CS958 Project

Coursework Assignment

Original Screenplay By

Lewis W. Britton, L.W.B.

202194412

University of Strathclyde

Glasgow City, Scotland

Burning Roots

Written & Directed By

John Hughes

Executive Producer

Michael Mann

Created By

Anthony Yerkovich

Music Composed & Performed By

Jan Hammer

... Words

Dissertation Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science Software Development at the University of Strathclyde

Academic Year 2021/2022

Abstract

Index terms:

Declaration & Information

This dissertation is submitted in partial fulfilment of the requirements for the degree of Master of Science Software Development at the University of Strathclyde. It accords with the University's regulations for the programme as detailed in the University Calendar.

Mail To: wi.lbritton@yahoo.com	Telephone: 07415 212 ***
Website: lewisbritton.com	GitHub: FedeRog1977

This document's presentation reflects the use of LaTeX typesetting (Figure B7), using Computer Modern Unicode (Figure B8) (I haven't reached my GNU Troff phase yet). This escapes the inane formatting requirements of my institution. References are presented using BibTeX, favouring *oblique* over *italic*, in-line with Donald E. Knuth's preference (Knuth, 2020). The process is executed in command line using Vim, which is a very powerful editor that has many commands, too many to explain in a tutor such as this. For maximum optical pleasure, the use of MμPDF is vigorously advised with [-I]. Navigate this document using h(←), j(↓), k(↑), l(→), ensuring that the Caps-Lock, Super-Key 'mod', or any other command key is not depressed. Note that the Oxford Serial Comma is favoured throughout this text. This study's sentence structure focuses on pragmatics and syntax, disregarding bloated filler content. Arguments are coherent, logical, definitive and straight-to-the-point. Nugatory theory is ignored. If you are curious about any of the mathematical, operational, logical, etc., symbols or notation used in this report, a comprehensive LaTeX-syntax-based symbolist will be available from my website library from approximately summer 2021.

The word count of this piece reflects relevant content from titles, heading classes 1, 2 and 3, paragraphs, footnotes, tables (excluding [results] tables 4.7, 5.10 → 5.20), table titles, figures, and figure titles in *Chapters 1 → 6*. Word count excludes any pre/succeeding content from *Abstract*, *Declaration & Information*, *Acknowledgements*, *Table of Contents*, *Appendices*, and *Bibliography*.

I declare that this document embodies the results of my own work and that it has been composed by myself. Following normal academic conventions, I have made due acknowledgement of the work of others.

Signed:

Date:

Acknowledgements

I would like to thank my dissertation supervisor, Dr Devraj Basu, for his approach with regards belief that students must be self-disciplined, organised, structured and punctual to their own degree. This closely relates to my own personally practiced work ethic and philosophy of the ASAP standard.

I would like to give credit for the computational aspect of this study to one of my biggest inspirations, John "The Tzar" Kelly. He inspired my love for everything barebones computational, from simple arrays (of hope), through Hyperthreading-enabled, all the way to x86 Assembly. I would also like to accredit Luke Smith for the foundation of my knowledge of Bram Moolenaar's Vim and LaTeX. This study's presentation would not have optimal without Smith (2015).

This piece would not have been as efficient without the aid of the only acceptable Linux distribution, 'distro' if you will, Arch Linux. I would like to thank Judd Vinet for his eye-opening and life-altering contribution to the development and computer-system enthusiast community. 'The Arch Principal' is certainly out in high force. Finally, for making use of this software mechanically efficient, I would like to thank IBM for the creation of the ThinkPad T23, X30, T42, R50e, T60, X60, X200, X220 and T420 neoVimPads, the UltraDock, and the 1987 Model M *Catastrophically Buckling Compression Column Switch and Actuator* typehorse (US369 9296A, 1972). For your convenience, one of my blog posts can satisfy your interest in this.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose & Industry

The system developed throughout this project is a functional web application based on providing user-location-based and external GPS data. User-based GPS services are based on data relevant to the user's machine and external services are based on context-relevant data. The system, which so forth may be referred to as 'the system', 'the application', 'the site', provides detailed information, guidance and recommendations relevant to sport and leisure, particularly hiking and its associated practices, in Scotland's fine rural outdoors and [*these mist covered mountains, which are home now for me*]. Therefore, upon a hypothetical release of a full version of this system, it would be a direct competitor of services such as Walkhighlands, Strava, Garmin Connect, AllTrials, etc. Due to the autistic and comprehensive nature of its development, it would be in the market not in the competition-driven business, but in the [empire business].

1.2 System Structure

The site upon which the system is spread is static, not dynamic, meaning any possible requests made buy the user are based on existing data. The site does not reference an external database for any purpose. Thus, no PHP or SQL-relevant content. The system is segmented into four, approximately equivalent, parts with the metric(s) determining their weight being algorithmic volume, number of services, etc.

The first, 'home'/'drafting room', page allows the user to view a comprehensive overview of the site's offerings. It allows the user to quick-view activities in the 'overview' section; analyse their projected personal ability and their projected gear performance (upon their input(s)) in the 'conditioning' section under 'ability' and 'equipment cache'; and, view a coordinate-based weather briefing in the 'weather' section. All of these sections include 'key's and 'suggested reading's. Yes, this site is that obnoxious.

The second, 'conquest map', page allows the user to view and interact with various GPS and mapping features including their location and location seeking. It also includes functionality which allows pinpointing of particular features upon interaction, such as Munros, Munro Tops, Corbetts, Corbett Tops, etc. Further it includes aspects which allow the user to seek, be recommended, select and print GPX routes on the map.

The third segment is the 'ranger calculator' which is used purely for analytical purposes and allows the user to input data relevant to their ability, equipment, routes, etc. and will deliver output in the form of statistics tables and charts based on various computations. This element does not implement any GPS functionality in-line with the system's primary focus however, is extremely relevant.

The final segment is the 'general search' function which allows the user to input or select search criteria which returns a comprehensive overview of all statistics relevant to the match(es). Unlike the other sections, this is more subjective and informative, as opposed to being logical / statistics-oriented. That is, is exists more so for the users understanding of what they're doing and how to actually interpret some of the statistics they're being delivered in other elements. It is open to their use and interpretation.

The service is called '*Burning Roots*'. No, the rhetorical use of satiric misspelling is not unintentional malapropism; it is in fact deliberate. In harmony with the feeling you'd experience when [racing south-west to Lone Stallion Ranch], the term references that certain '*burning* desire' for freedom in the sweet country air, the one you only experience when digging to your deepest '*roots*' to achieve a new personal record or firmly assert your dominance over your inferiors. This service uses and computes data to encourage a user to take to the trails with motivation to be the fastest, most efficient, most prepared and most endured athlete on their *routes*.

1.3 Users & Platforms

All users of this system will be sport/leisure oriented and as this is focused on a specific group of enthusiasts who have a firm set of beliefs and a strong pre-developed relationship with their sport (lifestyle), it will likely only receive traffic from athletes who are already hiking-inclined. It may encourage new hikers due to the comprehensive nature and customization opportunity of the learning and planning material however, it is unlikely. The most efficient empires dominate only one type of market. This market does however expand to: walkers, hillwalking enthusiasts, scrambling-inclined 4x4s, [T6 van-life] climbers, and Scottish mountaineers / ice climbers.

Upon original briefing, this system was planned to include a road cycling section which would essentially mirror the hiking part with cycling-specific data. However, upon reflection this element is irrelevant for two reasons. The first being all functional areas are covered by the computational processes involved with hiking data. And second,

road cycling has little association with hiking and therefore the adjacent sport may be seen as irrelevant by a specific user. If it were to be included, it would only be logical to implement a wider array of sports for example, excluding road cycling and including mountain biking and (fell) running, which are actually relevant to hiking. Or an even wider array if road cycling were to blend in seamlessly. This is unnecessarily [time-consooming] and only duplicates processes and would not deliver additional benefit, only diminishing returns, upon the project marking process. I'm not [the Zuck'], not only do I not have the time or resources for this, I do not have the relevant background knowledge.

As far as platforms go, the user arrives at this site through a web browser. This system is deployed as a website and is therefore extremely versatile and usable on any device. Browser caching of script elements allows a user to view and interact with the relevant data offline, provided they receive GPS signal. For example, they can still view their location and route on the 'conquest map'. Of course, [this means that] the site is constructed using HTML, CSS and JavaScript. Elements of JavaScript allow this site to dynamically scale to various device sizes tailor relevant content to these devices.

1.4 Development Process

As this is a [solo project], it's one man, his [ThinkPad X220], [Artix Linux] and his [neoVim] setup. There is little requirement for extensive use of any formal [inane] project management methods such as team-based allocations or associated time-based or progress management coordination frameworks. Therefore, any adherence to processes aimed at mapping management of this project are / have been more logic-oriented and variable, allowing creative freedom. I work on an ASAP basis so one creative day of thinking may be followed by a [5am – 11pm] of implementation, which may then be followed by 2 days of idling. Any formal micro-level plan would be redundant.

Succeeding acquisition of user requirements, the most important part of gaining an understanding of how this system would look and operate is determining how the user interacts with the various aspects of their sport. That is, what data/inputs must the user provide, how will this be used, and what will it be used for to satisfy the requirements. Therefore, the mapping of the functional structure of the user interface (UI) leads this in the sense that it demonstrates the logic and process of interaction relevant to this data. Thus, this creates a valid starting point. And so forth, the development process of this project reflects what follows:

Requirement analysis
→ Structural design
→ Usable data construction
→ Graphical user interface functionality
→ Usable data implementation
→ Graphical user interface graphic design
→ Testing

Following the structural mapping, there is little sense in proceeding without any data to work with as incremental testing of site functionality would be challenging to impossible. So, it's at this point which the acquisition of relevant data takes place. In this case, this data accounts for non-user-centered data such as GPS coordinates, map regions, landmasses and their attributes, etc., which are essential for the majority of computations. Therefore, not only is [JavaBloat] implemented to manage site dynamics, it also computes based on data from these discussed files, in JSON format. It is only after this when the graphic design of the site can be allocated more focus, however of course much of it comes instinctively along the way also. After this, and frequent incremental developer tests, the system is ready for more expansive developer and user testing.

For the natural ease in workflow, for the developer's mental state, and for the minimization of [nugatory] methodologies and [bloated] task flow [cargo donkeys] such as IDEs and [froymeworks], all files (including 'code', data files, notes and write-up) are [composed and performed by Lewis Britton] in [Bram Moolenaar's Neo Vi Improved], in the command line of a pragmatic dwm setup on Artix. HTML, CSS and JavaScript is written from scratch in plain text format, therefore using no environment prompts or assistance, in order to keep the process practical. All write-up documentation is transcribed using [Donald 'Don' E. Knuth's TEX]. Or as some [neomoderinists] like to use, LATEX. Due to time constraint, there is unfortunately no mastering the fine art of [GNU Troff], so transcriptions may not appear [optically optimal] without the famous Groff-PostScript [multi-kill].

1.5 Disposition

So forth, the following elements of this project are responsible for...
2 Research & Evidential Background

Explores the areas of research and data gathering including hiking routes, hiking equip-

ment, personal fitness and ability, geography and geology. Furthermore, presenting and examining results and conclusions to evaluations of currently existing competitors' services. This section acts as a literature review would in a paper based on, say, an empirical piece investigation; providing the foundational material upon which development aims to further succeed and 'develop'.

3 System Requirements

Describing the scale and scope of the users and their requirements for this system, and mapping how these are prioritized at the beginning of and throughout the project. This is in the context of functional and non-functional requirements.

4 Data Gathering

Mapping and justifying the data selected for use in the system. This is broken into three segments, first being data acquisition which explains how and why data is sources from third parties and inputted from users. The second section explains how this data is manipulated and the third; statistical and informative output.

5 System Design

Displaying the structure of the system architecture and how the logic aligns with the requirements of the system. Also, describing the various aspects of the user interface's functional and graphical communication and design process. It's apparent at this point how the data structure is made relevant to the design of the system using the requirements.

6 System Construction

Providing a closer look at and justification of the development environment, languages and protocols selected for the creation of this system and exploring the various APIs and JavaScript libraries and other supporting tools used to enhance the system and allow it to function in harmony. Also, Providing an overview of how these elements were implemented from a project management point of view.

7 Evaluation

An evaluation of requirements gathering and the feasibility and tangibility of their implementation, an review of self-testing methods and additional tests, demonstrations of prototyping and various other aspects of developer' and user-centered testing.

8 Development Conclusions

Summaries of development conclusions which are presented pragmatically as objective, critical notes and possible segues.

2 Research & Evidential Background

2.1 Areas of Exploration

This system is designed to combine and present aspects of the different types hiking routes and their attributes relative to their conditions; the recommended and available equipment for users to investigate and explore expansive opportunities within; the personal fitness and ability level of users and therefore, their ability to interact with different routes and opportunities; and, the geography and geology of various aspects of hiking routes which contributes to various other factors within user ability how users may interact with the routes themselves.

