a changed perspective on the product and its requirements. The product vision also contains critical information of development to launch a good product. According to Ken Schwaber the product vision is: The vision that describes why the project is being undertaken and what the desired end state is. (Schwaber, 2004, p68)

The Company
The Horseshoe Meadow Ranch (HMR) is located in Sisters, Oregon. The business has evolved to become a hospitality unit that caters to horse owners who are looking for lodging and recreation at the same site as the stable that serves their horses. Central Oregon is a destination location with 4 season attractions, wild scenic and majestic mountains and untamed rivers. The facility has four separate lodging opportunities with room for 4 to 16 human guests and roughly twice that for equines.

Problem Description
The HMR has access to more than 1618 hectares of open unimproved forestland. Guests and the general public have access to the land via several defined entrance points. The company spends a considerable portion of time explaining to the guests what to see and where to go as there are few guideposts in the available area. Additionally if the guest becomes disoriented, it can become problematic to identify the area in which they are. Therefore the team has been asked to develop a methodology whereby the trails and features can be mapped. Once that data has been secured, the team needs to develop a delivery system that will enable the user to use readily available assets to be self-reliant with regard to opportunities (e.g. routes and times).
The Horseshoe Meadows Ranch application will be an interactive method to plan trips that utilizes national forestland appropriately and is preferably integrated with social media. The initial product will contain the available routes of the Indian Ford area only and can be expanded once the system has been validated. The saved routes should contain the distances, the estimated time of travel and a map. The system should advise trips based on either a starting point along with an estimated time of travel or based on a starting and end point.

This chapter describes the final product in three different ways:

- The product vision describes what users expect and need from the product on a very abstract level. It does not describe specific features or solutions to problems, but it describes the requests from the users.
- The high-level product backlog splits the project in a short, prioritized list of tasks. When all these tasks are completed, the product is finished.
- The roadmap gives expected release dates for items on the high-level product backlog. It is the planning for the product.

The vision should act the overarching goal and guiding everyone involved in the development process. On the basis of the following sections Figure B.1 has been put together to summarize the product vision.

**Vision Statement**
Search for beautiful routes to organize trail rides, select one of the suggested routes that fit your preferences and invite friends to join your trip.

**Target Group**
The target users of this product are currently looking for a method which helps the user to guide through the unimproved forest lands. The user may be interested in finding new routes, or just needs a guiding method because the user has no experience in the area at all. The user might as well want to send invites to others with the trail information so others are able to join in.

**Needs**
A robust and effective tool that contains all the routes of the selected area. A search query will show results of preferred routes. It is also possible to invite friends when the trip has been planned. The routes are printable and available on several devices such as a personal computer or a smart phone.

**Product**
The product has three key features:

- The product visualizes all available hiking and trail ride routes within the selected area.
- The product will be web-based so it can be used by different devices.
- It connects neighbours who share the interest for horseback riding by sending invitations through the application.

**Value**
The unimproved national forests will be utilized appropriately. The method will also be used to proof the use of these forest lands. As long as the method can proof the recreational use of the forests, politics will refrain from changing the recreational purpose.
Personas
In accordance with the product vision, the team has described a few personas. A persona is a fictional character that represents a subset of the market the team wants to address. The information is based on direct observation and market research. Personas will encourage the development team to embrace a user-centered approach. The personas are given in Appendix A: Personas.

User stories
The user stories are used as an agile technique to capture product functionality and as a communication and collaboration tool. Therefore each story needs to be feasible and testable, in order to achieve this the stories are based on personas. Constructing user stories also helps in the creation of acceptance tests and are ready for implementation if everyone on the team has a shared understanding of the meaning of the stories. The user stories are presented in Appendix B: Userstories and use the following format: “As a <user role> I want to <goal> so that <benefit>.”
Product Roadmap
To realize the vision, a product roadmap has been written. The roadmap enables the customer to demand more frequent changes because it provides a high level view of the targeted releases. The development team is forced into thinking in steps of these targeted releases which helps to keep the detail of the product more appropriate. The product roadmap is comparable with a journey that has been planned upfront and shared with fellow travelers, where the journey is the development process and the fellow travelers are the delivery people.

