HERIOT WATT UNIVERSITY

School of Mathematical and Computer Sciences

Trail.

A Mobile App for Learning about Trees on HWU Edinburgh Campus using Spatial Triggering, Speech Synthesis, and QR Technologies.

by Ruaridh Mollica

Supervisor: Dr Phil Bartie

Second Reader: Prof. Oliver Lemon

Deliverable 2: Final Year Dissertation

BSc (Hons) Computer Science

Declaration

I, Ruaridh Mollica confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: 

Date: 20th of April 2021

Acknowledgments

I would like to begin by taking a moment to express my gratitude to the following people for their continued assistance and support during the project.

Firstly, I would like to thank my supervisor **Dr Phil Bartie** for his kindness and help throughout the process. His expertise in the area of spatial technology and his genuine interest in the project has been paramount in the success of the Trail application.

I would also like to thank **Professor Oliver Lemon** for his valuable feedback on the first deliverable, without his insight I would not have pushed to experiment with more novel technologies, and hence, would not have had as much fun!

Finally, I would like to thank **Robbie Fraser,** Operations Manager of landscape, recycling, and waste for proposing the idea, and spending time with me on campus to create the new trail and teach me about all of the beautiful trees.

Abstract

Field Trips in the educational system provide a key experience for students, to not only have a break from the classroom but also solidify material in their mind via real-world, interactive situations. Applications of Virtual Guides have seen success in the tourism industry using trail-like map interfaces, however, these tools lack the technologies to allow for complete device cross-compatibility and support with current QR/AR systems and spatial triggering. The demand for virtual tours has developed immensely since the start of the Covid-19 pandemic, with the term obtaining a 7-fold increase in online searches between February and March of 2020 alone.

This research document details the implementation of Trail - a Virtual Tour Guide developed as a Progressive Web Application(PWA), giving users the ability to learn about the trees located around the Heriot-Watt University Campus in Edinburgh.

Literature related to technologies such as QR, NFC, spatial triggering, speech synthesis, augmented reality, and haptic feedback was analysed in detail, along with the study of similar solutions. This analysis paved the way for the development of the application and found that NFC would not be compatible due to it being an upcoming technology with very little support in web app environments.

A system evaluation, along with a usability study involving 13 participants was carried out and analysed. The results found that the Trail application proves to be an effective tool in allowing a user to learn about trees on the Heriot-Watt campus via a wide range of technologies.

There were features of the system that demonstrated a variety of limitations, such as the behaviour of augmented reality being extremely volatile, issues arising when traversing overlapping geofences, or users struggling with the installation of the PWA. These limitations, along with many others, are identified and solutions are discussed - future work and extensions to the system are also proposed.

Table of Contents

Chapter 1: Introduction

This project stemmed from an idea proposed by Robbie Fraser, the Operations Manager for landscape, recycling and waste at Heriot-Watt University (HWU). This idea harbouring the vision of a mobile guide system that informs users about trees on the Heriot-Watt campus. It is worth noting that two versions of this application will be developed - the version discussed in this thesis, as well as a simplified version with less features that will be installed on the university campus for use by the general public – hopefully for years to come.

Physical field trips in the educational system enable students to solidify curriculum material in their mind through real-world experiences (Zanetis, 2010). With current Covid-19 guidelines imposing restrictions on travel and gatherings, the term “virtual tour” has seen a 7-fold increase in 2020 (Bloom, 2020). Existing applications also lack the technologies to support current QR/AR systems and spatial triggering. With these factors in play, there is an unprecedented desire for these virtual field trips and mobile guides.

This research document details the implementation of Trail - a Virtual Tour Guide developed as a Progressive Web Application(PWA), giving users the ability to learn about the trees located around the Heriot-Watt University Campus in Edinburgh.

Literature related to technologies such as QR, NFC, spatial triggering, speech synthesis, augmented reality, and haptic feedback is analysed in detail, along with the study of similar solutions. A system evaluation study is also carried out to gain valuable insights from users. These insights are then used to guide a discussion of achievements, limitations, and propose future work.

The following link is a video demo showcasing some of the Trail application’s main features in action, and may provide some useful context: https://www.youtube.com/watch?v=4opBHP18aIk

1.1 Aims

The aim of this project is to develop a dependable and robust mobile application that allows a user to get information about specific trees around the Heriot-Watt Campus in Edinburgh. The application will provide an interactive tour experience via the use of QR, spatial, and speech synthesis technologies and act as a go-to tool for tree exploration. Users should feel encouraged to appreciate their surrounding natural environments and learn about the benefits of ecosystems as a whole.

1.2 Objectives

- Investigate the demand in today's society for tour guide applications like Trail.
- Research related existing applications and solutions detailing any issues identified.
- Research different methods of event triggering with emphasis on current technologies, compare and contrast to deduce which are implementable options.
- Research the most effective methods of information delivery to present tree data to a user.
- Compare and contrast methods of sourcing and generating map data appropriate for use in an interactive manner.
- Research how user location data can be sourced and identify any issues that could arise.
- Develop a usable, effective tour guide application to inform users of the trees around Heriot-Watt Campus:
    - This will be a Progressive Web Application (PWA) which allows a user to view statistics on the general tree ecosystem at HWU, navigate a virtual map to view tree information from anywhere, undertake a physical tour of trees around HWU with the use of QR, spatial, and speech synthesis technologies.
    - The application will be robust and efficient, to ensure this testing will take place during development.
    - An array of functional and non-functional requirements will be created and referred to meticulously throughout development to ensure completeness of the application.
- Implement physical hardware on HWU Campus to assist the application tour such as QR codes attached to trees and a sign detailing the beginning of a tour.
- Ensure the web application has optimal performance to meet the PWA specification that allows it to run "natively" on a user's device when saved to their home screen.
- Conduct an in-depth usability study to evaluate the developed application and gain insightful test data and feedback.
- Analyse evaluation study results along with the successful implementation of requirements to conclude the effectiveness of the Trail application and propose future work based on this analysis.

Chapter 2: Background

2.1 Introduction

The terms Virtual Field Guide (VFG) and Virtual Field Trip (VFT) are used synonymously through literature and their definitions vary (Litherland, 2012). VFTs aim to virtualize the real-world environment of a specific location without the constraints of being physically present (Cliffe, 2017). For the duration of this research document VFGs will describe a Virtual Field Trip that is less focused on complete virtual immersion, and more on assistance of a user physically undertaking a tour. The term Virtual Field Application (VFA) will be used as an umbrella term encapsulating both VFGs and VFTs.

VFAs provide a user with the ability to quickly discover relevant, location specific information via a combined range of technologies. This section will examine the foundations of a VFA including an overview of the effectiveness of these applications, existing solutions, event triggering and data delivery methods, suitable positioning technologies, user interfaces including haptic feedback, and mobile app development approaches (e.g., PWA).

2.2 The Effectiveness and Demand of Virtual Field Applications

This section identifies the demand for, and effectiveness of Virtual Field Applications in different contexts (e.g., educational).

2.2.1 The Educational System

Field Trips in the educational system provide a key experience for students, not only to have a break from the classroom but also to solidify any curriculum material in their mind via real-world/interactive situations. Over the last decade there has been a steady decline in the number of physical field trips that have been taking place, mainly due to increases in travel costs (Zanetis, 2010).
Zanetis argues that the median of a virtual tour is in itself a powerful experience that can engage and enchant students via its ability to allow you to interact with people who are far away, as well as allowing students to explore materials covered in class and discover the wonders of the world without travelling across the globe to do so.

Hence, virtual field trips have been regarded as the logical next step in combatting these issues and providing a wide variety of benefits to students and teachers alike. Steven Fletcher (2002), author of “Fieldwork Education and Technology: A GEES Perspective” reinforces the benefits of virtual

environments and guides stating; e-learning resources have definitively shown to assist students in becoming much less passive learners by engaging their multi- sensory learning via interactive media. These virtual field trips also let students reflect and evaluate on their experiences in a much more in-depth way, with the option to do the tour an unlimited number of times (Dykes, 2000). They remove the aspect of uncertainty and cancellation in lot of situations where weather could prevent some aspects of a field trip or even the whole experience; time constraints may mean a trip cannot be attended, or perhaps the mobility of a student prevents them from being able to participate (Stott T., 2014).