2.1.1 Hiking Routes

The hiking routes are the foundation of this system. They provide the purpose and reason to the GPS aspect of the system. There are various demonstrations of how hiking routes are implemented in different ways across slightly different platforms. Walkhighlands for example, presents very static use of these; displaying a page per routes listing manually expressed data and literature. Each route cannot be interacted with and had no dynamic attributes. They are simply listed for user interpretation. In this system, GPS routes are made relevant to the particular user interacting with them. Routes are not only selected through subjective choice, they are dynamically relevant to both user conscious and subconscious attributes.

2.1.2 Hiking Equipment

Hiking equipment is arguably half the battle when it comes to most effectively tackling projects. Although I've had my fair share of 15 mile proj's with approach shoes and one litre of water, alongside [Griff] in shredded boots and MTB tee-shirt and shorts; it's still pretty important. To the [Maddie Owens] of the industry. Regardless, many routes require particular components and combinations of equipment, including the appropriate knowledge of such. This means it is completely necessary that, especially under-experienced hikers, are as aware of the precautions and hazards present on selected and suggested routes. Including a metric which accounts for the user's equipment, alongside relevant literature, ensures that this site takes the implements the correct protocol to see that the user does not make any unrealistic inference regarding routes. Once again, in other services such as Strava and Garmin Connect, there are not metrics which account for these attributes. Within Walkhighlands, there is plenty of literature available

however, this information is not quantified and translated into data input so therefore leaves routes static, relative to equipment.

2.1.3 Personal Fitness & Ability

Fitness is undoubtedly the single most important factor in any sport. Skill, knowledge, technique, understanding of kinematics and dynamics of the human body, and things alike all contribute to the degree to which you excel at a sport. However, without raw fitness you might as well sit on the bench. Keeping the heart rate regulated, understanding which parts of your body to engage and not to engage, correctly distributing force and converting torque are all closely related to personal fitness and therefore must be quantified in such a way which reflects a user's expected effort and ability to complete a route. This effects results such as elapsed time, breaks required, fatigue and estimated recovery time, etc. Again, route planners such as Walkhighlands do not offer any form of input using these metrics. Strava and Garmin Connect do however make estimates following completion of activities however, do not allow these statistics to be re-used and inputted as variables determining results of estimates of future activities.

2.1.4 Geography

The use of geography within this system is fairly static and informational. That is, it is not quantified and it's attributes cannot be used as inputs which determine future estimates and results. As physical geography is out of the control of user's however, there wouldn't be much use in quantifying it. It is however useful if users have an understanding of what geographical features are and how they can have an impact on their routes. Of course this particular feature is irrelevant to much of Strava and Garmin Connect's functionality and is therefore not included in any form. Walkhighlands does include excellent educational sources however there is not much continuity and consistency to their presence. Therefore, relevance is often unaligned.

2.2 Material Investigation & Heuristic Evaluations

As discussed, improving upon various features and aspects of Walkhighlands, Strava and Garmin Connect is a relevant step in developing requirements and informal desires from this system. It goes without saying that as an inexperienced sole developer, these improvements are not based on functionality and code efficiency etc. This would be intangible. Improvements are primarily focused on making particular features more relevant, accessible and usable. The three services under examination are significantly

more advanced and expansive than this system is at the end of development. Therefore, features which these services include but this system does not will not be examined. So forth, the services will be examined purely under the scope of this system. That is, basic services (overview, ability, equipment) integration; map services (OS map, GPX file and map feature) integration; and, statistical processing (route and mountain information, and personal data processing).

As an alternative to formal empirical methodologies, Nielsen (1994) proposes a critique-based method based on a heuristic evaluation which involves analysis based on areas of expertise. A heuristic evaluation of one's own system is also argued to be a useful method of allocating time to minor issues before final user testing. Nielsen also claims that the optimal number of 'experts' assigned to an evaluation is three-to-five in order to find the 'optimal' number of issues relative to the cost-benefit analysis. In this case however, one examiner is used for obvious reasons. Each issue is individually listed and valued against the set of ten heuristic factors and assigned a severity rating, as seen in Table 2.1.

The analyses conducted subsequently are not exhaustive however, are relevant to the context and features in this system. There is no heuristic evaluation for Garmin Connect as any differing functionality from Strava is more advanced than that which this system accounts for and therefore, does not need to be evaluated.

Heuristics
H_1: Visibility of System Status
H_2: System-Real-World Match
H_3: User Control & Freedom
H_4: Consistency & Standards
H_5: Error Prevention
H_6: Recognition Rather than Recall
H_7: Flexibility & Efficiency of Use
H_8: Aesthetic & Minimalist Design
H_9: User Recognition, Diagnostic & Recovery from Error
H_{10}: Help & Documentation
Severity Ratings
S_0: Don't think it is a usability problem
S_1: Cosmetic issue; repair in additional time
S_2: Minor usability problem; allocate low priority to repair
S_3: Major usability problem; allocate high priority to repair
S_4: Critical error; repair immediately

Table 2.1: Heuristics & Severity Rating

2.2.1 Walkhighlands

The heuristic evaluation for Walkhighlands is listed under Figure A1 in Appendix 1.

2.2.2 Strava

The heuristic evaluation for Strava is listed under Figure A2 in Appendix 1.

3 System Requirements

System requirements gathering is essentially the stage at which the concept of a system's functionality is aligned with real-world user desires (requirements). This stage provides a context for creativity and a blueprint upon which this can be mapped. Although the founder of a system or concept may have a clear vision of the intended outcome of their development, understanding what final users need and want allows the creator to constantly tailor development. This may extend to how they and/or the software could/should gather information, store data, transact data, and output data/information.

3.1 Scale & Scope

I repeat, this is a [solo project] so there are some restrictions regarding the overall scalability of the project as a whole. If this system were to be designed by a team of professionals, it would be very large-scale and offer much expansible functionality. However, due to the number of personnel assigned to the task (a.k.a. me), the time constraint, and budget constraint, the system finds itself with two major general downscales: [1] there is no user-data back-end, meaning user accounts are unavailable on this system, which is acceptable as the static functionality of the site is most relevant; and, [2] the sample region for data collection is significantly smaller than that offered by other services, which is also acceptable as adding a wider scope of data (to the master JSON in this case) would only consist of repeating the same patterns perpetually. As this system does not aggregate this data for any form of cross-sectional statistical analysis, larger sample sizes become irrelevant after a certain point. Overall, this implies these factors are not directly related to any implemented requirements. Many of the specifics of these restrictions, and others, are discussed latterly in the *Evaluation* section.

In an ideal world, users would be able to expand their scope of interactions within the system (i.e. different sports with a wider and varying array of attributes), and the individual scale of these. [This means that] as sports differ, attributes and statistics differ, and information and guidance differs; offering a better-rounded service. Ideally, users should have the opportunity to fully customize their experience however, to do this on such a scale feasible with the development team available (myself) would be a significant over-effort for an under-achievement. This is why user accounts have been disregarded. This decision helps keep the system and implementation of requirements more manageable and makes it easier to achieve a polished product within the time

constraint.

3.2 Gathering & Prioritization Methodology

In this scenario, there are three methods of requirements gathering and inference. The first is the purely user-centered method. As I am surrounded by people who share an interest in the form of this system, including professional developers, student developers and various other [NPCs] of the sort, they act as an accurate representation of market users as they share the same attributes. The second mode is alike however, is argued to be subject to various aspects of contextual bias. This consists of creative direction explicitly from the creator, me. In this context, these inputs will generally align with those of the formerly discussed however, due to the bias, is not considered a viable user-centered method unless used in conjunction with others affirmative of said criteria. The final method involves basic inference from the *Research & Evidential Background* section. That is, much of the material investigation of Walkhighlands, Strava and Garmin Connect in this section highlights areas for linear development. These refer to aspects which generally do not require user-centered input and must be developed purely functionally.

Many aspects of this system are purely functional and exist to serve an objective purpose. For example, relaying GPX and JSON data related to components such as route and weather information. [This means that] the primary functionality of the system is majorly accounted for in the latter of the three requirements gathering methods, in that the goal of the system is to create wider-scoped versions of much of the existing content in the explored areas. Therefore, the two former methods generally account for improvements which can be implemented upon these predecessors throughout development, which are primarily focussed on enhancing the user's experience. Myself and the discussed group of experienced others are a credible source for this, considering the scale of the project.

Randomly assigning requirements to development would be irresponsible. To help better-address the importance of the components of the required functionality of a system, requirements should be analysed using some form of hierarchical tool which highlights a clearer path for the development process. In this case, the Must-Have/Should-Have/Could-Have/Won't-Have (MoSCoW) methodology is selected for this purpose. This helps differentiate between what functionality is essential to make the system run as intended, what functionality is required for optimization of the system, and what

is required for additional enhancements. This approach generally shows functional requirements to average at the top-priority end and non-functional requirements to be distributed further down the hierarchy. With regards to the framework itself, note that:

- *Must-Have* implementations refer to aspects of the system which must be present in order to make it basically functional and behave as it is intended and as the user desires;
- *Should-Have* implementations refer to features of the system which should be implemented in order to make the essential features of the system more accessible and useable to the masses. They may also exist to improve efficiency, but [fly Under the Radar] and therefore go unnoticed by the user. They may offer additional functionality which makes the system more unique (or something of the sort) and therefore, more 'creative' and attractive to users.
- *Could-Have* implementations refer to aspects which may become present or relevant during the development of the former two. They may add additional functionality or usability to existing aspects or simply add final touches to the system overall. They are sometimes more contemporary.
- *Won't Have* non-implementations refer to aspects of the system which aren't necessarily impossible or are of a nature which the system 'can't have' but, which will probably be omitted or postponed due to constraints such as time, man-power, technical ability, finance, etc.

There is no formal client base for this system which means that no face-to-face client-oriented interviews or surveys can take place with regards to determining specific user desires. However, the aforementioned three-method protocol leads to the subsequently discussed requirements. To reiterate, the following list of requirements is generated through discussion of the desires of [1] an existing group of users of systems alike (some of whom are developers), [2] myself, another existing user of the sort, and [3] analysis of other services. This list is of course not exhaustive, as there is always room for perpetual development. However, it does account for every currently visible desire and possibility given the constraints.

3.3 Requirements

As discussed, it's important at this stage to clearly differentiate between essential functionality for the foundations of the system, generally accounted for in *functional requirements*; and functionality which more contemporary, generally accounted for in *non-*

functional requirements. This is often considered 'making things *work* and making things *relevant*'. It is also a useful method from which to infer associations between elements of the design and construction stages of development.Retaining these requirements, their position, and hierarchy at the center of development throughout the process allows accurate amendment of software and/or requirements along the way as more functionalities and possibilities become apparent and tangible or alternatively, further from reach.

3.3.1 Functional

Requirements here are presented in a numeric format. This does not refer to any hierarchical order, it simply creates a reference point for user stories. Hierarchy remains determined by the MoSCoW methodology. So forth, as required by the three discussed groups, users and the system *must have* the ability to:

1. Access the user's current location upon various types of request
2. View an 'overview' of recommended routes
3. Accept inputs related to user 'abilities' and compute and display results based on them
4. Accept inputs related to user 'equipment' and compute and display results based on them
5. View a weather forecast breakdown using real weather data
6. Display an Ordnance Survey map

Additionally, users and the system *should have* the ability to:

1. On OS Map, alternate between 'OS Leisure' (primary/default), 'OS Road', and 'OS Outdoor' topographical structures
2. On OS Map, display grid markers at center-pan

Additionally, given the various constraints, users and the system *could have* the ability to:

1. Create an account and have their data stored
2. Protect user data using an appropriate authentication and security system
3. Store and re-use user attributes such as the aforementioned 'ability' and 'equipment cache'

4. Choose 'priority attributes' relevant to their routes which are stored and used to generate more relevant route recommendations etc.

Additionally, users and the system *won't have* the ability to:

1. Store route-relevant data and display them as 'historically completed routes', or something of the sort, under the aforementioned 'overview'

3.3.2 Non-Functional

To ensure a stable, trustworthy, useable and relevant system; users and the system will have the ability to / have the capacity to:

1. Be written in such a manner which allows full compatibility with the majority of web browsers (this could effect markup languages, font packages, JavaScript libraries, etc.)
2. Be written in such a manner which allows full scalability between desktop and mobile use
3. Offer the user various descriptive background [pieces], otherwise referred to as 'key's and 'suggested reading's which provide aid to users' background knowledge, understanding and decision making
4. Offer the appropriate combination of relevant graphic design and actual functionality
5. Adhere to the appropriate accessibility standards, with reference primarily to graphic design and system structure

3.3.3 User Stories

'User stories' are often an effective method of contextualizing requirements and presenting how they can and will be interpreted and used by the final user. Putting yourself in the hypothetical context of a user is a good method of delivering basic feedback on how to apply a solution to a requirement. Additionally, it helps identify possible negations or irrelevant content. So forth, the user stories found in Figure A3 in Appendix 1 follow the syntax: "I wish to be able to <interact_with_feature> in anticipation of <returned_result> which will provide me with <payoff>". Development of the solution to these user stories listed in the figure are prioritized using the metric seen in the *Priority* column.