This chapter contains the product backlog and determines the features of the product, the roadmap with the dates, contents and objectives of the foreseen releases.

Product Backlog
The product backlog determines a list of desired features which needs to be built. In anticipation of the release planning, the product backlog contains a prioritized list of all items so that the development team can provide estimates for the realization of each feature. These features are based on the MoSCoW model (Table B.1) and user stories.

MoSCoW model
This section contains the features of the system described using the MoSCoW model (Hatton, 2008). The MoSCoW model divides all features in four different groups:

- **Must Have**: Core functionality of the system, without these features the system will not work.
- **Should Have**: Features the system needs to function properly, but could do without.
- **Could Have**: Features that will probably not make it in the system, but that would be great for a follow up project.
- **Won’t Have**: Features that will not be implemented in the system, even if there is enough time to do so.

The MoSCoW model is given in Table B.1:

<table>
<thead>
<tr>
<th>MoSCoW</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Must Have</strong></td>
<td>Search for route</td>
</tr>
<tr>
<td></td>
<td>Display routes in map</td>
</tr>
<tr>
<td></td>
<td>Normal &amp; Mobile layout</td>
</tr>
<tr>
<td><strong>Should Have</strong></td>
<td>Save routes (unique identifier)</td>
</tr>
<tr>
<td></td>
<td>Sent invitation through Social Media</td>
</tr>
<tr>
<td></td>
<td>Make printout of selected route</td>
</tr>
<tr>
<td><strong>Could Have</strong></td>
<td>Display route on smartphone</td>
</tr>
<tr>
<td></td>
<td>System to add new routes/datapoints</td>
</tr>
<tr>
<td></td>
<td>Show extra features of routes (viewpoints)</td>
</tr>
<tr>
<td><strong>Won’t Have</strong></td>
<td>Return current location</td>
</tr>
<tr>
<td></td>
<td>Login system</td>
</tr>
<tr>
<td></td>
<td>Collection of personal data</td>
</tr>
<tr>
<td></td>
<td>Fees or collection of money</td>
</tr>
</tbody>
</table>

Table B.1: MoSCoW model

Backlog
The product backlog is a table with product requirements. It briefly describes the features derived from the user stories (Figure B) and the MoSCoW model (Table B.1), so that the development team can provide estimates for the realization of each prioritized feature.

The prioritizing method used for the product backlog (Table B.2) is a simple ranking method. The requirements are ranked according to Fibonacci’s numbers $1, 2, 3, 5, \ldots, ((n - 1) + (n - 2))$, where the most important requirement is ranked one and the least $((n - 1) + (n - 2))$. 
## Feature Description Priority

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Route Database</strong></td>
<td>Create a database with several routes</td>
<td>1</td>
</tr>
<tr>
<td><strong>Search Interface</strong></td>
<td>Create web-interface to search for different routes</td>
<td>2</td>
</tr>
<tr>
<td><strong>Map Interface</strong></td>
<td>Display routes into map for navigation</td>
<td>2</td>
</tr>
<tr>
<td><strong>Mobile Interface</strong></td>
<td>Make the current interface mobile compatible for several devices</td>
<td>3</td>
</tr>
<tr>
<td><strong>Unique Identifier</strong></td>
<td>Identify routes with a unique URL</td>
<td>5</td>
</tr>
<tr>
<td><strong>Invite Interface</strong></td>
<td>Create web-interface which enables the user to invite others</td>
<td>5</td>
</tr>
<tr>
<td><strong>Print Route</strong></td>
<td>Create a method that converts selected route into PDF for printout</td>
<td>8</td>
</tr>
<tr>
<td><strong>Select Speed</strong></td>
<td>Select speed to calculate the estimated travel time more accurately</td>
<td>8</td>
</tr>
<tr>
<td><strong>Viewpoints</strong></td>
<td>Show the special viewpoints per trip</td>
<td>13</td>
</tr>
<tr>
<td><strong>Export GPS Data</strong></td>
<td>Export GPS data to a handheld GPS device.</td>
<td>13</td>
</tr>
<tr>
<td><strong>GPS integration</strong></td>
<td>Enable location tracking during navigation</td>
<td>21</td>
</tr>
<tr>
<td><strong>Add routes</strong></td>
<td>Create user-friendly interface to add routes</td>
<td>21</td>
</tr>
</tbody>
</table>

### Table B.2: Product Backlog

#### Roadmap

The project has been split up in two phases: The Research Phase and The Implementation Phase. The research phase will span 7 weeks, starting with week 28 and ending with week 35. The implementation phase will span 10 weeks, starting with week 46 and ending with week 5. The roadmap (Table B.3) is based on the Scrum principle and focuses on having a shippable product after every sprint.