However, the literature is not unanimous on the benefits of virtual field trips with Anthony David Cliffe (2017) writing in "*A review of the benefits and drawbacks to virtual field guides in today's Geoscience higher education environment*" that a study showed students disagreed that virtual field trips helped them learn more and they should not replace physical field trips. The challenge that they have being they cannot replace the innate nature of field trips which is to be outside and explore the world (Bellan J. M., 1998).

2.2.2 Covid-19 applications

Covid-19 has imposed many restrictions on society including social distancing, curfews, and limiting the number of people you can be with in a group (Scottish Government, 2020).
These restrictions have created an unprecedented hurdle in the way organizations can hold events and activities. Many organizations including Heriot-Watt University have adapted to using virtual systems to combat the difficulties imposed. Physical campus tours have been replaced with virtual tours, with an aim to stay within the restrictions but still provide new students with the experience they need to begin university .

The term "virtual tour" has seen a 7-fold increase in search engines from February 2020 – March 2020 due to the ongoing pandemic (Bloom, 2020). Virtual Tours have multiple different applications; tools such as "VR Gorilla" can be used to tour entire cities, museums have set up virtual tours that allow the general public to experience their exhibitions, and many U.S. national parks are also incorporating virtual tours into their business models (Vasishta, 2020).

It is evident from these findings that there is a definite demand for virtual tours when used to combat and adapt to the restrictions that Covid-19 has created for the population.

Overall, the research in the two sections above successfully outline the market for a VFA like Trail in today's climate.

2.3 Similar or Existing Solutions

2.3.1 Google – Maps

Google Maps launched on the 8th February 2005 (Gibbs, 2015), over the course of 15 years it has become the most used and recognisable maps application, popularizing the known and loved "pinch and zoom" map style interface. Nowadays, Google Maps allows a user to get directions to specific places (e.g., shops), add stops along a journey, check the current traffic level, get public transport information, and much more (Kantra, 2020).

Google Maps (if running in a browser) gets location data using HTML5's geolocation API, but as a standalone application on a mobile device the user's location will be obtained using the Global Navigation Satellite System (GNSS) or the Network Provider (P, 2016). This location is then displayed on the interactive map using JavaScript, XML, and Ajax.

Google Maps provides a functional, feature packed maps experience, however, it does not provide the specificity that a Virtual Tour Guide can. Maps allows a user to take a route to a specific place (e.g., Tesco) and the user can stop at many other places on route (e.g. a park) but, the application is limited to **places and landmarks** with the information about these places also being very brief. The functionality for if a user wishes to get directions to, and detailed information of specific trees, or memorial benches, or other smaller, but indefinitely important things, is simply not there.

2.3.2 HistoryPin

An example of a more Tour Guide-Oriented application is HistoryPin. HistoryPin is a not-for-profit Virtual Tour Guide application that allows a user to explore cultural and historical stories via an interactive map interface. It currently hosts 365,000+ stories within 27,844 tours. These tours are implemented across 2,600 cities with help from over 80,000 people (HistoryPin, 2020).

Users are initially greeted with a page that is split into two sections: an interactive map(on the left), and the corresponding tours available in that region(on the right). The map uses clusters of labels to indicate how many tours are in each location on the map, as the user zooms in, the clusters disperse into more specific places along with the corresponding tour information also becoming more specific. However, due to the constant auto-refresh of tours in the right-hand section, zooming and navigating the map at speed results in flashing of images, going against the W3C Web Accessibility Guidelines in place to minimize the possible risk of triggering seizure in susceptible individuals (w3, 2008). A benefit of HistoryPin is that it is a Progressive Web App, and it can be added to the home screen of a user's device where it can run "natively" similar to any other application regardless of their smartphone's operating system (OS).

HistoryPin proves to be a more effective system than Google Maps when it comes to gaining detailed information about events and locations. This is because of the ability a user has to undertake bespoke made tours designed to tell stories to their user base. However, History Pin is limited to virtual tours and provides no scope for undertaking physical tours assisted by their application.

2.3.3 Locatify – Automatic Tourist Guide

Locatify is another tour guide application service, focussing more on achieving this personalised "trail like" experience as opposed to the interactive map style exhibited by HistoryPin. Its primary goal is to offer guided informative tours to a user, it has been implemented in multiple museums for touring exhibits and also has applications in rental car agencies, transport companies, and other tourism related organisations (Locatify, 2020).

Locatify provides companies with the facilities to create their own completely custom virtual tour guides using an intuitive "drag and drop" style tour maker. All points of interest on a tour can be triggered via geolocation (i.e. when the user is close to the point) or clicking on the point on the map interface. Locatify works outdoors by using a smartphone's built in GNSS to locate the user, and then positioning them accordingly on the map. For indoor usage, industry standard Bluetooth Low Energy Beacons are placed around venues which then pair with a user's phone and sense their location. All data is stored locally on the user's device and hence no internet access is required to operate the application (Locatify, 2020).

Overall Locatify solves the limitation that HistoryPin demonstrated, by allowing users to undertake physical tours both indoor and outdoor, successfully creating an immersive educational experience. Unfortunately, the application interface has an outdated look resulting in an unfinished feel. Certain functions e.g. the "open in Google Maps" buttons are placed in unnatural and unfamiliar locations with unfamiliar icons breaking the consistency UI rule for button placement (Syzonenko, 2019), leading to some degree of ambiguity and confusion in usage.

2.3.4 Augmented Reality (AR) Solutions

Augmented Reality allows a user to overlay a digital augmentation across a physical environment (The Frankline Institute, 2020). This sub-section briefly highlights some AR tour guide solutions.

2.3.4.1 TUI + wikitude

Travel company TUI have been testing wearable AR technologies that enable their clients to visualize historical facts when walking around foreign holiday destinations. Their most recent 7-week testing phase allowed travellers in Palma to explore art history and middle ages events during a tour through

the city. Users are able to look at paintings with AR active zones to trigger an overlay of information that displays in their glasses. To achieve this TUI makes use of the Wikitude AR SDK and the Epson BT-300 Head Mounted Display, a state-of-the-art AR smart glasses system (Wikitude, 2019). Overall, the AR system TUI have been testing proves to be a valuable asset in the context of tourism and provides a user with an innovative method of navigating and learning about new environments, however, it is still in its testing phase and is not commercially available as of yet.

2.3.4.2 Layar

Layar is an AR system that allows corporations and individuals to develop custom AR enabled applications. Layar have a standalone mobile application hosting their own AR experience and a web application designed for custom app building. The web app consists of a simple drag and drop interface with selectable components and full personalization abilities. Layar has seen success in the AR industry with clients such as Pepsi, Coca-Cola, and BMW, along with over 46 million app downloads across their clients (Layar, 2020). Based on the Layar app reviews they provide an experience that is: “slick and accurate”, “genius and engaging”, and “has great potential”.
However, users have argued that their solution is still “US-centric”, “limited by the cost of AR layers”, and that “mobile phones do not provide the immersive experience everyone is looking for”. Appendix 1.1 contains screen captures of the full reviews.

Overall, it is evident that AR has scope for Tour Guide applications with AR mobile applications in abundance (Tankovska, 2020), however, AR is currently limited by requiring a user to hold their device in front of them to interact with their surroundings thereby reducing immersion. This issue could be solved via wearable AR technology such as “smart” glasses, but these devices are not commonplace and are only just being tested within these environments. Hence, more time for research and development is needed before immersive AR reaches its full potential in the tourism industry.

2.4 Triggering Events, User-Data Interaction

Mobile applications that connect a user to the environment face the challenge of ensuring that the relevant information is made available in a timely, user-friendly manner. There are many ways that the available data sources can be filtered to ensure relevance, such as scanning object QR codes with the device’s camera or using the user’s location to trigger events.

This section reviews the relevant technologies associated to filtering the datasets, along with methods used for event triggering. Advantages and disadvantages are also discussed.
Table 1 below details each event trigger that is explored:

Table 1 - Push/Pull Trigger Table

Trigger	Type[1]	Example	Implementable
User Interface	PULL	User presses a button in the application	**Yes**
QR Codes	PULL	User scans QR code on a tree	**Yes**
NFC	PULL	User scans NFC chip on a tree	**No**
Spatial	PUSH	User walks into trigger zone	**Yes**

2.4.1 Spatial Triggers / Geofences

Events can be triggered by the user's position, for example when walking in a predefined zone near a marked tree. This spatial event triggering uses geolocation technology such as GNSS[2] (e.g. GPS), GSM or Wi-Fi Positioning (WPS), to track a user's device in relation to a location (e.g. a tree), and triggers data delivery when a user is within a certain distance (Per Persson, 2003). This method is known as Geofencing, where a virtual polygon boundary called a Geofence is created around a spatial range of interest to be used as the "triggering zone" (White, 2017). Technically, the Geofence is used to detect when a point[3] is inside the polygon, i.e., directly pairing a point against a polygon (Suhua Tang, 2015). Suhua Tang further exclaims the benefits of geofencing; highlighting its ability to generate continuous data streams and strong event triggering capabilities, making it a potentially valuable tool in the aspect of Tour Guide Application development. Geofencing has seen success in applications such as GeoNotes where a user can access geopositioned virtual post-it notes via triggering an event through entry into their geofence (Cöster, 2003).