3.4 User Accessibility

This system is available as a website application, it could be accessed and used by anyone with access to the internet. Therefore, it is to some degree essential that users of all types are welcomed to the site, even if they have no prior knowledge of the industry or field. More importantly, the graphic design and methods of interfacing in the user interface should be accommodating of users with possible imparities, such as a [severe mental imparity], by taking a user-centered approach which considers visible accessibility controls which allow these users to interface with the system more easily. Accessibility does not however only refer to disability. Ensuring optimal accessibility also accounts for factors such as implementing clean and concise CSS and graphic design for simple and intuitive navigation, stimulation, relevance, etc. These approaches are widely considered and implemented.

In context, whether using the desktop or scaled mobile version of the site, the user interface implements many 'friendly' universal standards. For example, it primarily uses the readable and elegant Sans-Serif font Audi AG and Serif font Garamond, with suitable font-sizing and color etc. Hue, saturation and luminance are appropriately considered/assigned to create [maximal optical pleasure] and proper readability with the correct contrast between elements. The well respected font-family Font Awesome, which is universally recognised, to display various standard symbols. In theory, these considerations lead to more efficient understanding and transitioning between elements of the site.

4 Data Gathering

4.1 Data

A program has no applicable context without the appropriate data. In this case, data is required to even present the base functionality and content related to many features. That speaks loudly for the fact that this system is primarily made of JavaScript and provides users with no utility as a pure site. At least it's static however, if that's any compensation. Seen in Figure A4 details a brief overview of all the data sources, purposes and uses throughout every component of this system, which are discussed under the seven subsequent sub-sections of this section.

4.1.1 Location

A digital mapping system for mountain navigation will never function as required if no location services are in place. And, at the heart of location tracking with map APIs and other services alike, is Geolocation. Once a call to `navigator.geolocation` has been made which interfaces with the browser being used, the user is prompted to grant access to device location services. Once this is confirmed, GPS data is easily accessible. The two important values captured using Geolocation are of course latitude and longitude (`lat`, `lon`), which are used in two important ways. The first being `Geolocation.getCurrentPosition()` which is useful for using latitude and longitude in calculations such as measures of distance, historic placement tracking, etc. The second is `Geolocation.watchPosition()` which is a method of tracking location constantly upon update, clearly more useful for icon display. This is the simplest form of data collection in the this system as it only requires an API call and perhaps a few more lines to manipulate a result.

4.1.2 Regional

The regional data refers to GPS coordinate records of [1] all counties of Scotland, [2] all regions of Scotland, [3] all sub-regions of Scotland, and [4] all sub-sub-regions of Scotland. For example, *Moray* is a County; *Glenfinnan* is a sub-sub-region, in the sub-region *Fort William, Lochaber and Lorn*, in the region *Highlands*. All of this regional data was acquired by inspecting the 'Search all walks' section of Walkhighlands (Walkhighlands, 2022), although did require some hierarchical and grouping amendments, especially throughout the [Western Ocean] and Western Isles. It is stored in a

JSON file created locally by myself and hosted on my GitHub. Regions are ordered south-to-north in the file.

4.1.3 Landmass

Landmass data is of a similar format in that is stores records of GPS coordinate data upon the landmasses of Scotland. However, it extends to be far more complex. It includes data on landmasses themselves, which are anything resembling some significant land-form of Scotland. In this context, a landmass may either be a *Mountain*, *Mountain Range*, or *Stand Alone*. A mountain is a single prominent feature which contains one or more hills without a contour drop of under 2000ft between summits; a mountain range is a group of prominent features which contain hills without a contour drop of under 1200ft between summits; and, a stand alone is a single prominent feature which contains one hill. 'Hill' refers to *Munros* and *Corbetts*, not *Munro/Corbett Tops*. Therefore, tops are not limited on stand alones but also do not determine a summit. For example, "a stand alone landmass may have one Munro and two Munro Tops". Note that *Munro Tops* and *Corbett Tops* sit at the height of Munros/Corbetts respectively however do not meet the elevation and distance criteria to be classed as their own hills. They must be present at maximum stationary points, not increasing points such as those at which cairns are often found. All attributes of landmasses and hills etc., and their hierarchy can be found in the breakdown of the JSON file in Figure A4.

Data was gathered, [wait for it...], primarily manually by myself by staring at OS Explorer 376[1], Explorer 377[2] and Explorer 384[3]. This provided access to information not yet recorded, such as Corries, Lochains and boulder fields of a landmass, for example. To ensure inclusion of all Munros and Corbetts, reference to Walkhighlands was made (Walkhighlands, 2022); and, to ensure inclusion of all Munro and Corbett Tops, Harold Street (2022) and Peakbagger.com (2022) were also investigated as they include more comprehensive community-driven data. Once again, data is stored in a JSON file hosted on my GitHub. Landmasses and their components are ordered south-to-north in the file, which will become more relevant subsequently.

Under the time constraint of the project, some omissions, amendments and notes were made as follows:

[1]Oban & North Lorn (Figure B4)
[2]Loch Etive & Glen Orchy
[3]Glen Coe & Glen Etive

- The sample size is limited to the South-West Highlands, which refers to the region spanning Glen Etive, Glen Orchy and Glen Coe, a.k.a. the land between the A85 to Oban and the A82 through Glen Orchy, through Glen Coe, to Ballachullish (side note: don't confuse 'Glen Coe' with 'Glencoe' in the context of this data! Critical Errors will occur. Glencoe is the village, Glen Coe is the glen. And we aren't Mr. Walkhighlands so it isn't ['Glinco']).
- Sub2000s, Donalds, Grahams and Grahams tops do not appear in the data as this would have taken an inappropriate amount of time to collect and the six-thousand-line JSON was enough already.
- Some details for Corbett Tops and Munro Tops, are not provided as even the most of the aforementioned 'detailed' sites were lacking some data.
- Munros, Munro Tops, Corbetts and Corbett Tops are ordered primarily by category and secondarily from max elevation highest-to-lowest within their category.
- There is no pre-determined 'order' to Corrie hierarchy on landmasses so they are simply ordered by myself from south-to-north.
- Ideally, Corries, Gullies, Lochains and Waterfalls would be named and assigned GPS coordinates in the JSON file. However, OS maps were too inconsistent with naming the features so only the 100% consistent Corrie category is given this luxury. Others are simply assigned boolean values.
- HTML symbols such as ò (`&ograve;`) and ê (`&ecric;`), which appear frequently in Gaelic spellings, are unfortunately not favoured in the JSON or any other iteration of the relevant words, as much time is saved developing functions and methods to search; loop through and match the data.

4.1.4 Route

Route data is delivered in two different forms. The first is much like the former, in fact as seen in Figure A4, the route attribute appears as a component of a landmass; "a route on a landmass". A route has the standard components such as a name, distance, total elevation gain, estimated duration, and of course the hills included on the route, etc. The route data was again gathered from Walkhighlands (2022), in a combination of official Walkhighlands routes, community-contributed Walkhighlands routes, and in some cases where there was no existing content, my personal Strava (often my historic record and knowledge also).

Another component of a route is its GPS data. Global Positioning System Exchange Format (GPS Exchange Format (GPX)) is the most commonly used file type to store GPS data uploaded to Walkhighlands and Strava. Therefore, these files were selected again from Walkhighlands' official routes, community-contributed routes and my personal Strava, and associated accordingly. Relevant paths to these files are included as attributes under the appropriate routes in the discussed JSON file. Note that JavaScript operates most efficiently using a GeoJSON GPS data file format therefore, upon use they're converted. Due to routes being components of landmasses and the south-to-north ordering of landmasses, routes also occur south-to-north in the file.

4.1.5 Ability

Ability data is once again manufactured data, by myself, relevant to how different people may interact with landmasses, their features and their routes. It offers characteristics relating to numerical difficulties of hills routes such as distance, elevation; summit features; types of route; stages of routes; terrain types; and terrain difficulty factors. Included are various strings describing these aspects and their purposes and implications, etc. File paths to supplementary images are also included. There isn't necessarily a data source in this case, apart from my endless hours spend in the Scottish Highlands. All of this data is stored in a JSON file which is hosted on my GitHub.

4.1.6 Equipment

Equipment data follows the same format again. I is data manufactured by myself which refers to a very comprehensive list of equipment including packs and travel equipment; technical equipment; footwear, footwear components and crampons, etc.; and clothing. It's constructed of basic strings describing the good and its purpose, with a file path to a supplementary image. The only source for this portion is countless hours spent browsing Tiso and talking to other enthusiasts and experts. The data is contained within a JSON file hosted on my GitHub.

4.1.7 Weather

Weather data is another [key component] in mountain planning. It would be unreasonable to manually gather weather data from a larger source such as the Met Office however, OpenWeather (OpenWeather 2022) offers a comprehensive API which covers all of the relevant locations to this system. All weather data used in this system originates from the OpenWeather One Call API, which provides a wide selection of minutely,

hourly and daily data.

4.2 Information

In the context of the subsequent output of processed data, sometimes the limited parameters in which it is and can be presented is too restrictive. Hence, ongoing research is conducted into each of the elements of the theme of this system to gather and provide additional nominal guidance. This nominal guidance must then by applied in a context by the user to make it relevant to their engagements. For example, this extends to research relating to the different elements of the weather feature and displaying their definitions and origin for the user in a 'key'. So far, as this process is ongoing and will continuously be added to throughout the lifespan of the system, the aforementioned key is included along with 'suggested readings' on cloud types and suggested readings on compass bearings. These elements are designed not to offer a more advanced system, but to provide a more comprehensive experience and enhance knowledge, application and experience.

5 System Design

5.1 System Architecture & Concept

This system uses the modern 'framework' of an attractive front-end which simplifies [for the user] and allows access to more complex data transactions. Although this may not be through a full-back-end in this context, supporting files and data libraries act in this role accordingly. Through an appropriate and relevant front-end, users communicate through various input methods and requests in order to communicate with and manipulate data in the system for a contextualized and beneficial output. That is, data which is displayed in an informative and relevant manner which can aid a user's experience in the real-world context of the system, the field (hills), based on their use of it. This process strictly adheres to analysis of requirements and functionality.

After consideration of the tangibility, feasibility and relevance of requirements, understanding how to most efficiently and easily allow a user to interact with these elements is the most [crucial component]. Therefore, the most logical approach to designing the system is to begin experimentation with the user interface. This then highlights areas in which user-defined requirements and functionality is relevant and what data must be acquired for their implementation. Once the dataset had been identified and mapped, this system took the approach of being as comprehensive as possible while using the least amount of UI [real estate], and therefore the least amount of code, possible given the dataset. This makes for a more efficient system and is therefore why this one is limited to two HTML web pages.

Note that this system was originally intended to include three pages, the third being 'Ranger' which is a series of user-input-based and pre-defined graphing functions based on the Chart.js (Chart.js, 2022) JavaScript library. This was omitted however, due to time constraints and lack of relevance to the brief. Requirements remain in-tact as the idea was pitched originally to potential users and the component will likely be developed for, at least, personal and friend use post-project-completion. So forth, this stage involves fast prototyping; continuous user, practical and command line testing; and, rapid change. This is discussed at length in the *Evaluation* section. Given all time, technical and financial constraints, the correct protocol was executed.

5.2 User Interface

5.2.1 Logical Design

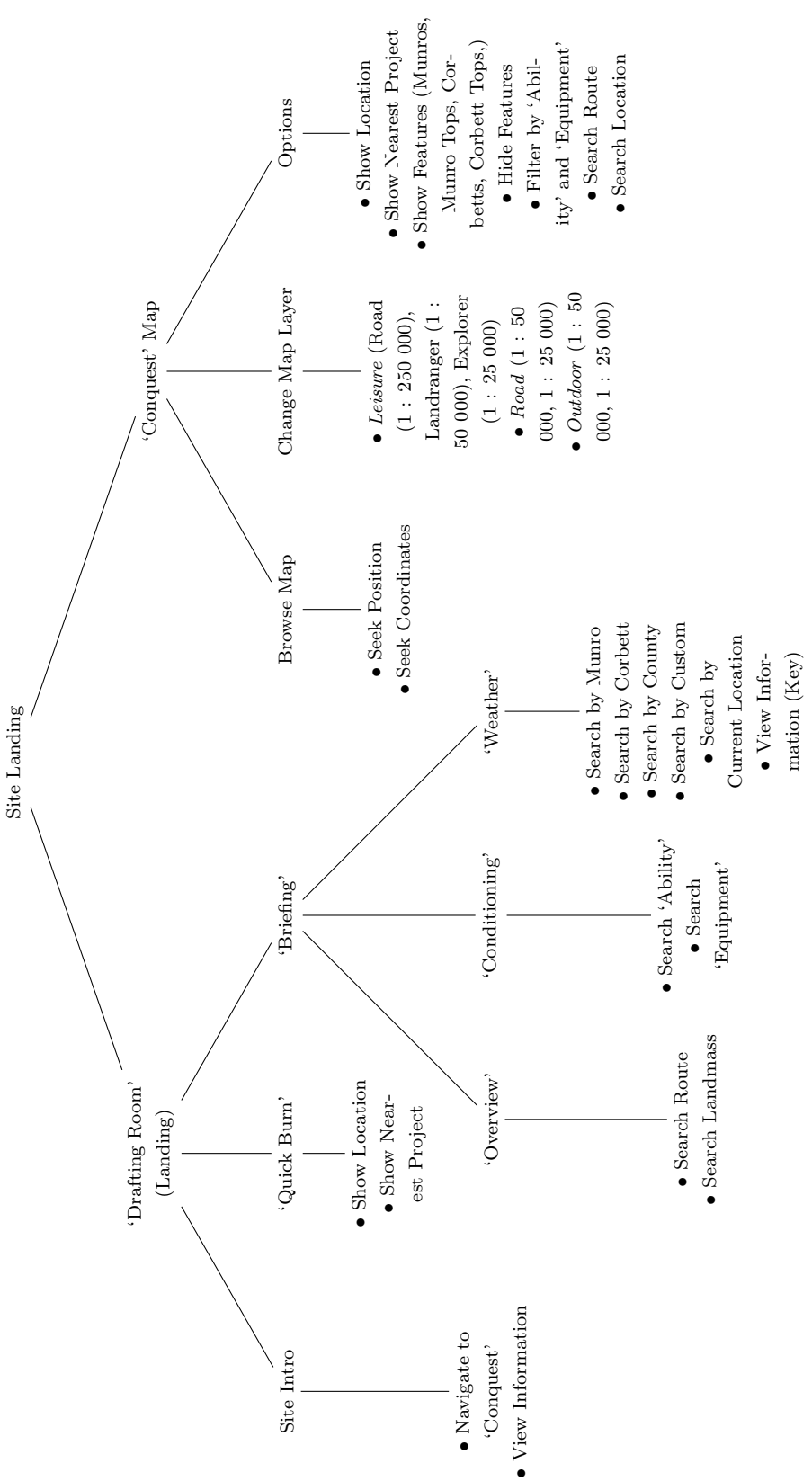
This system is based on a choice and is therefore inclusive of and relevant to all aspects of demographics ranging from age groups, personalities, genders, races, etc., who simply have a passion to practice or learn about the Scottish Highlands. This implies that the system must be as widely available as possible, meaning it should be accessible on both desktop and mobile platforms which meet two criteria: they must [1] be capable of supporting the user's choice of media such as their web browser, and [2] be available on the user's choice of device which is accessible in the field (i.e. if using the system's GPS features while on a hill [conquest]).