<table>
<thead>
<tr>
<th>Week</th>
<th>Day</th>
<th>Name</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Friday</td>
<td>Product Vision (draft)</td>
<td>Collect client needs</td>
</tr>
<tr>
<td>1.2</td>
<td>Wednesday</td>
<td>Data Collection</td>
<td>Collect and analyze GPS data</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td><strong>Product Vision</strong></td>
<td>...</td>
</tr>
<tr>
<td>1.3</td>
<td>Friday</td>
<td>System Analysis</td>
<td>Find proper Data Representation</td>
</tr>
<tr>
<td>1.4</td>
<td>Wednesday</td>
<td>Data Processing</td>
<td>Process the collected data to fill database</td>
</tr>
<tr>
<td>1.4</td>
<td>Friday</td>
<td>Project Analysis (draft)</td>
<td>Describe the system and first design</td>
</tr>
<tr>
<td>1.5</td>
<td>Wednesday</td>
<td>Construct User Test</td>
<td>Construct Visual Design</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td><strong>Project Analysis</strong></td>
<td>...</td>
</tr>
<tr>
<td>1.6</td>
<td>Wednesday</td>
<td>Conduct User Test</td>
<td>...</td>
</tr>
<tr>
<td>2.1</td>
<td>Wednesday</td>
<td>Initial Database</td>
<td>Set up the initial database</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td>Algorithm Analysis</td>
<td>Analyze available algorithm</td>
</tr>
<tr>
<td>2.2</td>
<td>Wednesday</td>
<td>General Interface</td>
<td>Implement the general interface / lay out</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td>Map Module</td>
<td>Implement the map Module</td>
</tr>
<tr>
<td>2.3</td>
<td>Wednesday</td>
<td>Search Module</td>
<td>Implement the search module to find routes</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td><strong>Release</strong></td>
<td>First official release of project</td>
</tr>
<tr>
<td>2.4</td>
<td>Wednesday</td>
<td>Update Database</td>
<td>Fill database with extra data and refine its design</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td>Unique Identifier</td>
<td>Implement the functionality with interface</td>
</tr>
<tr>
<td>2.5</td>
<td>Wednesday</td>
<td>Mobile Interface</td>
<td>Make the interface compatible for several devices</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td>Create Printable Version</td>
<td>Export a selected route to a printable version</td>
</tr>
<tr>
<td>2.6</td>
<td>Friday</td>
<td>Invite Interface</td>
<td>Enable a user to invite others through social media</td>
</tr>
<tr>
<td>2.7</td>
<td>Wednesday</td>
<td>Select Speed</td>
<td>Select speed for more accurate planning in trip</td>
</tr>
<tr>
<td></td>
<td>Friday</td>
<td><strong>Release</strong></td>
<td>Second official release</td>
</tr>
<tr>
<td>2.8</td>
<td>Wednesday</td>
<td>Final Report (draft)</td>
<td>Sent draft of final report</td>
</tr>
<tr>
<td>2.9</td>
<td>Wednesday</td>
<td>Final Report</td>
<td>...</td>
</tr>
<tr>
<td>2.10</td>
<td>Friday</td>
<td>Presentation</td>
<td>...</td>
</tr>
</tbody>
</table>

### Table B.3: Roadmap
Definition of Done
At its most basic level, a definition of Done creates a shared understanding of what it means to be finished, so everybody in the project means the same thing when they say “it's done”. (Stevens, 2012)
It is important for the development team to have a shared understanding of what it means for work to be complete, to ensure transparency. The Definition of Done Schwaber and Sutherland (2013) is used to assess when work is complete after the implementation of a feature, at the end of the sprint or release.
Each category maintains its own Definition of Done, and might change over time. The Definition of Done guides the development team in knowing how many Product Backlog items it can select during a Sprint Planning.