To implement geofencing, tools such as Google's Geofencing API could be used. This works in partnership with Android's API and provides an easy-to-use system for handling geolocation data and triggering events via geofences (Google, 2020). Being by Google it allows for strong compatibility with Google Map driven applications, however, due to the API being only available for Android devices[4] this tool is not suitable for a cross platform web application.

Another geofencing tool is PostGIS, an extension that adds spatial datatypes and functions to PostgreSQL[5] . PostGIS allows for location queries to be run on data, for example analysing how far a

[1] PUSH or PULL where PUSH refers to the application pushing data to the user and PULL refers to the user performing an action to pull data from the application.
[2] Global Navigation Satellite System
[3] i.e., a user
[4] IOS implementation would require a developer to use IOS Core geofencing tools with Swift, and hence is not integrated with the Geofencing API
[5] PostgreSQL is an SQL open-source object relational database system that allows user to create custom types, custom functions and more (PostgreSQL, 2020).

user is from some geolocation; this data can be stored in a filetype called a GeoJSON file (PostGIS, 2020). A geofence event trigger can be built using some spatial SQL provided by PostGIS on top of PostgreSQL data. PostGIS can also store polygon data used for geofencing, this data can then be applied with the user's location and piped to a compatible Map API for display. To generate a geofence a point and a radius is needed, once obtained, the function `ST_Buffer (geometry, distance)` can be used to create the geofence[6]. Using the method `ST_DWithin()` paired with a SELECT statement a user can check if a point is within the geofence. These methods allow successful and efficient creation of Web Apps that use geofencing technology (DZone, 2017).

Unfortunately, there is no cross compatible HTML5 geofencing API for web apps; but through the use of PostGIS and external map APIs e.g. Google Maps an effective spatial triggering system could be created for a web application like Trail.

2.4.2 Quick Response (QR) Codes

A limitation to spatial triggering is its lack of ability to geolocate accurately enough for specific objects to trigger an event i.e., it is confined by at best a 3-metre accuracy (agup, 2013).
QR codes provide a method of event triggering directly from an image attached to the desired object. These codes are versatile methods of data storage, allowing a developer to embed any kind information into the code, for example tree data. For 15 years QR code have seen an exponential growth in terms of their uses, from an automated SMS message to business cards (Ching-yin Law, 2010), and in the current climate of Covid-19 they are proving extremely beneficial. QR codes are easy to set-up and use, and have been found to require a low technical barrier with standard smartphones now coming equipped with QR readers pre-installed onto the device (Ching-yin Law, 2010).

Quick Response codes fall under the family of 2D barcodes, however, QR codes have a much higher data capacity; being able to store up to several hundred times more information than the 20-digit barcode, precisely 7,089 characters are able to be encoded into a single symbol. They are also able to handle multiple data types ranging from alphabetic characters, to binary (Denso, 2010).

These codes demonstrate physical benefits over regular barcodes, being able to hold the same amount of data in one tenth of the size and carry this information both vertically and horizontally. Alongside this they are robust and able to withstand dirt and damage via error correction capabilities provided by adding Reed-Solomon Code(a mathematical error correction traditionally used for CD's but is capable of correcting at the byte level). A QR Code is made up of codewords where one codeword is equal to 8-bits, when Reed-Solomon Code is applied to the original data it is possible to

[6] Where geometry longitude and latitude of a point, and distance is the radius.

correct up to 25% of a damaged codeword (Denso, 2019). Furthermore, QR codes provide full 360-degree readability via position detection patterns (Fig. 1) that are place in three corners of the code, guaranteeing fast and stable reading while minimising interference from potential background hazards (Denso, 2019).

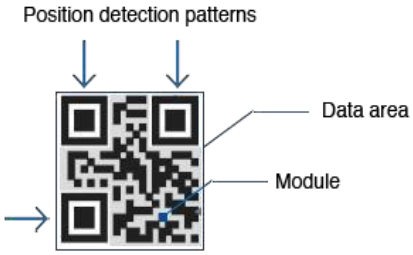


Figure 1– How Position Detection is Implemented in a Code

QR codes must abide by the standard distance to size ratio of 10:1, meaning the optimal distance to scan a QR code is 10 times the width of the QR code (sproutQR, 2020).

While there are advantages to QR codes, there are also a variety of disadvantages (see Table 2).

Table 2 - QR Code Pros and Cons

Advantages	Disadvantages
Quick and convenient event triggering (Mall, 2016).	QR codes only work in line of sight, if a code is on the other side of an object it will be hidden
Provides instant access to potentially unlimited information (Mall, 2016).	Codes can be difficult to scan in very bright or very low lighting (Fotaflo, 2018).
360-degree readability (Denso, 2019).	Rajendra Singh found that QR codes can be used fraudulently to send malware if scanning offline, as no trusted source can be identified (Rajendra Singh, 2013).
Multiple data type support (Denso, 2019).	User must own a smartphone
Widely compatible with current devices (Denso, 2019).	Susceptible to vandalism
Can be as small as 2 x 2 cm and as large as a user wishes, provided the pixels are clear (sproutQR, 2020).	If QR code redirects to a URL then an internet connection is required
QR codes can be added to virtually any surface (Mall, 2016).	
QR codes can be used offline if data is encoded directly into the image (Jayce, 2019).	

There are an abundance of tools available for developers to create their own QR code readers and generators. Some of the most widely used readers are the ones already implemented in the average smartphone. However, some devices like Huawei or any Android device running version 7 or below requires a 3rd party application for QR code scanning (Choudhary, 2020).

A range of 3rd party QR code scanners and generators are discussed below, highlighting their main features, advantages, and limitations.

2.4.2.1 Google Charts API

Google Charts is an API service created by Google that allows developers to create custom graphical charts and embed them into their webpages (Google, 2020). Their infographics server dynamically creates an image given a URL POST or GET request. However, there are limitations to this tool; URLs are limited to a max size of 2Kb which means if you have more data then you will have to use POST requests instead of GET and requires using another language such as PHP (Google, 2019). Google also stated that the infographics server is deprecated and only the charts server is actively maintained, hence, Google Charts QR API is not extendable and outdated.

2.4.2.2 QRCode Monkey API

A more well maintained, customisable QR code generator is QRCode Monkey. QRCode monkey is an API that has multiple language (e.g. jQuery, PHP) support and many implementation examples ready to be added to a user's source code. The API offers the generation of QR codes in PNG, SVG, PDF and EPS formats, with the ability to customise colours, pixels, frame shapes, and add logos (QRCode Monkey, 2020). An example of the possibilities achievable with QRCode monkey API is shown below (Fig. 2).

Figure 2 - API result examples – RapidAPI QRcode monkey

Overall, QRCode Monkey provides a modern, efficient and customisable solution to generating QR codes for use in any environment.

2.4.2.3 Minhaz's html5-qrcode

HTML5-QRcode is an open source QR code scanner written using JavaScript, it is cross compatible with all HTML5 supporting browsers across all devices (with exception to IOS where it is limited to safari due to other versions of IOS browsers not having webcam support) (Minhaz, 2020). The scanner allows a user to read a QR code live via the camera feed or alternatively, a user can upload an image (all images are stored and analysed locally).

The basic implementation of Minhaz's html5-qrcode is without any external frameworks and provides the core functionality of the QR code scanner; if a user chooses, there are many extra features that can be added to this scanner such as personalised UI's, the ability to stop scanning once a code is scanned, and custom error handling. Due to camera permission and access via web browser only being a feature on HTTPS Secure webpages, this scanner can only function on HTTPS certified sites.

From this analysis, Minhaz's html5-qrcode proves to provide an easy to implement, user friendly option to embedding efficient, cross compatible live QR code scanners in web apps such as Trail.