Hence, the system design as a web-based application, with JavaScript as the language delivering the required 'functionality'. This implies that HTML delivers pure functionality and informative purposes and CSS delivers [bloat] the modern aspect of engagement, interactiveness and [percieved] 'attractiveness'. Functionally, this site would look like [Harvard.edu] in 2007, but that approach would only benefit myself and the few others who remain of my kind. This all implies that the desktop site is intended for the use of the 'planning and informative' stages of hiking, such as using the map to plan routes and guides to learn about them; and, the mobile site is intended for navigational (GPS) and safety use while in the field. Given a full development team, time and financial input, this system could be developed as a full mobile and application, as well as in-browser, in which GPS features can be used offline as many user's may find themselves lacking internet connection in the field.

5.2.2 Logic Tree

The following recursion tree highlights the paths of interaction with the interface of this system. The tree's nodes do not exhaust listing of the system's functions however, they do exhaust listing of the UI's elements which interact with the functions. Keep noted that this system is very linear and only includes two pages. The hierarchical structure and organization of these pages is what give them their versatile functionality. The system does not perfectly fit the definition of 'object oriented' although it heavily relies on JavaScript. Therefore, a navigation map is a more logical demonstration than, for example, a use case diagram of how this system is used.

Turn & Rotate Page


Site Landing
'Drafting Room' (Landing)
Site Intro
• Navigate to 'Conquest'
• View Information
'Quick Burn'
• Show Location
• Show Nearest Project
'Briefing'
'Overview'
• Search Route
• Search Landmass
'Conditioning'
• Search 'Ability'
• Search 'Equipment'
'Weather'
• Search by Munro
• Search by Corbett
• Search by County
• Search by Custom
• Search by Current Location
• View Information (Key)
'Conquest' Map
Browse Map
• Seek Position
• Seek Coordinates
Change Map Layer
• *Leisure* (Road (1 : 250 000), Landranger (1 : 50 000), Explorer (1 : 25 000)
• *Road* (1 : 50 000, 1 : 25 000)
• *Outdoor* (1 : 50 000, 1 : 25 000)
Options
• Show Location
• Show Nearest Project
• Show Features (Munros, Munro Tops, Corbetts, Corbett Tops,)
• Hide Features
• Filter by 'Ability' and 'Equipment'
• Search Route
• Search Location

5.2.3 Graphic Design & Communication

[Peter goddamn "Crimp" Roy], how I miss him. He would agree that a system's graphic design is often overlooked. Especially in very small scale, more functional developments such as this, where there is lacking knowledge of or specialization in the artistic practices; this aspect is, almost all of the time, outsourced to commercial graphic design teams. In this scenario, all graphic design was crafted in-house using CSS for the site directly and desktop publishing (DTP) and photo manipulation software such as GIMP and Serif Page Plus for additional graphical content. Elements and principals of graphic design such as color schemes, shape, texture, balance, etc., are all used to accordingly represent aspects of the associated landscapes and dynamics.

5.2.3.1 Cascading Stylesheet (CSS)

Being a website, CSS is the most important styling aspect. It serves two primary functions: [1] make a site relevant and contextualized, and [2] make a site accessible. These were achieved by implementing aspects of *color* to make the surrounding foreground of the site pop in a manner such as that of a mid-2000's military briefing room. That background remains shades of whites and the foreground shades of grey, which also allows the natural colors of the OS map and other features alike to be further *emphasized* and more easily viewed. Gradients were used in appropriate places to continue the theme of *motion*. The element of *motion* was used in the logo's entraince to the site where it's seen to enter in an animation from left to right, highlighting elements of movement discussed in section 5.2.3.4. Fonts used also play a similar role in relevance to themes. Font packages used include the Sans-Serif *NFS* and *Audi Type Variable*, Audi's famous typeface, were used in adding a modern 'sporty' feel to a service which traditionally projects itself using Garamond EB and no styling. Both of these fonts are detailed in Figure 5.1.

(i) NFS Typeface (ii) Audi Type Variable Typeface

Furthermore, FontAwesome (FontAwesome, 2021) was used in simplifying the user experience by implementing recognizable and relatable symbols in places where text would

create too much [bloat] and unecessary reading. This also aids accessibility and imporves navigation. Common uses of these are detailed in Table 5.2 below.

...

Table 5.2: FontAwesome Symbols

Finally; hue, saturation and luminance are all appropriately assigned values to elemetns of the design to allow optimal readability and minimal strain. When viewing the main portion of the site, the OS map, there isn't much exposed blue light. It is of course, as always, recommended that a user adjusts the gamma output of their display accordingly. Improvement can always be made in areas of accessibility. Some may include JavaScript functionality which allows the user to control the font size, colors, element sizes, etc.; features alike. Given the constraints, brief and target users (myself and a marker), the time-reward trade-off was not valueable.

5.2.3.2 Desktop Publishing (DTP)

This is an unrequired portion of the project and exists to serve a purpose defined long before completion. The promotional piece is out of date however, is still beautiful and is displayed in Figure A5 for your leisure.

5.2.3.3 Photo Manipulation

The primary use of GIMP in this project was to design an appropriate logo to-be implemented on every page of the site, which relates closely to it's themes. This is seen below in Figure 5.2.

Figure 5.2: Logo (Made in GIMP)

The logo considers four relevant layers: text, shape, color, effects. The font (*text*) represents an association with the 'outdoor/natural' life theme of the system in it's rustic semi-Sans-Serif nature. *Shape* is used in the 30° slant of the text, representing perceived distortion of space around the user and 'natural ([420 B L A Z E ∗ I T])

high' experienced while moving at a fast-pace in the field. A red *color* is used in the 'burning' word to *harmonize* with the word's semantic definition and *emphasise* the 'burning desire' for aspects discussed in the *Introduction* section. A reverse-distress overlay (grunge effect) is used on the 'roots' word to *emphasise* the diverse combination of 'worn', 'distressed', 'fatigued', 'excited' and 'accomplished' emotions experienced by the user in the field. This process is shown below in Figure 5.3.

(i) Pre-Grunge (ii) Post-Grunge

Figure 5.3: Distress Process

5.2.4 Use Cases

...

5.3 Data Application & Manipulation

All data discussed in Section 4 of this report is included in a basic input-output system [(BIOS, wheyy)] which, through functions and others methods, produces the desired outcomes which satisfy user requirements. A summary of this process can be seen Figure A6. Reference to data can be observed in Figure A4.

5.3.1 'Overview'

The 'Overview' portion of the home page of this site refers to a more basic iteration of the route search and landmass search functions, seen subsequently in the 'Conquest Map'. For the sake of continuity, all basic features of these functions' use of data is discussed under this heading; additional data required for the 'Conquest Map' version are discussed latterly.

5.3.1.1 Score Route

This function exists purely to serve the Search Route function and has no purpose as a stand-alone. As it involves quite the lengthy arithmetic process however, it is granted its own sub-sub-sub-section. Although, to my judgement, this metric is fairly mathematically sound although will be constantly developed as I generate more ideas, it is

still very subjective and based on my own experiences with and discussions about the Scottish hills and their conditions. Therefore, this 'score' is not a definitive metric to describe the condition of a route to the masses. It should be judged and considered on an individual basis.

Henceforth, scoring a route takes into account [1] four groups of subjective scores based on weights and values of the four route element factors seen under routes in the JSON file (type, stages, terrain types, terrain difficulties), and [2] a more objective and logical multiplier based on total number of [Munro, 'Top, Corbett, 'Top] tops covered on-route, the total elevation gained, and the total distance covered. Actual values of the weights were gathered through personal consideration and conversation in the field and is therefore subject to constant adjustment. So forth, the multiplier was developed purely by myself. All parameters of this equation are as follows:

1. Conversion constants (ft–m, mi–km) – values
2. Number of tops – value
3. Elevation – value
4. Distance – value
5. Route type (weight, {weights, 'values'}, 'score') – matrix
6. Route stages (weight, {weights, 'values'}, 'score') – matrix
7. Terrain types (weight, {weights, 'values'}, 'score') – matrix
8. Terrain difficulties (weight, {weights, 'values'}, 'score') – matrix

This site using the traditionally functional imperial system therefore, as this computation actually calls for the one and only reasonable use of the metric system (mathematical scalability), conversion constants are required for elevation and distance. These are both justified to four significant figures. Mathematical notation for each of these processes follow (excuse the notational accuracy, it's been a while).

[1.1] Conversion Constants:

$$c_{ft} = 0.3048$$

$$c_{mi} = 1.609$$

Where:
c_{ft} = Conversion constant for imperial feet to metric meters
c_{mi} = Conversion constant for imperial miles to metric kilometers ($\times 10^3$m)

[1.2] Number of Tops:

$$N_{\text{Tops}} = \sum_{i=1}^{I} N_i + \sum_{j=1}^{J} N_j + \sum_{k=1}^{K} N_k + \sum_{l=1}^{L} N_l$$

$$\vee \; N_{\text{Tops}} = |i| + |j| + |k| + |l|$$

Where:
N_{Tops} = Total number of tops present on given route
$N_{i,j,k,l}$ = Numerical value representing the position of a string element i, j, k, l in arrays of i: route type, j: route stages, k: terrain types, l: terrain difficulties; $i \in \{1, ...I\}$, $j \in \{1, ...J\}$, $k \in \{1, ...K\}$, $l \in \{1, ...L\}$

[1.3] Elevation:

$$\text{Elev}_m = c_{ft}\text{Elev}_{ft}$$

Where:
Elev_m = Total elevation gained on given route, in meters (m)
Elev_{ft} = Total elevation gained on given route, in feet (ft)

[1.4] Distance:

$$\text{Dist}_{km} = c_{mi}\text{Dist}_{mi}$$

Where:
Dist_{km} = Total distance covered on given route, in kilometers (km)
Dist_{mi} = Total distance covered on given route, in miles (mi)

[2.1] Weights:

For Route Type's, Route Stages', Terrain Types', Terrain Difficulties' Elements

This method defines equally weighted elements by dividing 1 by the number of elements in the arrays. Where i, j, k, l refer to arrays and positions within their respective arrays:

$$w_i = \frac{1}{\sum_{i=1}^{I} N_i},\ w_j = \frac{1}{\sum_{j=1}^{J} N_j},\ w_k = \frac{1}{\sum_{k=1}^{K} N_k},\ w_l = \frac{1}{\sum_{l=1}^{L} N_l}$$

$$\vee\ w_i = \frac{1}{|i|},\ w_j = \frac{1}{|j|},\ w_k = \frac{1}{|k|},\ w_l = \frac{1}{|l|}$$

$$\therefore w_i, w_j, w_k, w_l \text{ always constant and equally weighted}$$

Where:
$w_{i,j,k,l}$ = Equally weighted value based on number of elements for element i, j, k, l in arrays of i: route type, j: route stages, k: terrain types, l: terrain difficulties; $i \in \{1, ...I\}$, $j \in \{1, ...J\}$, $k \in \{1, ...K\}$, $l \in \{1, ...L\}$
$N_{i,j,k,l}$ = Numerical value representing the position of a string element i, j, k, l in arrays of i: route type, j: route stages, k: terrain types, l: terrain difficulties; $i \in \{1, ...I\}$, $j \in \{1, ...J\}$, $k \in \{1, ...K\}$, $l \in \{1, ...L\}$