Feature
Each feature is finished if it meets the requirements of the following list:
1. The feature has been implemented according to its description.
2. The source code is properly commented
3. The source code committed on GIT and FTP
4. The code review completed (or pair-programmed)
5. Relevant documentation produced
6. Unit tests are written and passed
7. Regression tests are written and passed
8. Acceptance test passed

Sprint
Each sprint is finished if it meets the requirements of the following list:
1. All the features of the sprint backlog have been finished
2. Relevant documentation produced/updated
3. Summary of changes updated to include newly implemented features
4. Integration tests are written and passed
5. Usability tests have passed
6. Compatibility tests are written and passed
7. At the end of the sprint, a demonstration has been given to the client

Release
Each release is finished if it meets the requirements of the following list:
1. The features with the highest priority have been implemented during the sprints.
2. Relevant documentation produced/updated
3. Summary of changes updated to include newly implemented features
4. Performance tests have been performed
5. Usability tests have passed
6. Acceptance tests passed
7. Approval from Product Owner
Appendix A: Personas

Persona 1
Name: Derrick Patil
Birthday: March 30, 1981
User type: Regular user
Derrick is an experienced horseback rider and plans new events regularly. He rides his horse always in the same forests, and likes to invite friends to join him during the trips.

Persona 2
Name: Agnese Newman
Birthday: January 21, 1982
User type: Regular user
Agnese owns several horses of different ages, depending on the horse she will ride a route with a different length or difficulty. She rides daily and is always looking for new routes to ride.

Persona 3
Name: Margery Ottosen
Birthday: July 10, 1976
User type: Regular user
Margery is a great indoor horse rider but she likes to ride through the woods as well. With less experience outdoors she likes to join others for safety reasons.

Persona 4
Name: Laurence Brant
Birthday: November 7, 1988
User type: Regular user
Laurence just moved to Sisters with his horse. He likes easy trails and prefers short routes to take it easy. He’s looking for people to join him during trail rides to meet new neighbors.

Persona 5
Name: Raphael Dykstra
Birthday: April 24, 1994
User type: Regular user
Raphael is an adventurous horseback rider. He likes to try out new trails and he uses his phone to guide himself through nature. He usually rides for 1 hour or more.

Persona 6
Name: Anika Alma
Birthday: April 2, 1975
User type: Regular user
Anika is a fine horseback rider and travels throughout Oregon to find new trails in her favourite area. She does not want to depend on electric devices during her trail ride.
Appendix B: User stories

- Derrick:
  1. As a user I want to plan routes so that I can plan events.
  2. As a user I want to invite friends so that we can ride together.

- Agnese
  3. As a user I want to view the available routes so that I can ride the horse that suits the trail.
  4. As a user I want to know the length of the trail ride so that I can ride the horse that suits the trail.
  5. As a user I want to know the viewpoints of the routes so that I can select a route I like.
  6. As a user I want to select different routes so that I do not have to ride the same trail every time.

- Margery
  7. As a user I want to know the route so that I can confidently ride outdoors.
  8. As a user I want to join other riders so that I do not have to ride alone.
  9. As a user I want to select the trail so that I know the route.

- Laurence
  10. As a user I want to select trails based on their difficulty so that I can ride easy trails.
  11. As a user I want to select the length of the trail so that I can plan short trails.
  12. As a user I want to invite neighbors so that I can meet the people in my neighborhood.

- Raphael
  13. As a user I want to select adventurous trails so that I can enjoy nature.
  14. As a user I want to select new trails so that I can try new routes.
  15. As a user I want to have the route on my mobile device so that I can guide myself through nature.

- Anika
  16. As a user I want to view the selected trail ride so that I know how to ride the route.
  17. As a user I want to find new trails so that I can ride in my favourite area.
  18. As a user I want to print the route so that I do not depend on electrical devices.
**First SIG evaluation**

De code van het systeem scoort net 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Unit Size, Unit Complexity en Component Balance.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Binnen de langere units in dit systeem, zoals bijvoorbeeld de ‘generatePoly’ unit in ‘polyxml.php’ en de ‘searchLocationsNear’ unit in ‘maps.js’, zijn aparte stukken functionaliteit te vinden welke ge-refactored kunnen worden naar aparte units.