2.4.3 Near Field Communication (NFC)

Another object specific triggering approach and method of attaching digital information to a specific real-world object is to embed a Radio Frequency Identification (RFID) tag which can be scanned. NFC is a variation of RFID technology that operates at a 13.56MHz frequency (M. Mareli, 2013).
For more technical details on how NFC works see Appendix 1.0.

After looking in depth at NFC software such as NFC Tools (Wakadev, 2019), and libraries such as Core NFC (Apple, 2020), Pokusew's nfc-pcsc (pokusew, 2020), and Web NFC (WebNFC Community Group, 2020), it is clear that the prospects of NFC are huge. However, it is still very much a new and upcoming technology; this is highlighted by the little support for it in web environments and the limited open-source tools available (wwcdt, 2020). NFC is also limited by its slow data transfer rates at a maximum of 424Kbps usage proximity of 10-20cm which in the context of Trail would pose an issue of where to install the chip on the tree to cater for varying heights of users (RFwireless, 2020). NFC is growing and becoming more usable (BlueBite, 2020) but for the moment it is not yet ready for web environments, meaning it would not be possible to implement successfully in the Trail application.

2.4.4 User Interface

Another method of the user accessing information is by them pulling the data via the user interface. User Interface design is a key factor in the functionality and effectiveness of an application, it is how a user interacts with content and is arguably the most important element of any web project (Image+, 2018).

The button is a simple yet vital element of every user interface (Perea, 2019) and provides(in most cases) the bridge between triggering an event and displaying data. Buttons are styled links engineered to grab the user's attention and are strongly linked to productive and positive User Experience (Perea, 2019). There are many aspects that are taken into consideration in order to create a successful and useful button; one main button type is the CTA (Call to Action) button, these are specifically designed

to motivate a user to complete an action in an application (Broersma, 2020). HTML5 has a built in *<button>* element that generates a basic button to interact with that is cross- browser compatible. Using CSS3 or a Bootstrap framework, style and animation can be added to these buttons in order to design a more personalised, attractive element (W3schools, 2020).

However, while buttons and other User Interface components do provide a successful data triggering system, they lack tangible feel and interaction. For people who are physical learners (most likely users of the Trail Application) or are not used to using web apps (e.g., an older demographic) they can be fairly complex and daunting.

2.5 Information Delivery

Once events are activated, there must be some form of data delivery so that information can be obtained by the user. This could be done a number of ways from textual presentation(e.g., sentences), graphical presentation(e.g., images, graphs), to audible methods. There are advantages and disadvantages associated to each which are discussed in the following section.

2.5.1 Textual

One form of information delivery is via textual presentation: consisting of ordered text comparable to a search engine output (Emile Morse, 2002). Textual presentation of data allows a researcher or developer to represent certain data that can't be presented effectively in a graphical method e.g., qualitative data (ResearchArticles.com, 2019). A study conducted by S. Koshman analysing the effectiveness of different information delivery methods found that users preferred a textual based system because of its familiarity and low learning curve (Koshman, 1996). Due to this study being conducted in 1996, the results found may be dated due the general familiarity of other delivery methods having grown, however, the results are still valuable to provide a basis argument for textual based systems.

Majority of applications today - especially web apps, use textual delivery as their main delivery method and interlace graphical, and other methods on top to improve the experience. Primarily, this is due to websites and web apps consisting largely of HTML which is by definition a Text Mark-up Language (w3schools, 2020). With all web apps originally stemming from pure HTML coded websites, it is understandable that textual delivery is still an extremely prominent method in today's applications (Praveen, 2015). However, there are limitations to textual only based systems when it comes to accessibility and user experience – namely, it can be difficult for a user to draw conclusions at a glance and data can appear overwhelming (ResearchArticles.com, 2019). Hence, solely textual information delivery is not adequate for complete effective use of modern a web application.

2.5.2 Graphical

Graphical information delivery is a method of presenting data that overcomes some of the limitations highlighted previously in textual delivery. Graphics are used for enhancing the appearance and readability of web apps while in turn entertaining, educating, or impacting a user emotionally. They consist of things such as maps, images, diagrams, graphs, and more (w3c, 2020).

Contrary to the results found by Koshman supporting textual delivery; Lucia Mason, author of "Textual and Graphical Refutations: Effects on conceptual change learning" found in a study that more stable learning was found in users interacting with graphical data, providing solid arguments for the context of graphics and text-graphic pairing to sustain knowledge.

In an environment such as Trail – a virtual tour guide application, the sustenance of knowledge is a key goal of the system, and hence graphical methods of information delivery seem beneficial. Trail could implement a graphical map driven system showing a user where they are in relation to trees on the route. One proposed limitation of graphical delivery is it does not solve the accessibility issues for the visually impaired. According to the World Health Organisation, in 2010 there were approximately 285 million visually impaired people in the world (WHO, 2010). A survey by Dr Nora Griffin-Shirley (Griffin-Shirley N, 2017) conducted on 259 visually impaired participants found that apps that had special features like speech and font options were considered useful and accessible by ~90% of participants. Hence, it is important to include audible and textual solutions in Trail to solve the problems faced by those with visual impairment.

2.5.3 Audio and Speech

To cater to this larger audience and allow wider accessibility to the Trail application, audible information delivery methods can be implemented. Researcher Anna Chang (2015) conducted a study into the effect of audible delivery methods on the comprehension of data and showed that both test groups improved their comprehension levels when using audio assistance (Anna Chang, 2015). This finding proves to be extremely important when correlated with disability statistics across the UK; With estimations that 1 in 10 people have some form of dyslexia (NHS, 2018), and approximately 2 million people are living with visual impairment (NHS, 2018).

Audible information delivery can be implemented in multiple ways, the most common being via the HTML5 `<audio>` element where a developer can define the source content (e.g., mp3), media controls, descriptions, and many other attributes (Mozilla, 2020). However, these audio files are limited by the pre-recorded content they contain, meaning accessibility would only be possible in certain sections of an application.

To combat this limitation, tools such as text-to-speech synthesis can be used. Text-to-speech

synthesis is the process of using algorithms and textual analysis to convert written text into a synthetic generation of speech (Taylor, 2009). To implement speech synthesis in a web application a developer can using the SpeechSynthesis Web API, the controller for Web Speech API. This API provides a simple set of functions that can be used to synthesise text into speech. Using the following piece of code "Hello World!" can be synthesised:

```
let utterance = new SpeechSynthesisUtterance("Hello world!");
speechSynthesis.speak(utterance);
```

There are multiple parameters that can be edited to adjust pitch, speech rate, and the voice used for synthesis. The SpeechSynthesis API is compatible across majority of browsers other than Opera for android and proves to be a valuable tool to counter the issues of accessibility and pre-recorded audio.

Overall, from the arguments given above, audio improves a user's experience by not only allowing for wider accessibility and comprehension ability, but also assisting in memory recall and giving a user the freedom to listen to information while performing other activities e.g. walking (Samantha Abelinde, 2014). Hence, audible information delivery would be a valuable asset in an implementation like Trail, allowing a user to listen to tree information either by playing an MP3 clip or speech-synthesis initiated by a spatial event trigger while walking around Heriot-Watt Campus.

2.6 User Alerts and Notifications

Once data has been triggered or delivered, it would be beneficial to alert a user of this update. User alerts and notifications allow an application to "stay in contact" with a user and help promote user activity. This section aims to discuss the main methods of alerting a user and how they can be implemented in a PWA, specifically, for the environment of a virtual tour guide.

2.6.1 Push Notifications

A push notification is simply a small pop-up message alert directed to the user to update them with some information or suggest that they perform an action (Cornelis, 2020). These notifications are a well-known tool in today's technology with the average user receiving upwards of 60 notifications on their smartphone daily (Alireza Sahami Shirazi, 2014). A study conducted by Niranjan Bidargaddi (N.Bidargaddi, 2018) found that tailored push notifications had a directly positive result on user engagement and recurring use of an application. However, due to the Trail application only interacting with a user when they are using the application, push notifications do not seem necessary for the scope of this project. Full push notification discussion/analysis can be seen in Appendix 1.2.