[2.2] Scores:

For Route Type, Route Stages, Terrain Types, Terrain Difficulties

$$S_i = \sum_{i=1}^{I} w_i V_i,\ S_j = \sum_{j=1}^{J} w_j V_j,\ S_k = \sum_{k=1}^{K} w_k V_k,\ S_l = \sum_{l=1}^{L} w_l V_l,$$

Where:
$S_{i,j,k,l}$ = Relative score of element i, j, k, l in arrays of i: route type, j: route stages, k: terrain types, l: terrain difficulties; $i \in \{1, ...I\}$, $j \in \{1, ...J\}$, $k \in \{1, ...K\}$, $l \in \{1, ...L\}$
$V_{i,j,k,l}$ = Subjectively defined value assigned to element i, j, k, l in arrays of i: route type, j: route stages, k: terrain types, l: terrain difficulties; $i \in \{1, ...I\}$, $j \in \{1, ...J\}$, $k \in \{1, ...K\}$, $l \in \{1, ...L\}$

An example for the route type array (i) which contains three equally weighted elements follows:

$$S_i = w_1V_1 + w_2V_2 + w_3V_3$$
$$S_i = \frac{1}{3}\left(\frac{1}{4}\right) + \frac{1}{3}\left(\frac{1}{4}\right) + \frac{1}{3}\left(\frac{1}{2}\right)$$
$$S_i = \frac{1}{3} = 0.\dot{3}$$

As these values are generated from an array, they involve single-column (no inversion) matrix algebra in the context of JavaScript. An example of the appropriate notation relating to the previous example follows (decimal values favoured in matrices):

$$S_i = \left[0.\dot{3}\ 0.\dot{3}\ 0.\dot{3}\right]\ [0.25\ 0.25\ 0.5]$$
$$S_i = 0.\dot{3}$$

Again, as this is single-column matrix algebra, standard notation is also viable:

$$S_i = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$$
$$S_i = \begin{bmatrix} 0.\dot{3} \\ 0.\dot{3} \\ 0.\dot{3} \end{bmatrix} \begin{bmatrix} 0.25 \\ 0.25 \\ 0.5 \end{bmatrix}$$
$$S_i = 0.\dot{3}$$

For each of the route element factor arrays, these are then aggregated to give a total value. However, they still do not have any context which is why a multiplier is used.

[2.3] Weights:

For Route Type, Route Stages, Terrain Types, Terrain Difficulties

$$w_{S_i},\ w_{S_j},\ w_{S_k},\ w_{S_l}$$

Do not confuse these with the equal weightings for each individual element within the arrays, these four weightings exist on a subjective basis to weight the four scores for

each array.
Where:
$w_{S_{i,j,k,l}}$ = Weight values determining the importance of arrays of i: route type, j: route stages, k: terrain types, l: terrain difficulties; $i \in \{1, ...I\}$, $j \in \{1, ...J\}$, $k \in \{1, ...K\}$, $l \in \{1, ...L\}$

[3.1] Complete Multiplier:

$$m_R = \frac{N_{\text{Tops}}}{\left(\frac{\text{Elev}_{\text{m}}}{1000(\text{Dist}_{\text{km}})}\right)}$$

$$\textbf{Expanded: } m_R = \frac{\sum_{i=1}^{I} N_i + \sum_{j=1}^{J} N_j + \sum_{k=1}^{K} N_k + \sum_{l=1}^{L} N_l}{\left(\frac{c_{ft}\text{Elev}_{\text{ft}}}{1000(c_{mi}\text{Dist}_{\text{mi}})}\right)}$$

Where:
m_R = Multiplier for given route R

In English, this equation reads: "number of tops (Munro, Munro Top, Corbett, Corbett Top) achieved on the route per vertical meter gained on the route, per horizontal meter gained on the route".

[3.2] Complete Aggregate Route Score:

$$S_R = w_{S_i}S_i + w_{S_j}S_j + w_{S_k}S_k + w_{S_l}S_l$$

$$\textbf{Expanded (1): } S_R = w_{S_i}\left(\sum_{i=1}^{I} w_i V_i\right) + ... + w_{S_l}\left(\sum_{l=1}^{L} w_l V_l\right)$$

$$\textbf{Expanded (2): } S_R = w_{S_i}\left(\sum_{i=1}^{I}\left(\frac{1}{\sum_{i=1}^{I} N_i}\right)V_i\right) + ... + w_{S_l}\left(\sum_{l=1}^{L}\left(\frac{1}{\sum_{l=1}^{L} N_l}\right)V_l\right)$$

Where:
S_R = Aggregate route score for given route R

[4] Complete Adjusted Route Score:

$$S_{R(A)} = m_R\left(w_{S_i}S_i + w_{S_j}S_j + w_{S_k}S_k + w_{S_l}S_l\right)$$

Expanded (1):

$$S_{R(A)} = \left(\frac{N_{\text{Tops}}}{\left(\frac{\text{Elev}_{\text{m}}}{1000(\text{Dist}_{\text{km}})} \right)} \right) \left(w_{S_i} \left(\sum_{i=1}^{I} w_i V_i \right) + ... + w_{S_l} \left(\sum_{l=1}^{L} w_l V_l \right) \right)$$

Expanded (2):

$$S_{R(A)} = \left(\frac{\sum_{i=1}^{I} N_i + \sum_{j=1}^{J} N_j + \sum_{k=1}^{K} N_k + \sum_{l=1}^{L} N_l}{\left(\frac{c_{ft}\text{Elev}_{\text{ft}}}{1000(c_{mi}\text{Dist}_{\text{mi}})} \right)} \right)$$

$$\bullet \left(w_{S_i} \left(\sum_{i=1}^{I} \left(\frac{1}{\sum_{i=1}^{I} N_i} \right) V_i \right) + ... + w_{S_l} \left(\sum_{l=1}^{L} \left(\frac{1}{\sum_{l=1}^{L} N_l} \right) V_l \right) \right)$$

Where:
$S_{R(A)}$ = Adjusted route score for given route R

In English, this equation reads: "number of tops (Munro, Munro Top, Corbett, Corbett Top) achieved on the route per vertical meter gained on the route, per horizontal meter gained on the route; scaled by the route type, route stages, terrain types, and terrain difficulty elements found on-route".

Given the time constraint of the project, some restrictions have been applied and notes made, as follows:

- Writing the score route function would be too-lengthy-a-process if calculating the score of each of the four route elements based on the individual values of the 5, 4, 11 and 20 sub-elements respectively using the factorial method, so the latter 3 are calculated simply by volume/variety. This does not accurately reflect the difficulty of terrain for example as a route with grass and concrete will be considered more difficult than a route with grade 3 slab, simply because it has one more element. In this context it doesn't matter too much as the math has been demonstrated successfully however.
- If using the exhaustive method, volume is accounted for by the inclusion and exclusion of values which, if all-inclusive, should add up to N sub-elements in that group. But this is rare.
- All four of the route elements do not include consistent examinations of their sub-elements (value ($V_{i,j,k,l}$)) when scoring a route. That is, for example using the route type array, sub-element $i = 1$ and $i = 5$ may not be examined together if

they do not appear as a combination in the JSON file and therefore don't critically need to be examined. Equally, in the route stage array, perhaps sub-element $j = 3$ isn't examined soley because it doesn't appear on its own in the JSON file. In a perfect world, every combination and isolation would be examined however, with the time allocated here, it is not worth the effort.

5.3.1.2 Describe Route Score

This is a small function which is used simply to evaluate the meaning of the given route score and translate it into more appropriate and understandable human terms. It's based on multiple increments from 0–100+, including: 0-33.$\dot{3}$: "Playground (Amateur)"; 34–66.$\dot{6}$: "Normie (Easy)"; 67–100: "Trad (Moderate)"; 100+: "GigaChad (Challenging)". It gives the user a slightly better indication of their what the route requires from their ability.

5.3.1.3 Search Route

This function searches JSON [1] (Figure A4) for data under a `landmass` → `route` for a match with input from a two-tier HTML `select` process. The first `select` allows the user to choose their desired landmass from their names matching those in the JSON file. Once an `option` has been selected, a relevant element is generated which displays the second `select` process that allows users to view and select from routes, matching those with names in the JSON file, on their selected landmass. This process could of course be executed using a manual text `input` process however, this relies on the user having pre-existing knowledge of landmasses and their routes and takes away from the comprehensiveness and inclusiveness of allowing them to view all possibilities. Essentially, this function outputs a body of text which fetches and displays the different attributes of routes in a human-readable context, including a mathematically computed unique and elegant 'route score' and the option to download the route's GPX file. This is the fasted method I have witnessed from landing on a site to being able to access a route, download it and use it.

5.3.1.4 Search Landmass

This function searches JSON [1] (Figure A4) for data under a `landmass` → `munro`, `munrotop`, `corbetttop` or `corbett` for a match with input from an HTML text `input` process. This input allows users to choose their desired hill which appears on a land-

mass; landmasses are looped until a Munro, Munro Top, Corbett or Corbett Top name appearing under a landmass is matched with the user's input. Note two things: search criteria does not have to be character-matching, case-matching or syntactically correct as there is JavaScript protocol in place to ensure more 'loose' data matching; and [2] if there are multiple matches under a user input, results are prioritized firstly by landmass appearance in the JSON file (as landmasses are ordered south-to-north and the majority of users will live south of the highlands, this will produce a more logical results), and secondly by hierarchy of peak (Munro, 'Top, Corbett, 'Top) according to their ordering in the file (which is also logical as users most often are searching for Munros). This function outputs a body of text which fetches and displays attributes of hills and the landmasses on which they appear, in a human-readable context. Also included is an image of the hill.

5.3.2 'Conditioning'

The 'Conditioning' portion of the home page of this site presents the user with the ability to [1] view 'Ability' attribute, and [2] view 'Equipment' attributes. This, like the 'Overview' section is a simplified and more informative version of what the user will see and use ability and equipment data for in the 'Conquest Map'.

5.3.2.1 Search 'Ability' & 'Equipment'

Users are presented with two initial separate HTML `select` protocol on a two-tier basis (four total `select` protocol) for each of the 'Ability' and 'Equipment' sections. The first allows them to choose an `option` based on which set of ability/equipment factors they wish to inspect, such as hill and route elements, summit features, route types and stages, terrain types and difficulties, etc.; and packs, technical equipment, footwear and clothing, etc. This generates the second `select` protocol which displays the individual factors under each heading. When the user selects a factor, the element is matched by a loop through the relevant JSON file for the seven ability headings or four equipment headings and a body of text is then generated summarizing the attributes. This includes all of the basics such as brief descriptions and images, etc. This feature is informative and serves no dynamic function until 'Conquest Map'.

5.3.3 'Weather'

Fetching and putting weather data to use only requires two parameters: latitude and longitude (`lat`, `lon`). This leaves huge opportunity to relate these parameters to many

different variables and therefore, many different human-interpreted scenarios. Accepted in the weather system are four different methods of acquiring input of latitude and longitude values. The first is two `select` options which list Munros, Corbetts and regions and loop the relevant JSON file for a match then relay their `lat` and `lon` attributes to input. The second is a search feature much like that seen in the 'Overview' section where the user can search loosely for the names of Munros and Corbetts and have their attributes captured from the JSON file. Thirdly, the user can use a Geolocation function to fetch their current coordinates and have them entered directly into the script. Finally, there is the option to manually input latitude and longitude directly into the `input` boxes. This is possible as the script takes its `lat` and `lon` inputs directly from the values of these input boxes which are displayed on the UI.

As seen in Figure A4, most data which is fetched using the OpenWeather API is displayed directly on the UI witch some form of string, symbol and CSS design aspects to contextualize them. However, there are some components such as [1] the day and time (daily and hourly), [2] sunrise time, [3] sunset time, [4] precipitation (daily and hourly) and [5] visibility which require some basic arithmetic and JavaScript conversions to be presentable. This extends no further than one line of code however. Further, there are several components which do not exist as part of the OpenWeather API and require more complex computations based on aspects of the API. These include [1] inspecting the complete range weather icon provided by OpenWeather and reassigning icon values to custom ones (daily and hourly); [2] assigning string descriptions of wind direction[4] at different angular intervals of wind direction provided by OpenWeather (daily and hourly)[5]; and [3] assigning an arrow symbol and computing a CSS angular rotation value relative to the wind direction (daily and hourly). This symbol is provided by the Version 5 FontAwesome Free Library (FontAwesome, 2021). Demonstrations of how these are implemented are highlighted in Table 5.3 below. The four basic computations can be seen in Figure A4. These computations also include various alike calculations relevant to CSS, for example, to color the background of the temperature section at different increments and change the font color accordingly, etc.