Commentaarregels zoals bijvoorbeeld ‘// Compute length of trail’ en ‘// Get all GPS coordinates for the route’ zijn een goede indicatie dat er een autonoom stuk functionaliteit te ontdekken is. Ook de aparte functionaliteiten voor het zoeken naar ‘trails’ of een ‘poi’ kunnen uit elkaar getrokken worden. Het is aan te raden kritisch te kijken naar de langere methodes binnen dit systeem en deze waar mogelijk op te splitsen.

Voor Unit Complexity wordt er gekeken naar het percentage code dat bovengemiddeld complex is. Ook hier geldt dat het opsplitsen van dit soort methodes in kleinere stukken ervoor zorgt dat elk onderdeel makkelijker te begrijpen, makkelijker te testen en daardoor eenvoudiger te onderhouden wordt. In dit geval komen de meest complexe methoden ook naar voren als de langste methoden, waardoor het oplossen van het eerste probleem ook dit probleem zal verhelpen.

Wat betreft de Component Balance valt het op dat de componenten-structuur op het file-systeem draait om verschillende programmeertalen. Dit maakt het voor een ontwikkelaar in eerste instantie lastiger om een algemeen beeld te krijgen van de functionaliteit die het systeem aanbiedt. Zeker als het systeem gaat groeien is het aan te raden om de code in verschillende (functionele) componenten op te delen om zo een eerste indruk te geven van de high-level structuur van het systeem.

Daarnaast valt het op dat niet alle gerefereerde code is aangeleverd, zo zijn de bestanden ‘jquery.min.js’, ‘jquery.scroll.min.js’ en ‘skel.min.js’ niet meegeleverd. Uit de ‘index.html’ valt op te maken dat deze library code naast de zelf geschreven code staat. Om ervoor te zorgen dat het duidelijk voor nieuwe ontwikkelaars is dat deze code niet aangepast dient te worden, raden wij aan om dit type code in een aparte directory te zetten die aangeeft dat dit externe code bevat.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkel fase. Als laatste nog de opmerking dat er geen (unit)test-code is gevonden in de code-upload. Het is sterk aan te raden om in ieder geval voor de belangrijkste delen van de functionaliteit automatiche tests gedefinieerd te hebben om ervoor te zorgen dat eventuele aanpassingen niet voor ongewenst gedrag zorgen.
Second SIG evaluation

In de tweede upload zien we dat zowel de omvang van het systeem als de score voor onderhoudbaarheid licht is gestegen. Op het gebied van Unit Size en Unit Complexity zien we een duidelijke vooruitgang, de units in de tweede aanlevering zijn een stuk korter en simpeler. Het specifieke commentaar ‘// Get all GPS coordinates for the route’ is echter nog wel aanwezig. Wat betreft de Component Balance zien we nu een component ‘website’, wat al iets aangeeft van de functionaliteit van het project. Ook dit keer lijkt echter niet alle code aangeleverd te zijn, in de HTML is nu echter wel een ‘lib’ directory gemaakt te zijn. Als laatste is het goed om te zien dat de aanlevering nu geautomatiseerde (unit-)testcode bevat.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject. Het is goed om te zien dat naast een verbetering in de onderhoudbaarheid er ook een stijging in het volume van de test-code te zien is.


J. Loeliger and M. McCullough, Version Control with Git (O'Reilly, 2012).


Creative Commons, Creative Commons, (2015).

M. Graves, E. Bergeman, and C. Lawrence, Graph database systems, Engineering in Medicine and Biology 14, 737 (1995).


G. Miller, *The magical number seven, plus minus two: some limits on our capacity for processing information*, Psychological Review 63, 81 (1956a).


AJ (n33), *Skel*, (2014).


G. Miller, *The magical number seven, plus minus two: some limits on our capacity for processing information*, Psychological Review 63, 81 (1956b).