2.6.2 Haptic Feedback

Haptic feedback is the process of interacting with the user via complex vibration (ultraleap, 2020). Using a complex vibration as an alert method eliminates the negative aspects that pop-up style push notifications were found to create. Other notification systems such as SMS use simple vibration (e.g. one single vibration) to alert a user when a new message has arrived (precisionMicrodrives, 2020). Because every application has its own vibration alerts, haptic feedback from the Trail application could get lost if only a single vibration is used. Hence, to maximize the success of haptic feedback a continuous 3 second vibration could be used to directly notify the user that the Trail app has detected the user's entry into a geofence.

To implement a feature such as haptic feedback in a PWA like Trail, a developer can use HTML5's vibration API. The following segment of code highlights the function that allows a user to trigger device vibration (Mozilla, 2020):

```
window.navigator.vibrate(200);
```
(where 200 is the vibration time in ms)

One limitation of this function is that it is not compatible with the Safari browser on IOS devices, however, it has full support across all other smartphone browsers (Mozilla, 2020).

Some key advantages and disadvantages of haptic feedback are described in Table 4:

Table 3 - Pros and Cons of Haptic Feedback

Advantages	Disadvantages
A study conducted into the use of haptic technology in the education system found that haptic feedback can enhance authenticity and immersion of interaction (Lu-miao Liu, 2017).	Haptic feedback is implemented with all push notifications and hence, it can be confusing to differentiate between haptic vibrations. (precisionMicrodrives, 2020)
Research from YaleNUS College shows that haptic feedback effectively alerts users during physical activities, more so than audible or visual alerts (College, 2020)	A study led by Yu-Hsuan Lin (Yu-Hsuan Lin, 2013) discovered that approximately 80% of people experience "Phantom Vibration" – the hallucination of mobile vibration. Potentially leading to incorrect checking of the device throughout the trail
Rhonda Hadi (Rhonda Hadi, 2020) writing in the *"Journal of Consumer Research"* found that haptic alerts can improve consumer performance by creating a more personal technological exchange .	Haptic phone alerts fall under the same category as push notifications and hence they are also prone to annoying a user (Localysts, 2015)

Overall, in an environment like trail where a user may frequently put away their device to travel from point to point, haptic feedback would prove to be an extremely beneficial tool to notify a user when they are in close proximity of a specified tree.

2.7 Sourcing and Generating Map Data

Geographical map data must be sourced and then generated in order for a user to undertake a map driven physical or virtual tour using the Trail application. This section aims to identify the most efficient and usable tools to do so.

2.7.1 Google Maps

Google Maps has a set of extensions specifically designed for web app developers, two of which are called My Maps and the Google Maps API. These tools allow a user to manipulate and interact with Map features in their own applications. My Maps gives a user the ability to generate embeddable, completely custom maps while providing the simplicity and cross compatibility of Google Maps. The My Maps platform is a web-based application and with it, a user can add custom points(e.g., trees), existing places, import large datasets of geocoordinates to plot, and add complete personalization - with style options for all attributes and the functionality to add images and videos (Google, 2020). A custom map is easily embedded into a webpage via one line of code.

In summary, My Maps provides an extremely functional and intuitive system for the creation and embedding of custom maps into a web application. However, because it is a UI driven tool designed for the “general” public, it is fairly limited in the sense it lacks the ability to manipulate map data at a more technical code level.

Alternatively, the Google Maps API provides the ability to create custom maps directly from the code of your application, with no need for a “middle-man” in between. Using simple API calls, HTML, and JavaScript a user can embed maps, import data sets, and customize features all via a more low-level interface.

The benefits of the Google Maps API are evident by its ability to provide such specific map manipulation options. However, the data that can be added to a map is limited to simple vector and raster overlays, furthermore, the system as a whole is more cumbersome and less efficient than that of My Maps. For developing an application like Trail, the best approach would be to use My Maps in conjunction with the Google Maps API i.e., generate the custom map with My Maps and then import it to be manipulated further using the Google Maps API.

2.7.2 OpenStreetMap

An open-source map dataset that could be used is OpenStreetMap (OSM) which has worldwide coverage and is editable by the public (OpenStreetMap, 2020). OSM allows for offline map generation, which can be extremely useful when a user cannot obtain an internet connection or phone signal (Selle, 2015). OpenStreetMap consists of a variety of frameworks to help a user achieve specific goals. An example use sequence of these frameworks could be accessing raw OSM data, processing the data, generating map images, displaying a static or interactive map, and navigation tools. Each respective section has an abundance of different libraries for different platforms and languages (OpenStreetMap, 2020). OSM gives a developer the option to create a custom map in something like Google's My Maps and import that dataset using the OSM frameworks for generation of an offline fully custom map. Datasets can be imported into OSM using a tool such as Umap[7].

Overall, both Google Maps and OpenStreetMap provide solid functionality for generating custom usable maps that can be embedded in web applications. OSM shows strengths that the Google Maps tools do not such as the offline support. However due to Trail being a Progressive Web App, offline support is not a main feature the system would cater to.

2.8 Sourcing User Location Data

In order to use geolocation features in the Trail application, the user's location must first be sourced. This section will cover the main method of sourcing that location and any issues that could arise when doing so.

2.8.1 HTML5 Geolocation API

HTML5 has a built in API specifically designed for geolocating a user of a webpage. This API is cross-compatible with all devices and common browsers, however, as of Chrome version 50.0 a secure HTTPS connection is required in order to use the API. Because sourcing the location of a someone compromises privacy, the user must consent the geolocation before the API can begin functioning (w3c, 2020). Once a HTTPS (SSL) certificate is issued on the site and consent has been given, a developer can easily obtain the current location of a user via the following method:

```
getCurrentPosition()
```

The return data of this method can be specified to give the developer the longitude and latitude of the location. Using these values, a point can be generated on a map using one of the map generation tools detailed in the previous section, or many other tools, e.g., GeoPandas – a Python geospatial data

[7] Umap works with OSM to create OSM maps with custom points via a user interface interaction.

tool (GeoPandas, 2021). This in turn allows visualisation of the user's location to assist with features such as geofencing and navigation (w3c, 2020), and hence, is an ideal tool to help develop the functionality needed for the Trail web application.

2.8.2 GNSS Interference and Issues

In general, a phone's Global Navigation Satellite System (GNSS) using HTML5's geolocation API with `enableHighAccuracy`[8] enabled can geolocate with an accuracy as close as 3 metres (only in open landscapes) (agup, 2013). However, there are external factors that can cause interference, in turn worsening the accuracy of GNSS. The GNSS signals (e.g., GPS, GLONASS) are extremely low power because of their long satellite-receiver distance (Matthias Wildemeersch, 2010), this makes them easily susceptible to interference from the atmosphere and radio frequencies (Bours, et al., 2014). Ahmad Norhisyam Idris author of "Effect of radio frequency interference (RFI) on the Global Positioning System (GPS) signals" found that these radio frequencies can weaken the accuracy of GNSS .

RFI is not the only issue presented when using GPS, non-Line-of-Sight (NLOS) and multipath interference also pose inaccuracies to a system when in an urban environment (Enge, 2011). This is due to buildings, vehicles, and other large structures reflecting and blocking the GNSS signals (Li-Ta Hsu, 2015).

Overall, it is clear that urban environments can cause interference when it comes to the accuracy of geolocating a user. Fortunately, Heriot-Watt Campus is located in a fairly rural area and hence, the interference problems outlined above may not arise to the same degree.

2.9 Progressive Web Application (PWA)

It is important that the Trail app is cross-compatible and downloadable, and that it provides an easy, friendly user experience. Normal websites and web-applications solve the issue of being cross compatible by being hosted on the internet and accessed via standard browsers, however, these websites are restricted to being accessed via web browsers. A study conducted by Compuware showed that 85% of consumers prefer using a mobile application to a mobile website (Moth, 2013). This high percentage could be a result of the easy accessibility of an app– clicking an app icon on the home page vs navigating to a standalone webpage, the usability – an app is designed specifically for a mobile device whereas many webpages are not, or the features –apps support features such as push notifications and can link to other applications on a user's device.

[8] This is a Boolean that indicates if the application wishes to receive the best results possible, it may compromise response time and increase power consumption (Mozilla, 2020).

A PWA is a web application that is intended to be completely cross-compatible, working with standard browsers on both Desktop and Mobile. PWAs are designed to be added to a user's home screen or desktop where they can run "natively" like a standard application. To create a progressive web-app that runs locally, a series of strict requirements must be met (see Table 5).