[4]The direction *from* which the wind is travelling (originates)

[5]Note there were no precise-enough sources supplying these intervals so much effort with a calculator was devoted

Weather Icons

```
let wicon = "";
for (var i in day.weather) {
  if (day.weather[i].icon === "01d") {
    wicon = "<i class='fas fa-sun'></i>";
  } else if (day.weather[i].icon === "01n") {
    wicon = "<i class='fas fa-moon'></i>";
  ...
  } else if (day.weather[i].icon === "50d") {
    wicon = "<i class='fas fa-smog'></i>";
  } else if (day.weather[i].icon === "50n") {
    wicon = "<i class='fas fa-smog'></i>";
  }
}
```

Wind Direction Description

```
let day_wind_dir = "";
if ((day.wind_deg) >= 348.76) {
  day_wind_dir = "N";
} else if ((day.wind_deg) >= 0 && (day.wind_deg) <= 11.25) {
  day_wind_dir = "N";
} else if ((day.wind_deg) >= 11.26 && (day.wind_deg) <= 33.75) {
  day_wind_dir = "N/NE";
} else if ((day.wind_deg) >= 33.76 && (day.wind_deg) <= 56.25) {
  day_wind_dir = "NE";
...
} else if ((day.wind_deg) >= 303.76 && (day.wind_deg) <= 326.25) {
  day_wind_dir = "NW";
} else if ((day.wind_deg) >= 326.26 && (day.wind_deg) <= 348.75) {
  day_wind_dir = "N/NW";
}
```

Wind Direction Symbol

$$\angle = (-45) + 180 + W$$

Where:
$\angle$ = Angle Assigned to CSS Rotate of Object (°) (no variable name as used directly in-line)
W = Wind Direction (°) = `hour.wind_deg` (as named in the OpenWeather API)
'−45' accounts for the 45° visual offset as standard design in the symbol (return to 0°)
'+180' accounts for the 'direction from which the wind is travelling' and rotates the symbol by 180° to point the nose of the arrow in the opposite direction which symbolizes the 'direction in which the wind is travelling'

Note that these examples are extracted from 'daily' weather forecasting. Near-identical code is replicated for the the 'hourly' equivalent.

Table 5.3: Weather System Computations

5.3.4 'Conquest' Map

The 'Conquest Map' is the heart of this system. It uses the Ordnance Survey Leisure Map API (OS API) (Ordnance Survey, 2022) to display a base map, along with other JavaScript features such as use of the Leaflet (Leaflet, 2022) library and others noted in the *System Construction* section, upon which many previously discussed, and further layers of, data is processed and displayed using various methods and forms of media.

5.3.4.1 Change Map Layer

A basic three-option `radio` input feature is available to the user as soon as they enter the page, giving them the option to fetch three different sets of base map layer data from the OS API: 'Leisure', 'Road', 'Outdoor'. These are the three most relevant topographical maps to the data displayed.

5.3.4.2 Show Current Location & Find Nearest

An array of two `button`s are presented which call functions that display the current location of the user's device and find the nearest route to the user, respectively. The first of these uses the Geolocation API to fetch the `lat` and `lon` attributes and inputs then into a function which outputs the user's location by [1] centering the map crosshair to the coordinates, [2] creating and displaying a marker upon this location, and [3] calculating and presenting an accuracy/tolerance radius around this. Finding the nearest project involves a similar process in that it fetches the user's coordinates using Geolocation and inputs them into a function which takes into account the coordinates of all routes in the relevant JSON file (Figure A4) and calculates their distance form the current coordinates. It then displays a route as standard; attributes of which, such as route markers, lines, hill markers and pop-ups etc., are discussed under the relevant sections 5.3.4.7 and 5.3.4.8.

5.3.4.3 Show & Hide Features

An array of four `button`s are presented which call functions that display markers for all Munros, Munro Tops, Corbetts, Corbett Tops, and hide all markers, respectively. These functions operate in a near-identical manner, in which they loop the relevant JSON file to fetch the `lat` and `lon` values of all the iterations of either `munro`, `munrotop`, `corbett`,

or `corbetttop`. The output is the creation and display of a Leaflet marker for every hill relative to the button selected, with a bound Leaflet pop-up which accepts inputs such as the name and different statistics and images of the hill from the JSON file, and is displayed upon click [of the marker]. The 'hide' function simply loops through the array used to store the markers upon creation and removes their relative layer from the map.

5.3.4.4 Filter Routes By 'Ability' & 'Equipment'

See section 5.3.2.1 for the breakdown of these components. After the user makes their relevant selections from the relevant HTML `select` protocol, instead of having all of the attributes of the associated matched items in the JSON file simply printed in a body of text, the selections are added to a pool. The values of this this pool (which are elements of ability and equipment) are then matched with routes under the landmasses of the appropriate JSON file. Once the user submits their request, all routes which either **do** or **do not** contain these attributes, based on the user preference through a `radio` input feature, are located in the appropriate JSON file. It then displays all of these routes as standard; attributes of which, such as route markers, lines, hill markers and pop-ups etc., are discussed under the relevant sections 5.3.4.7 and 5.3.4.8. Unlike when seeking an individual route, the map crosshair is not centered to any particular route as there may be more than one.

5.3.4.5 Score Route

See section 5.3.1.1 for the breakdown of this component. No attributes differ.

5.3.4.6 Describe Route Score

See section 5.3.1.2 for the breakdown of this component. No attributes differ.

5.3.4.7 Search Route

See section 5.3.1.3 for the breakdown of this component. There are additional features implemented in this function, in the relevance of the 'Conquest Map'. The user is presented now with four options instead of just the original 'summary' option under 'Overview'. They are: [1] present a summary of the route as detailed in section 5.3.1.3, [2] print the route, [3] hide all routes, and [4] begin a 'Galactic Conquest'.

The print route function essentially uses the same data from the JSON file as the summary function however, it uses the GPX suffix attribute of the selected route to match with an appropriate prefix. The file matching this path is then fetched and loaded by the function using the GPX to GeoJSON method, as this is a more appropriate use in JavaScript. This GPX data is then printed on the map. Secondly, the function calls another function discussed earlier which prints a route (Leaflet) marker. This marker contains a bound Leaflet pop-up which, upon click, displays variations of data displayed in the original route search output. Finally, the function also makes calls to other functions to which attributes names of hills appearing on the route are relayed and appropriate Leaflet markers are displayed. These functions behave similarly to those discussed in section 5.3.4.2, except this time they do not print all of the Munro, 'Top, Corbett and 'Top markers, they just print the relevant ones.

The hide all routes function acts much like the hide the function used to hide all hill markers. It simply loops through the array used to store the routes upon creation and removes their relative layer from the map. This includes all of the route markers, lines and hill markers. The fourth and final option, 'Galactic Conquest' is a meme which simply loops through the entire JSON file, collects every GPX attribute suffix, matches with the appropriate prefix, and prints them on the map with the appropriate route markers, lines and hill markers.

5.3.4.8 Search Landmass

See section 5.3.1.4 for the breakdown of this component. There are additional features implemented in this function, in the relevance of the 'Conquest Map'. As well as printing the body of text with the summary of all the attributes like before, this function also uses the `lat` and `lon` attributes of the hill firstly to input into another function which calculates the hill's distance from the user's current location and display this as part of the body-text output; and secondly, to center the map crosshair to the hill's location; and, calls a function which creates a single hill marker for the appropriate hill. This marker follows the same structure as those detailed in sections 5.3.4.4 and 5.3.4.7, except this time it prints only the single relevant Munro, 'Top, Corbett or 'Top.

6 System Construction

6.1 Development & Languages

6.1.1 Development Environment

6.1.2 Front-End & User Interface

6.1.3 Supporting Data (JSON)

6.2 Application Programming Interfaces & Libraries

Application Programming Interfaces (APIs) and JavaScript libraries are included in the development process of this system in order to use and build upon unique layers of existing web material and interface with them using various, in some cases, [free and open-source] JavaScript libraries. Libraries of course being additional pre-defined functions and methods etc., which can be used with the addition of various scripts and stylesheets in order to enable and initialize use of APIs.

6.2.1 Geolocation

6.2.2 Ordnance Survey

OS Road (1 : 250 000), OS Landranger (1 : 50 000), OS Explorer (1 : 25 000)

6.2.3 Open Weather

6.2.4 Leaflet

Leaflet JavaScript library for maps

6.2.5 Mapbox

Mapbox script for GPX-GeoJSON conversion

6.2.6 Chart.js

Chart.js JavaScript library for ranger

6.3 Project Management

6.3.1 Life-Cycle & Timing

6.3.2 Supporting Tools

7 Evaluation

7.1 Requirements

7.2 Design & Construction

7.2.1 Heuristic Evaluation

7.2.2 User Acceptance (UAT), Accessibility & Usability

7.2.3 'Prototyping' & Continuous Evaluation

7.3 Additional Testing

7.3.1 'Test-Driven' Development

7.3.2 Unit Testing

7.4 Constrictions

REVISE DATA AND DESIGN SECTIONS
Restricted to Glen Coe and Glen Etive
Some GPX files do not function as intended

7.5 Potential Enhancements

8 Development Conclusions

Table re-visiting requirements to check them off and answer any questions, like in last year's diss.

Appendices

Testing questionnaires, test cards (exp. result etc.), heuristic evaluation of system

Appendix 1: Foundational Material

Figure A1: Heuristic Evaluation – Walkhighalnds

Description	Vio.	Sev.	Proposed Solution
	Basic Features		
Sparse Navigation: There is not much order to the site navigation and in places thee is a lack pf relevance and hierarchy to the options. For example, some sub-sets of pages/features are listed in navigation on the same 'tier'; whether that being represented by the link having the same alignment as it's parent, or something of the sort	H_3, H_4	S_3	Correctly use HTML and CSS to display the hierarchy of pages and features in this system
Difficult Navigation: Some essential literature relating to components such as ability and equipment, as featured in this system, is difficult to find and requires a long path of navigation through pages and their children. Often some literature is simply linked through an in-body hyperlink (i.e. in a paragraph) and not even allocated it's own title etc.	H_3, H_4	S_3	Correctly use HTML and CSS to display the hierarchy of pages and features in this system. Ensure no relevance is lost in masses of text
	Map Features		
Feature Separation: The OS Maps displaying Munros and Corbetts etc. are displayed on different pages of the site, with navigation links between them. This means that interaction with features is limited to the point where all a user can do is view the information provided about it upon a click and follow further links from there. They cannot for example, toggle on/off different features while interacting with the map meaning isolating different features on a GPS route a user is viewing is impossible.	H_3, H_7	S_3	Implement a toggle feature for access to different map feature when the user desires, as listed in requirements
	Statistical Features		
...			
Total Violations: 3 Evaluator: Lewis Britton Platform(s): Brave Browser (Desktop), Brave Browser (Mobile)			

Table 8.4: Heuristic Evaluation – Wallhighlands

Figure A2: Heuristic Evaluation – Strava

Description	Vio.	Sev.	Proposed Solution
	Basic Features		
Weather Misuse: Weather data is only available as part of complete routes. I.e. activities which are complete and have been saved. Strava use a weather API so it is hard to understand why they would bot integrate this as a basic summary feature on the home page or on a profile etc. Historic weather data, i.e. that printed on complete routes, is the most irrelevant use of such as, come on, the user already knows what the weather was like on their routes and if they want to show off a 20hr hike in a snow storm or something, upload a picture	H_3	S_3	Include a comprehensive weather system, as listed in requirements
	Map Features		
Map Layers: Although the Mapbox map seen throughout Strava is used very effectively in hosting many dynamic layers such as customizable GPX routes with terrain' and athlete-specific attributes, it is just a basic map with contours, not a full topographical map. This significantly limits interaction with map features and interpretation as it simply doesn't show many of the relevant aspects to hiking	H_2, H_3	S_3	Utilize the OS Map API instead of a more basic one to ensure full access to topographical mapping, as listed in requirements
	Statistical Features		
...			
Total Violations: 2			
Evaluator: Lewis Britton			
Platform(s): Brave Browser (Desktop), Brave Brows er (Mobile)			

Table 8.5: Heuristic Evaluation – Strava

FIGURE A3: USER STORIES, PRIORITIZATION & SOLUTIONS

SCENARIO	REQ.	RELEVANCE	PRIORITY*	SOLUTION
		FUNCTIONAL – MUST-HAVE		
"I wish to be able to [use the current location of my device] in anticipation of [seeing my location printed on a map] which would allow me to [navigate towards map features relative to my location]"	1	UI OS Maps	1	Utilize *Geolocation* API to fetch location of user's device
"I wish to be able to [use the current location of my device] in anticipation of [weather forecast results for my location] which would allow me to [quick-access my weather forecast, likely while on-route]"	1, 5	UI Weather Service	1	Utilize *Geolocation* API to fetch location of user's device and place in weather function
"I wish to be able to [quickly view a summary of suggested routes] which would allow me to [see all the relevant attributes of a route on one return] and therefore, [select routes with less thought, time and effort]"	2	UI Route Summary	1	Include a section on the home page with this/these information/statistics
"I wish to be able to [include attributes related to my ability] which contribute to my [suggested routes] therefore, making them [more relevant to my needs]"	3	UI Ability Service	1	Include a section which accepts relevant user inputs and returns route suggestions as required
"I wish to be able to [include attributes related to the equipment I own] which contribute to my [suggested routes] therefore, making them [more relevant to my needs]"	4	UI Equipment Service	1	Include a section which accepts relevant user inputs and returns route suggestions as required
"I wish to be able to [seek locations by their name] in anticipation of the return of [weather forecast results related to their coordinates] which would allow me to [plan my routes more effectively]"	5	UI Weather Service	1	Include a section which accepts location-based inputs, whether this be pre-defined coordinates of locations or the user's current location as discussed previously, to display weather forecast results for these locations