Table 4 - PWA Requirements

Criteria	Description
Served over HTTPS	A secure HTTPS, SSL certified website is required. This is primarily due to the Service workers (the backbone of a PWA) needing to be protected behind HTTPS for security, as they run in the background and undertake tasks that could compromise privacy (Love, 2018).
Includes a Web Manifest File	A web manifest file is a json file that gives a user's browser information about the PWA, such as, the application name, the start URL, home screen icons, and display options like "*fullscreen*" (LePage, 2020).
A Service Worker with a *fetch* handler for Offline use	A JavaScript file running in the background that caches files, retrieves files from the cache, and intercepts network requests. The *fetch* handler serves files from the cache by intercepts network requests when offline.

PWAs have shown to extremely beneficial for user engagement with large organisations such as Alibaba, Twitter, Pinterest, Trivago, and many more all adopting PWAs alongside their native applications (Lee, 2020). However, while PWAs have their benefits they also harbour a variety of drawbacks (see Table 6).

Table 5 - Benefits and Drawbacks of PWAs

Benefits	Drawbacks
PWAs are secure because they must be hosted via HTTPS. This in turn can prevent spoofing and eavesdropping of data.	iOS has severe limitations with the features it allows for PWAs. Meaning there is an imbalance between what can be developed for android vs iOS .
Because they are developed using standard web languages and tools it makes them cheaper to create and run than device specific alternatives (Suschevich, 2020).	Because PWAs are written in JavaScript and web languages the applications are limited to the scope of those languages (Kościelniak, 2021).
PWAs improve user interaction. This is proven by organisations on average reaching a 134% increase in page views once they convert to a PWA (Software Brothers, 2020).	Older mobile devices with older browsers will simply not be able to support PWAs and hence individuals may be excluded (Warcholinski, 2021).
Load times are drastically better on a PWA. The average load time of a PWA is 2.75 seconds which is roughly 8x faster than a standard mobile webpage (Software Brothers, 2020).	

Chapter 3: Project Management

3.1 Project Scheduling

The Gantt Chart seen in Appendix 2.1 outlines how the project will be undertaken using an agile methodology and broken up into sprints. It details all aspects from design, development, to dissertation writing. First semester focussed on planning the project and completing Deliverable 1; application development began in 2nd semester along with thesis writing and further literature reviewing. The Kanban approach was used for organisation, where tasks are organised across a digital board into **needs to be done**, **in progress**, and **completed**. For an example see Appendix 1.4.

3.2 Consideration of Professional, Legal, Ethical, and Social Issues

3.2.1 Professional

All work produced throughout the Tour Guide Project was done so professionally, using industry standard methods for planning, analysis and development. The project was well organised following the schedule outlined in the Gantt Chart, and supervisor meetings were arranged weekly throughout the process. Furthermore, to ensure security, all user data was pseudo-anonymous and stored accordingly in encrypted files. Backups of the project were kept via tools such as GitLab (for source code). As a representative of Heriot-Watt University, professional conduct was carried out religiously in all situations, with emphasis on scenarios involving staff and evaluation study participants.

3.2.2 Legal

This project required the collection of personal data, because of this, certain legal aspects were considered. It was vital that the project adhered to the specifications outlined by the Data Protection Act (1998) and the EU General Data Protection Regulation (2016). A significant amount of the data that was be collected during the project (such as device locations, age, and gender) is considered sensitive information and was pseudo-anonymised, stored securely (in an encrypted file), and only accessible by authorised personnel. It is important that the project did not discriminate in any way, all persons were treated fairly and respectfully as highlighted in the Equality act (2010). Finally, when using external sources all copywrite issues were meticulously handled using appropriate accreditations and referencing in legal compliance with the Copywrite, Designs and Patents Act (1998).

3.2.3 Ethical

It is vital that ethical issues were taken into account throughout the completion of this project. This project and its user evaluation studies were granted ethical approval by Heriot-Watt University in response to the submission of an ethical approval form. Participants were fully briefed on studies being undertaken, along with what was required from them and how their data would be used. All participant data is pseudo-anonymised meaning no personal or identifiable information was stored. Finally, all participants must were over the age of 18 with explicit consent given.

3.2.4 Social

The Trail application has the ability promote outdoor activity in users. Allowing people to experience nature in a new way is beneficial to raise awareness of our environment and lead to the appreciation and preservation of the Earth. The application also creates the opportunity strengthen the links between Heriot-Watt and the wider community, deepening the accessibility to the university campus' vast ecosystem and potentially leading to new prospects. Finally, it is proven that exposure to the outdoors and nature has a directly positive effect on mental health (David Pearson, 2014).

Chapter 4: Project Requirements Met

This section outlines the Functional and Non-Functional Requirements proposed for both the database and front-end of the system. Functional refers to the services the system **must** offer while Non-Functional aims to define system attributes like reliability, usability and performance, reflecting the system quality (guru99, 2020).

Requirements have been marked with a **YES** or **NO** response in the "Completed" column. This outlines whether the requirement was met and implemented successfully in the final system. Alongside this, a column referencing a specific section of the thesis is included, pointing to where that requirements successful implementation was discussed in depth.

4.1 The MOSCOW Method

MoSCoW is a technique used to prioritise system requirements. Using a Must, Should, Could, and Would system it means the most important requirements can be highlighted and implemented first, theoretically minimising scope for an unfinished application. The specificities of this method are outlined below:

Table 6 - MOSCOW Definitions

Letter	Meaning	Description
M	Must Have	These requirements **must** be implemented to be deemed a complete project.
S	Should Have	These requirements **should** be implemented but are not vital.
C	Could Have	These requirements **could** be implemented but are considered optional.
W	Would Have	These requirements **would** be implemented if there was enough time.

4.2 Functional Requirements

Database:

Table 7 - Database Functional Requirements

Requirement ID	Requirement	Priority	Completed	Section
DFR1	The database must allow for the storage and querying of location data like coordinates.	M	YES	5.1.4

Application:

Table 8 - Application Functional Requirements

Requirement ID	Requirement	Priority	Completed	Section
AFR1	The user must be able to interact with a map interface	M	YES	5.6
AFR2	The application must obtain an active user's location	M	YES	5.4
AFR3	The application must display an active user's location on the map	M	YES	5.4
AFR4	The user must be able to view data about trees on the HWU campus	M	YES	5.5.2
AFR5	The application must allow for integration with QR technologies	M	YES	5.5.2
AFR6	The user should be able to scan QR codes using the app interface	S	YES	5.5.2
AFR7	The application should notify a user when they are in the vicinity of a tree	S	YES	5.5.1
AFR8	The user could be able to view data about trees in greater Edinburgh	C	NO	5.2.5
AFR9	The user could be able to add trees to the application	C	NO	n/a
AFR10	The user would be able to create their own tours	W	NO	n/a

4.3 Non-Functional Requirements

Table 9 - Non-Functional Requirements

Requirement ID	Requirement	Priority	Completed	Section
NFR1	The application must be cross-compatible across different devices	M	YES	5.3
NFR2	The application interface should adjust to different screen sizes	S	YES	6.1
NFR3	The application must be HTTPS secure	M	YES	5.3.1
NFR4	The user should be able to run the web app natively as a PWA	S	YES	5.3.2
NFR5	The application GUI should be clear, attractive, and well laid out	S	YES	5.2
NFR6	The application could be able to notify the user via push notifications or haptic feedback alerts	C	YES (haptic)	5.5.1
NFR7	The application should be efficient with ratings > 80 on Google Lighthouse	S	NO (performance affected)	7.1
NFR8	The application should have special accessibility options	S	YES	5.5.1
NFR9	The application should securely store user location data	S	YES	5.4
NFR10	The application should be easy to navigate	S	YES	5.2
NFR11	The application must be available 24/7	M	YES	5.1.1

Chapter 5: Implementation

This section discusses the implementation of the Trail tour guide application. The core functionality of the system allows a user to get information about a tree via spatial triggering, scanning a QR code, using AR, or interacting with the map UI. The app alerts the user when they are near a tree (point) using haptics (vibration), the relevant information relating to that tree is then relayed to the user via speech and graphical interfaces. If the user should stray far from the HWU campus tour route, then the system will alert them and show a direction arrow to guide them back to the tour. Each individual feature and its components are discussed in depth in the following section, covering the requirements met, implementation (method), and any testing that took place.