"I wish to be able to [view an Ordance Survey map with OS Road (1 : 250 000), OS Landranger (1 : 50 000), and OS Explorer (1 : 25 000) layers simultaneously, alternative upon zoom] in anticipation of [viewing initially basic features such as my location as a symbol on the map and center-pan coordinates adjacent to the map] therefore, [making location seeking more interactive and understanable]"	1, 6	UI OS Maps	1	Utilize *Ordnance Survey* API to host the map Utilize *Leaflet* JavaScript library to initialize and interact with the map
"I wish to have the option to [interact with the OS map] which allows me to [toggle on/off display of features], allowing me to [view only specific features I wish to include on my routes]"	6	UI OS Map	1	Utilize *Leaflet* JavaScript library to display and toggle on/off symbols representing the location of various geographical features such as Munros, Munro Tops, Corbetts, and Corbett Tops
"I wish to have the option to [interact with the OS map] which allows me to [toggle on/off GPS routes based on search criteria] therefore, [making route results relevant to my subjective choice], as opposed to other factors such as recommendations based on my ability and equipment, or raw search"	6	UI OS Map	1	Utilize *Leaflet* JavaScript library to display and toggle on/off GPS route (GPX data) layers over the map, relative to various search criteria Utilize *Mapbox GL-JS* JavaScript library to convert GeoJSON files to GPX
"I wish to be able to [search for routes close to my current location] which returns the [closest relevant GPS overlay] to my location, making it easy for me to [make a fast decision if I just want a close project]"	6, 1	UI OS Map	1	Utilize *Geolocation* API to fetch location of user's device and compare to the location of all listed projects
FUNCTIONAL – SHOULD-HAVE				
"I wish to be able to [interact with the OS Map] in order to [change its layers] which will allow me to [interpret the features and contours of the land in different ways]"	1	UI OS Map	2	Include some form of input which *Leaflet* JavaScript library to allow the user to change primary map layer

"I wish to [view grid markers at the center of the OS Map] which, when panning, will [pinpoint the center coordinates displayed on the map] therefore, [making it easy to view, locate and pan to and from features and locations etc.]"	2	UI Map OS	2	Include basic HTML and CSS which displays reticle
		Functional – Could-Have		
"I wish to [have my data stored] in the system so I can [access it down the line] to [use it in various site features]"	1	Back-End	3	Implement a back-end database system to store and interact with user data
"With regards to [data access], I wish to be [assured of appropriate security measures] implemented by the system controllers which will give me [peace of mind] regarding data protection	2	Back-End	3	Implement appropriate authentication and data protection measures in the system"
"I wish to be able to [use and reuse attributes related to my account] in order to [tailor results to my own abilities and equipment], for example; allowing me to [gain a more accurate understanding of what I'm able to do] with less effort"	3	UI Back-End	3	Allow already-implemented features such as user ability and equipment services to access and use stored user data, as opposed to using basic inputs
"I wish to be able to [choose 'priority attributes'] relevant to data stored upon my previous routes which returns [more relevant route recommendations] to historic ones, relevant to what I have and haven't done based on my preference; allowing me to [more dynamically choose routes]"	4	UI Back-end	3	Implement appropriate measures
		Functional – Won't Have		
"I wish to use a GPS device such as a watch or cellular device to [record GPX data] which can be used to [generate route information] and [display useful route records] which I can interpret and benefit/learn from"	1	Back-End	4	(Will not) implement functionality to accept GPX uploads and transform relevant components into human readable information
		Non-Functional		

"I wish to be able to [use the site on any browser] therefore meaning I can [view all content] regardless of what browser platform I use, meaning I'll be able to [consistently interact with the functionality]"	1	UI Browser Support	5	Ensure all languages, packages, libraries, etc., are using the most versatile syntax and have aggregate browser support
"I wish to be able to [us ethe site on any device platform] therefore meaning I can [view all content] regardless of what device I use, meaning I'll be able to [consistently interact with the functionality]"	2	UI	5	Implement the correct CSS protocol to allow the site do dynamically scale to various device sizes; using appropriate adjustment and hiding/showing of content
"I wish to [view additional] literature based upon [information regarding topics I wish to explore]; [enhancing my knowledge of the components of the sport and my decision making]"	3	UI Research	5	Implement appropriate bodies of text relevant to areas upon which they may be if use to the user
"I wish to [view elements of graphic design] which [show relevance to the areas in which they appear] therefore, stimulating, engaging and immersing me further in the sport]"	4	UI	5	Use CSS to implement the correct elements and principals of graphic design, primarily color schemes and shape/geometry, which are relevant to the outdoor context Use graphic design tools (such as Serif PagePlus and GIMP) to create and implement relevant features such as logos, images and renders, which are relevant to the outdoor context
"I wish to be able to optimally interact with the site when it comes to navigation and viewing experience, regardless of any impairity or things of the sort"	5	UI Research	5	Implement appropriate design protocol to make this experience as easy as possible

*: 'Must-Have' = 1; 'Should-Have' = 2; 'Could-Have' = 3; Won't Have = 4; Non-Functional = 5

Various Sources (2022)

FIGURE A4: DATA SOURCES

Data/Type	Purpose/Desc.	Location
	Location (Geolocation API)	
latitude Number	Provide geographical data relative to the user's machine	Geolocation API
longitude Number	Provide geographical data relative to the user's machine	Geolocation API
	Regional (Research[1])	
county List (of Dict.)	Store data on each county in Scotland	JSON [1]
name $\in$ county String	Provide name of each county in Scotland	JSON [1]
lat $\in$ county Number	Provide latitude of each county in Scotland	JSON [1]
lon $\in$ county Number	Provide longitude of each county in Scotland	JSON [1]
region List (of Dict.)	Store data on each region in Scotland	JSON [1]
name $\in$ region String	Provide name of each region in Scotland	JSON [1]
subregion $\in$ region List (of Dict.)	Store data on each sub-region in Scotland	JSON [1]
name $\in$ subregion $\subset$ region String	Provide name of each subregion in Scotland	JSON [1]
subsubregion $\in$ subregion $\subset$ region List (of Dict.)	Store data on each sub-sub-region in Scotland	JSON [1]
name $\in$ subsubregion $\subset$ subregion $\subset$ region String	Provide name of each sub-sub-region in Scotland	JSON [1]
lat $\in$ subsubregion $\subset$ subregion $\subset$ region Number	Provide latitude of each sub-sub-region in Scotland	JSON [1]
lon $\in$ subsubregion $\subset$ subregion $\subset$ region Number	Provide longitude of each sub-sub-region in Scotland	JSON [1]
	Landmass (Research)	
landmass List (of Dict.)	Store data on each landmass in Scotland, within the sample size	JSON [1]
name $\in$ landmass String	Provide name of each landmass in Scotland, within the sample size	JSON [1]
type $\in$ landmass String	Provide type of each landmass in Scotland, within the sample size (see Home Page $\rightarrow$ Overview for possible values)	JSON [1]
subtype $\in$ landmass String	Provide sub-type of each landmass in Scotland, within the sample size (see Home Page $\rightarrow$ Overview for possible values)	JSON [1]

subsubtype ∈ landmass String	Provide sub-sub-type of each landmass in Scotland, within the sample size	JSON [1]
munro, munrotop, corbett, corbetttop [2] ∈ landmass List (of Dict.)	Store data on each Munro, Munro Top, Corbett and Corbett Top, respectively, present on each landmass in Scotland, within the sample size	JSON [1]
name ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass String	Provide name of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
lat ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass Number	Provide latitude of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
lon ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass Number	Provide longitude of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
OSgrid ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass String	Provide OS Grid Coordinates of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
elevation ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass Number	Provide elevation (ft) of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
prominence ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass Number	Provide prominence (ft) of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
isolation ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass Number	Provide isolation (ft) of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
summit ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass String	Provide name/type of summit feature of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
image ∈ {munro ∨ munrotop ∨ corbett ∨ corbetttop} ⊂ landmass String	Provide suffix of the image file path of each Munro, Munro Top, Corbett and Corbett Top present on each landmass in Scotland, within the sample size	JSON [1]
corrie [3] ∈ landmass List (of Dict.)	Store data on each Corrie present on each landmass in Scotland, within the sample size	JSON [1]
name ∈ corrie ⊂ landmass String	Provide name of each Corrie present on each landmass in Scotland, within the sample size	JSON [1]
gully ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more gullies, within the sample size	JSON [1]
lochain ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more lochains, within the sample size	JSON [1]
waterfall ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more waterfalls, within the sample size	JSON [1]

peatbog ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more peat bogs or peat hag fields, within the sample size	JSON [1]
mudbog ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more (usually unnatural) mud bogs, within the sample size	JSON [1]
boulderfield ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more boulder fields, within the sample size	JSON [1]
scree ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more scree slopes or deposits, within the sample size	JSON [1]
shoulder ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more shoulders, within the sample size	JSON [1]
arete ∈ landmass Boolean	Provide true/false value based on whether each landmass in Scotland has one or more arêtes, within the sample size	JSON [1]
humanfeatures ∈ landmass List (of Dict.)	Store data on each human feature present on each landmass in Scotland, within the sample size	JSON [1]
name ∈ humanfeatures ⊂ landmass String	Provide name of each human feature present on each landmass in Scotland, within the sample size	JSON [1]
type ∈ humanfeatures ⊂ landmass String	Provide type of each human feature present on each landmass in Scotland, within the sample size	JSON [1]
parentlandmass ∈ landmass String	Provide name of the parent landmass of each landmass in Scotland, within the sample size	JSON [1]
parentpeak ∈ landmass String	Provide name of the parent peak of each landmass in Scotland, within the sample size	JSON [1]
region ∈ landmass String	Provide name of the region in which each landmass in Scotland is located, within the sample size	JSON [1]
subregion ∈ landmass String	Provide name of the sub-region in which each landmass in Scotland is located, within the sample size	JSON [1]
informalregion ∈ landmass String	Provide name of the informal region in which each landmass in Scotland is located, within the sample size	JSON [1]
	Route (Research)	
route ∈ landmass List (of Dict.)	Store data on each present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
name ∈ route ⊂ landmass String	Provide name of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
distance ∈ route ⊂ landmass Number	Provide distance (mi) of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
elevationgain ∈ route ⊂ landmass Number	Provide total elevation gain (ft) of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]

stdtime ∈ route ⊂ landmass Number	Provide standard time (hrs) taken by a normie to complete each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
type ∈ route ⊂ landmass List (of Strings)	Provide array of descriptions of the types/purposes of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
stage ∈ route ⊂ landmass List (of Strings)	Provide array of descriptions of the stages of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
terraintype ∈ route ⊂ landmass List (of Strings)	Provide array of descriptions of the terrain types of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
terraindiff ∈ route ⊂ landmass List (of Strings)	Provide array of descriptions of the terrain difficulties of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
gear ∈ route ⊂ landmass List (of Strings)	Provide array of descriptions of the recommended gear required for each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
munro ∈ route ⊂ landmass List (of Strings)	Provide array of the names of Munros on each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
munrotop ∈ route ⊂ landmass List (of Strings)	Provide array of the names of Munro Tops on each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
corbett ∈ route ⊂ landmass List (of Strings)	Provide array of the names of Corbetts on each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
corbetttop ∈ route ⊂ landmass List (of Strings)	Provide array of the names of Corbett Tops on each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
GPX ∈ route ⊂ landmass String	Provide suffix of the GPX file path of each route present on each landmass in Scotland, within the sample size and the particular routes selected	JSON [1]
	Ability (Research)	
elementshill List (of Dict.)	Store data on each attribute of a hill (see Home Page → Overview for possible values)	JSON [2]
elementsroute List (of Dict.)	Store data on each attribute of a route (see Home Page → Overview for possible values)	JSON [2]
summitfeats List (of Dict.)	Store data on each type of summit feature (see Home Page → Overview for possible values)	JSON [2]
type List (of Dict.)	Store data on each type of traverse/sport you can do/practice on a hill its routes (see Home Page → Overview for possible values)	JSON [2]
stage List (of Dict.)	Store data on each of the traverse stages included on a hill and its routes (see Home Page → Overview for possible values)	JSON [2]

terraintype List (of Dict.)	Store data on each type of terrain on a hill and its routes (see Home Page → Overview for possible values)	JSON [2]
terraindiff List (of Dict.)	Store data on each relative terrain difficulty factor of a hill and its routes (see Home Page → Overview for possible values)	JSON [2]
name[4] ∈ {elementshill ∨ elementsroute ∨ summitfeats ∨ type ∨ stage ∨ terraintype ∨ terraindiff} String	Provide name of each attribute of a hill, attribute of a route, type of summit feature, type of traverse/sport you can do/practice, type of traverse stages, type of terrain, relative terrain difficulty factor	JSON [2]
desc[5] ∈ {elementshill ∨ elementsroute ∨ type ∨ stage ∨ terraintype} String	Provide description of each attribute of a hill, attribute of a route, type of traverse/sport you can do/practice, type of traverse stages, type of terrain	JSON [2]
image[6] ∈ {summitfeats ∨ type ∨ stage ∨ terraintype ∨ terraindiff} String	Provide suffix of the image file path of each type of summit feature, type of traverse/sport you can do/practice, type of traverse stages, type of terrain, relative terrain difficulty factor	JSON [2]
Equipment (Research)		
packs List (of Dict.)	Store data on each type of pack, travel equipment and their accessories, relevant to outdoor activity (see Home Page → Overview for possible values)	JSON [2]
technical List (of Dict.)	Store data on each type of technical equipment and their accessories, relevant to outdoor activity (see Home Page → Overview for possible values)	JSON [2]
shoes List (of Dict.)	Store data on each type of footwear and their accessories, relevant to outdoor activity (see Home Page → Overview for possible values)	JSON [2]
clothing List (of Dict.)	Store data on each type of clothing and their accessories, relevant to outdoor activity (see Home Page → Overview for possible values)	JSON [2]
name[7] ∈ {packs ∨ technical ∨ shoes ∨ clothing} String	Provide name of each pack, technical component, shoe, item of clothing	JSON [2]
desc[8] ∈ {packs ∨ technical ∨ shoes} String	Provide description of each pack, technical component, shoe	JSON [2]
comp[9] ∈ {shoes} List	Provide array of compatibility options for each shoe (boots and crampons, etc.)	JSON [2]
feat[9] ∈ {shoes} List	Provide array of features available with each shoe	JSON [2]
adv[9] ∈ {shoes} List	Provide array of advantages of each shoe, item of clothing	JSON [2]
dangers[9] ∈ {shoes} List	Provide array of dangers of each shoe	JSON [2]
image[10] ∈ {packs ∨ technical ∨ shoes ∨ clothing} String	Provide suffix of the image file path of each pack, technical component, shoe, item of clothing	JSON [2]
Weather (OpenWeather API)		

dt – Day ∈ daily Number	Provide date and time of weather occurrence for the day	Computed from OpenWeather One Call API 1.0, as JavaScript date conversion of: 1000(day.dt)
icon – Day ∈ {weather ⊂ daily} String	Provide a graphical icon representing the average weather condition for the day	Computed from OpenWeather One Call API 1.0, associating existing symbols with custom ones
main – Day ∈ {weather ⊂ daily} String	Provide a word or phrase which summarises the average weather condition for the day	OpenWeather One Call API 1.0
description – Day ∈ {weather ⊂ daily} String	Provide a phrase or short sentence which summarises the average weather condition for the day	OpenWeather One Call API 1.0
max – Day ∈ {temp ⊂ daily} Number	Provide the value of the maximum forecasted temperature for the day	OpenWeather One Call API 1.0
min – Day ∈ {temp ⊂ daily} Number	Provide the value of the minimum forecasted temperature for the day	OpenWeather One Call API 1.0
day – Day ∈ {feels_like ⊂ daily} Number	Provide the value of the daytime 'feels like' temperature for the day	OpenWeather One Call API 1.0
night – Day ∈ {feels_like ⊂ daily} Number	Provide the value of the nighttime 'feels like' temperature for the day	OpenWeather One Call API 1.0
pop – Day ∈ daily Number	Provide a probability value between 0 and 1 which represents the chance of precipitation for the day	Computed from OpenWeather One Call API 1.0, as: 100(day.pop)
wind_deg – Day ∈ daily Number	Provide a value representing the average angle from which the wind is travelling for the day	OpenWeather One Call API 1.0
wind_deg (for Description) – Day ∈ daily Number	Provide a value representing the average angle from which the wind is travelling for the day, which can be compared to a defined group of angular ranges to determine a description	Computed from OpenWeather One Call API 1.0, based on angular ranges
wind_deg (for Symbol) – Day ∈ daily Number	Provide a value representing the average angle from which the wind is travelling for the day, which can be compared to a defined group of angular ranges to determine a suitable symbol	Computed from OpenWeather One Call API 1.0, associating a symbol with a value representing the relative rotation of the symbol
wind_speed – Day ∈ daily Number	Provide the value of the average wind speed for the day	OpenWeather One Call API 1.0
pressure – Day ∈ daily Number	Provide the value of the average air pressure for the day	OpenWeather One Call API 1.0
humidity – Day ∈ daily Number	Provide the value of the average humidity for the day	OpenWeather One Call API 1.0
dew_point – Day ∈ daily Number	Provide the value of the average dew point for the day	OpenWeather One Call API 1.0
uvi – Day ∈ daily Number	Provide the value from an index representing the average ultraviolet radiation emitted for the day	OpenWeather One Call API 1.0

sunrise – Day ∈ daily Number	Provide time of sunrise for the day	Computed from OpenWeather One Call API 1.0, as JavaScript date conversion of: 1000(`day.sunrise`)
sunset – Day ∈ daily Number	Provide time of sunset for the day	Computed from OpenWeather One Call API 1.0, as JavaScript date conversion of: 1000(`day.sunset`)
dt – Hour ∈ hourly Number	Provide date and time of weather occurrence for the hour	Computed from OpenWeather One Call API 1.0, as JavaScript date conversion of: 1000(`hour.dt`)
icon – Hour ∈ {weather ⊂ hourly} String	Provide a graphical icon representing the average weather condition for the hour	Computed from OpenWeather One Call API 1.0, associating existing symbols with custom ones
temp – Hour ∈ hourly Number	Provide the value of the average forecasted temperature for the hour	OpenWeather One Call API 1.0
feels_like – Hour ∈ hourly Number	Provide the value of the average 'feels like' temperature for the hour	OpenWeather One Call API 1.0
pop – Hour ∈ hourly Number	Provide a probability value between 0 and 1 which represents the chance of precipitation for the hour	Computed from OpenWeather One Call API 1.0, as: 100(`hour.pop`)
wind_deg – Hour ∈ hourly Number	Provide a value representing the average angle from which the wind is travelling for the hour	OpenWeather One Call API 1.0
wind_deg (for Description) – Hour ∈ hourly Number	Provide a value representing the average angle from which the wind is travelling for the hour, which can be compared to a defined group of angular ranges to determine a description	Computed from OpenWeather One Call API 1.0, based on angular ranges
wind_deg (for Symbol) – Hour ∈ hourly Number	Provide a value representing the average angle from which the wind is travelling for the hour, which can be compared to a defined group of angular ranges to determine a suitable symbol	Computed from OpenWeather One Call API 1.0, associating a symbol with a value representing the relative rotation of the symbol
wind_speed – Hour ∈ hourly Number	Provide the value of the average wind speed for the hour	OpenWeather One Call API 1.0
wind_gust – Hour ∈ hourly Number	Provide the value of the average wind 'gust' speed for the hour	OpenWeather One Call API 1.0
pressure – Hour ∈ daily Number	Provide the value of the average air pressure for the hour	OpenWeather One Call API 1.0
humidity – Hour ∈ daily Number	Provide the value of the average humidity for the hour	OpenWeather One Call API 1.0
dew_point – Hour ∈ daily Number	Provide the value of the average dew point for the hour	OpenWeather One Call API 1.0
visibility – Hour ∈ hourly Number	Provide the value from an index representing the average visibility for the hour	OpenWeather One Call API 1.0
uvi – Hour ∈ hourly Number	Provide the value from an index representing the average ultraviolet radiation emitted for the hour	OpenWeather One Call API 1.0

[1]: Any component labelled 'research' refers to my personal iterations through Ordnance Survey maps and in-person field trips to collect geographical information.
[2]: Munro, Munro Top, Corbett, Corbett Top variables are each their own lists containing different values however, the variables for each are identical and therefore are combined to save space. The only difference is Munros and Corbetts are included in weather location search, whereas Munro Tops and Corbett Tops are not as, well, who wants to know the weather on a Top?!
[3]: As this list of dictionaries just contains single-type attributes (the names of the corries, it could simply be a list of strings however, it remains this way in anticipation of the subsequent addition of Corrie latitude and longitude data.)
[4]: Each elementshill, elementsroute, summitfeats, type, stage, terraintype, terraindiff variables have their own name variable however, are included comprehensively here.
[5]: Same situation as footnote 4 however, summitfeats and terraindiff do not include description variables
[6]: Same situation as footnotes 4 and 5 however, elementshill and elementsroute do not include image variables
[7]: Each packs, technical, shoes, clothing variables have their own name variable however, are included comprehensively here.
[8]: Same situation as footnote 7 however, clothing does not include description variables
[9]: Same situation as footnotes 7 and 8 however, packs, technical, clothing do not include compatibility, feature, advantages, dangers variables
[10]: Same situation as footnotes 7, 8 and 9

Figure A5: Promotional Poster

Figure A6: Functions Summary

Function	Inputs	Data	Return Type	Output
		'Overview'		
		Score Route		
...				
		Describe Route Score		
...				
		Search Route		
...				
		Search Landmass		
...				
		'Conditioning'		
		Search 'Ability'		
...				
		Search 'Equipment'		
...				
		'Weather'		
		'Conquest Map'		
		Change Map Layer		
...				
		Show Current Location & Find Nearest		
...				
		Show & Hide Features		
...				

Filter Routes by 'Ability'
...
Filter Routes by 'Equipment'
...
Score Route
...
Describe Route Score
...
Search Route
...
Search Landmass
...

Table 8.8: Functions Summary

Appendix 2: Foundational Material

Author's Note

The following content is manufactured by myself in aid of basic understanding of the background to contexts, data and methods present within this study. It is presented as teaching material, in a format I would output if in such position.

Figure B1: Linguistics of 'LaTeX'

LaTeX (or LaTeX, even latex (Donald E. Knuth's more recent installment of TeX)) is usually pronounced /laːtɛk/ ('lah') or /leɪtɛk/ ('lei'/'lay') in English (that is, not with the /ks/ pronunciation English speakers normally associate with X, but with a /k/). The characters T, E, X in the name come from capital Greek letters tau, epsilon, and chi, as the name of TeX derives from the Greek: $\tau\varepsilon\chi\nu\eta$ (skill, art, technique, precision); for this reason, Donald E. Knuth promotes a pronunciation of /tɛk/ (tekh) (that is, with a voiceless velar fricative as in Modern Greek, similar to the last sound of the German word "Bach", the Spanish "j" sound, or as "ch" in a Scottish 'loch').

Figure B2: Don. Knuth's Computer Modern Unicode (CMU) Font Family

Figure B3: Why My Pre-Title's Right and You're Wrong

I have received numerous comments which anyone would regard naïve and under-educated regarding my pre-title of this study: *AG436 Dissertation Coursework Assignment.* The

Serif	Sans Serif	Monospaced
CMU Serif Roman	CMU Sans Serif	CMU Concrete
CMU Serif Bold	**CMU Sans Serif Bold**	
CMU Serif Italic		*CMU Concrete Italic*
CMU Serif Oblique	*CMU Sans Serif Oblique*	*CMU Concrete Oblique*
CMU Serif Small Caps		CMU Concrete Small Caps

In the presence of traditionalists, a suitable alternative to Donald E. Knuth's Computer Modern Unicode font family may be considered: Andale Mono.

argument originates in the 'Coursework Assignment' portion. People argue that a dissertation 'is not'/'does not have' an assignment. Not only is this poor characteristic recognition, it is semantically wrong. *AG436: Dissertation* is a class just like any other. However under this class, there are no lectures, no tutorials and therefore no exams as there is no [taught] content. Do not confuse this with the class 'having no content' though. AG436's content is apparent through literature of the student's choice. Therefore, it is possible for a 'coursework assignment' to be based on this. Hence, any further comments are null.

Figure B4: Lorn and Lorne

Ranking of Lorne from House of Bruar to Malcolm Allan.

...

Bibliography

[1] Nielsen, J. (1994). *Heuristic Evaluation.* Usability Inspection Methods, John Wiley & Sons

[2] Walkhighlands. (2022). *Walkhighlands: Scotland walks and accommodation* Available At: `https://www.walkhighlands.co.uk/` (Accessed: 22/03/2022).

[3] MDN Web Docs. (2022). *Geolocation API - MDN Web Docs* Available At: `https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API` (Accessed: 08/04/2022).

[4] Peakbagger.com (2022). *Search - Peakbagger.com* Available At: `https://www.peakbagger.com/Search.aspx` (Accessed: 03/04/2022).

[5] Harold Street. (2022). *Mountain GPS Waypoints & Hill Bagging Lists* Available At: `https://www.haroldstreet.org.uk/waypoints/` (Accessed: 03/04/2022).

[6] OpenWeather. (2022). *One Call API: weather data for any geographical coordinate - OpenWeatherMap* Available At: `https://openweathermap.org/api/one-call-api` (Accessed: 03/04/2022).

[7] Leaflet. (2022). Available At: `https://leafletjs.com/reference-1.7.1.html` (Accessed: 03/04/2022).

[8] FontAwesome. (2021). Available At: `https://fontawesome.com/v5/search` (Accessed: 08/04/2022).

[9] Ordnance Survey. (2022). Available At: `https://labs.os.uk/public/os-data-hub-examples/os-name` (Accessed: 03/04/2022).

[10] Chart.js. (2022). Available At: `https://www.chartjs.org/docs/latest/` (Accessed: 03/04/2022).