The following table highlights the some of the key features of the application:

Table 10 - Main Feature Sections

Feature	Section
Spatial Triggering (geofences)	5.5.1.1
QR Codes	5.5.2.1
Augmented Reality	5.5.2.2
Speech Synthesis	5.5.1.3
Haptic Feedback	5.5.1.2
Direction System	5.7

5.1 Code Stack | Entity-Relationships

This section explains how the application works “behind the scenes”; detailing the languages used, how it is hosted and deployed, how it communicates, how data is stored, and the relationships between everything. The following Entity-Relationship Diagram (Fig. 3) highlights how the system is connected. The layout of this diagram is based on the typical development flow and deployment exhibited over three months of implementation.

Going from left to right; completed code is pushed to GitHub. The Heroku application host then detects an update to the Master branch of the project and receives this code. Heroku then builds the application and deploys it.

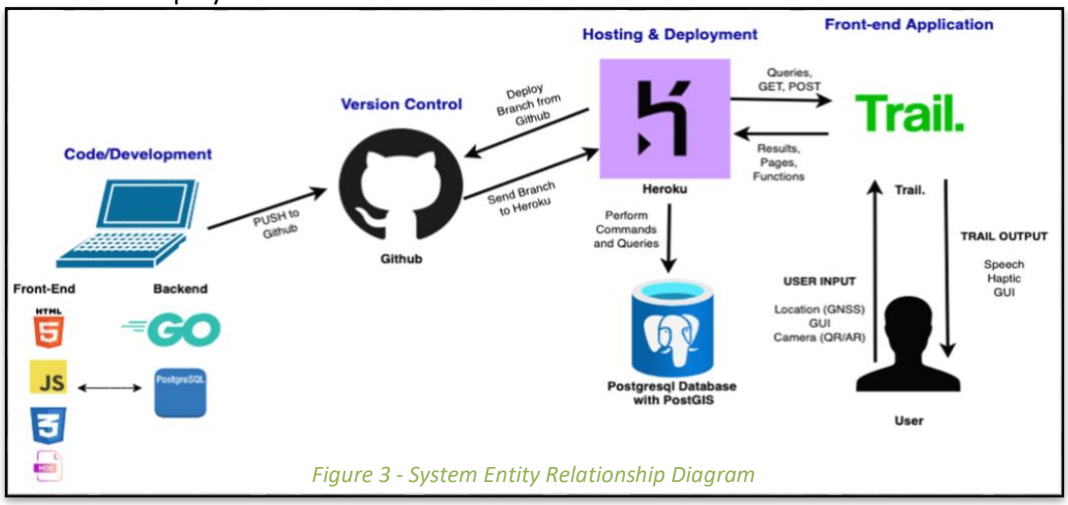


Figure 3 - System Entity Relationship Diagram

The user then interacts with the application, sending requests to the server. The server handles these requests and interacts with its database if it needs to satisfy any queries.

5.1.1 Hosting and Deployment

- **Heroku** – A cloud application platform for 24/7 hosting and deploying web-apps.
    - Supports multiple programming languages including Golang.
    - Built-in database hosting.
    - Paid subscriptions allow for SSL certified custom domains (https).
    - Automatic deployment when code is pushed to a repo - GitHub Integration.

5.1.2 Back-end Language and Frameworks

- **Golang v1.13** – Primary language used for backend development.
    - Memory safety, garbage collection, structural typing.
    - Handles page routing, database connections and server requests.

  Frameworks include: Gin-Gonic(HTTP web framework), Go-SQL-driver.

5.1.3 Front-end Language and Frameworks/Libraries

- **HTML5** – Markup language used as the foundation of any web application.
- **CSS3** – Describes how HTML elements are displayed.
- **JavaScript** – A front-end scripting language used to add functionality and handle API calls.

  Libraries include *jQuery* (simplifies complex JavaScript and Ajax), *Material Design Bootstrap "MDB"* (an open-source user interface kit), *Google Maps JavaScript API* (custom map generation), *html5-qrcode by Minhaz* (a JavaScript QR-code scanner), *WriteIT.js by Khushit* (text writing effect), Google *Marker Clusterer* (clusters markers on map), *Web Speech API* (text-speech synthesis), *User Agent Parser by Faisalman* (OS detection), and *AR.js* (JavaScript AR framework).

5.1.4 Database

- **PostgreSQL with PostGIS extension** – An open-source database management system with an extension that allows for the storage and querying of location data.

5.1.5 Local Machine and Integrated Development Environment (IDE)

- **MacBook Pro 2018 running Mojave 10.14.6**
- **Visual Studio Code** – An IDE with debugging support and syntax highlighting.

5.2 User Interface (UI)

The application User Interface was developed using a range of HTML5 elements and CSS3 features. The majority of the design was enhanced via the integration of the Material Design Bootstrap (MDB) UI kit which strives to make interface development tasks (such as layout and styling) less cumbersome. Both Mobile and Desktop User Interfaces will be discussed and compared in this section. The User Interface implementation achieves Non-Functional Requirement 2 (NFR2) stating the Application should adjust to different screen sizes, Non-Functional Requirement 5 (NFR5) outlining that the GUI should be clear, attractive, and well laid out, and Non-Functional Requirement 10 (NFR10) – the Application should be easy to navigate.

5.2.2 Home Screen

Upon navigating to the web-app, the first page the user is greeted with is the Home screen. The Home screen is the central location for users to access facts about trees, statistics, and useful information.

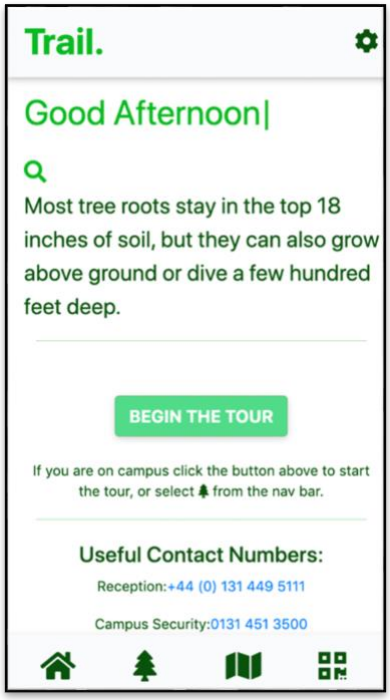


Figure 4 - Mobile Home Screen

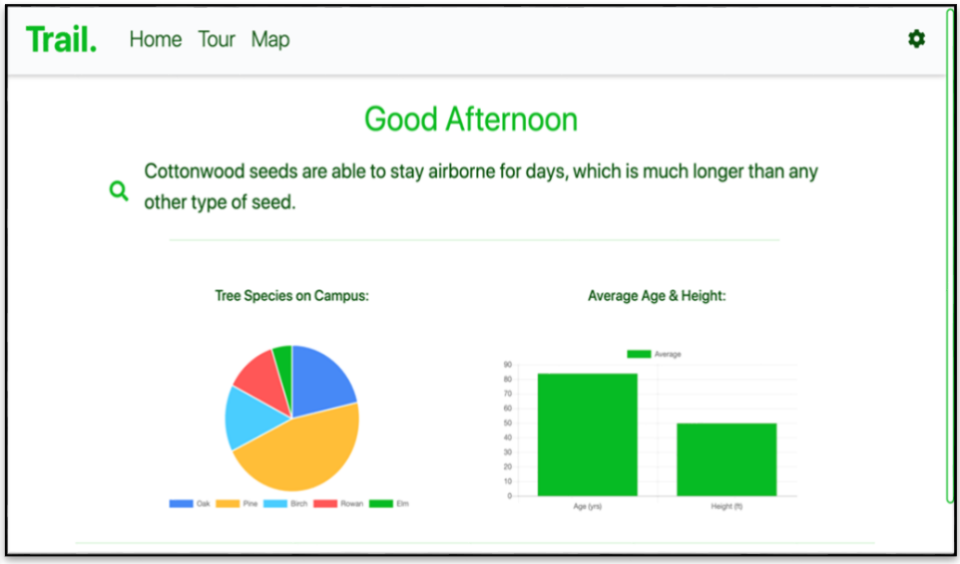


Figure 5 - Desktop Home Screen

In both versions the user is welcomed with a greeting that is dynamic depending on the time of day. The greeting works by using JavaScript to get the current time and assigns either 'Morning', 'Afternoon', or 'Night' depending on the value. This is then animated using the WriteIT.js framework. See Appendix 3.1 for source code. Furthermore, the user is also displayed a random heriot-watt/tree related fact from a json file (updated ever 20 seconds). See Appendix 1.4 for source code.

Mobile – Because the tour can only be undertaken on mobile, the mobile version shows a large 'BEGIN THE TOUR' button (Fig. 4) which links to the Tour Page. They are also given useful Heriot-Watt contact numbers should they need to contact anyone while on campus. Alongside this, the user can

select either Settings from the top right (cog icon), or one of Home, Tour, Map, and Scan options from the bottom navigation bar.

Desktop – Because majority of the features are to be used via mobile only, the options for the desktop version differ from that of mobile. On desktop, the user is given some statistics of campus tree data in a simple graphical format. In terms of options, the user can select Home, Tour (leads to information about mobile version), Map, or Settings from the top bar.

5.2.3 Tour Page

On mobile, selecting either the 'BEGIN THE TOUR' button (Fig. 4) or the tree icon on the nav bar takes a user to the Tour page. This page is designed to be used in conjunction with the physical walking-tour implemented on the Heriot-Watt Edinburgh campus. For this reason, an interactive map fills majority of the page, minimising distraction and confusion by making the primary function the largest feature (Fig. 6). The tour is focussed on trees and hence, the location markers identifiably resemble trees. The user's current location is red to create contrast and allow it to stand out in the scene. The arrow indicates the user's orientation (See Section 5.7).

Desktop – Because the Tour is mobile only, the desktop page advertises the mobile version and notifies the user about the downloadable PWA features (Fig. 7).

Mobile – From this page the user is now undertaking the tour. They can view their location, see the route, and interact with the trees on campus. The map options are generated by the Google Maps JavaScript API (Section 5.6).

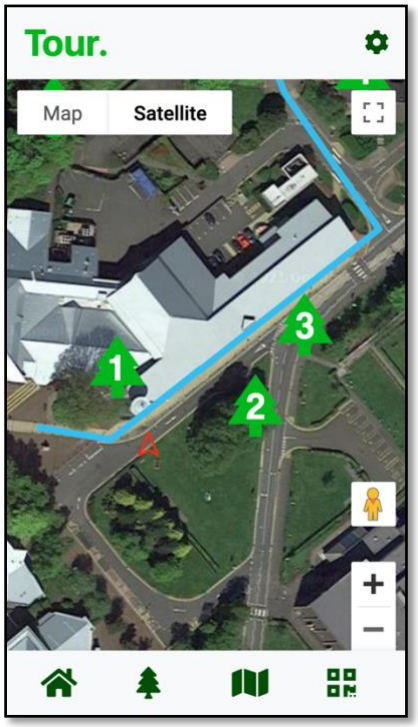


Figure 6 – Mobile Tour Page

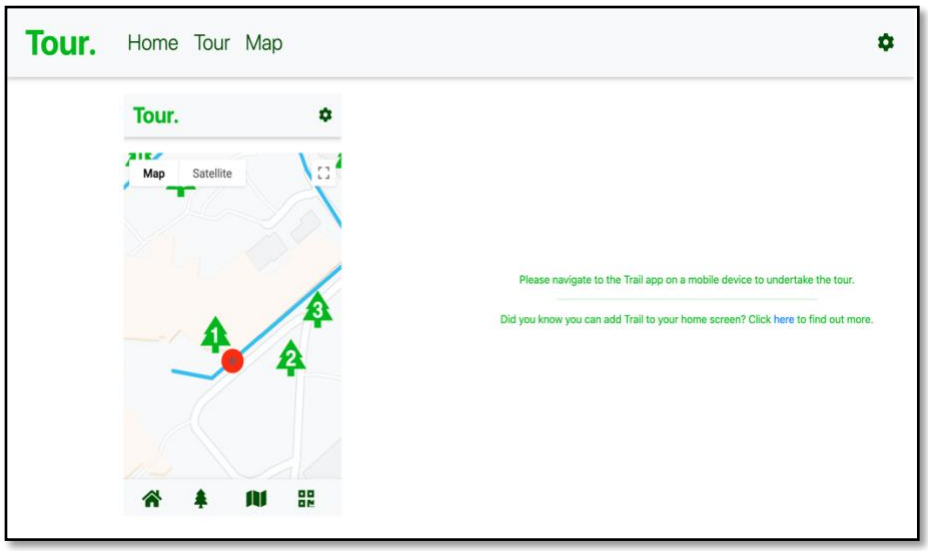


Figure 7 - Desktop Tour Page

5.2.4 Tree Information Box

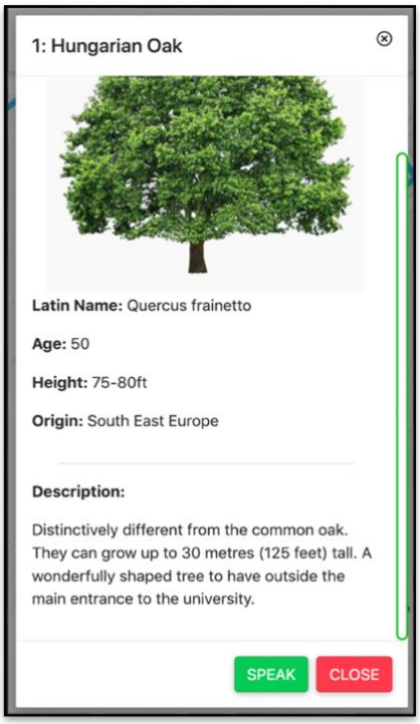


Figure 8 - Tree Information Box

Once a user triggers a tree - either by geofence, QR, or UI - an information box detailing specific information about that tree is displayed to them (Fig. 8). A user has the ability to then hear the information read aloud using speech synthesis (green SPEAK button) or close the box (red button or X in top right).

5.2.5 Map Page

The Map option from the menu bar shows the user an additional dataset of tree locations across Edinburgh (Fig. 9 and 10). Clustering was used to handle and organize the large (14,000+) number of trees (further discussed in 5.6 Generating Map Data). Campus trees are also displayed.

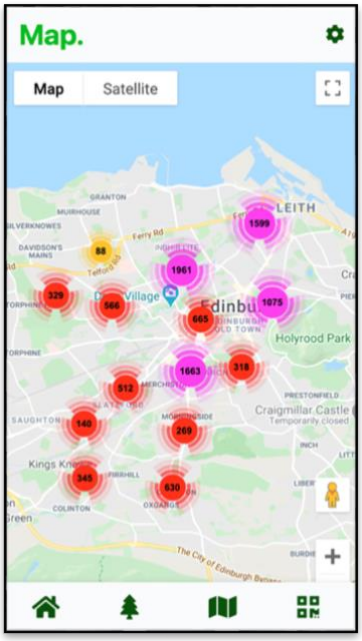


Figure 9 - Mobile Map Page

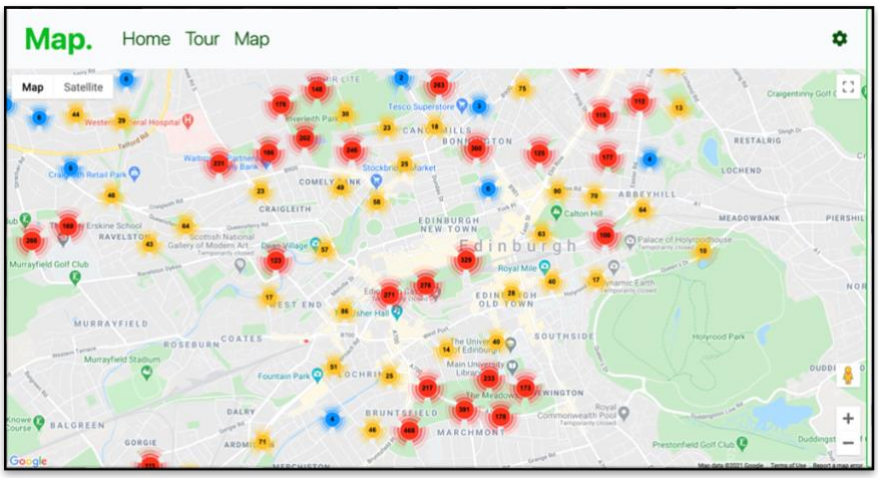


Figure 10 - Desktop Map Page

5.2.6 Scan Page

Upon navigating to the Scan page, a user can choose to either scan a QR code or try Augmented Reality (Fig. 11). A short instruction set is included below the buttons. When a user begins scanning a

QR code, a camera window appears and the 'Start Scanning' button turns red, allowing the user to stop the scanning process (Fig. 12). Selecting AR triggers a full screen camera window (Fig. 13).

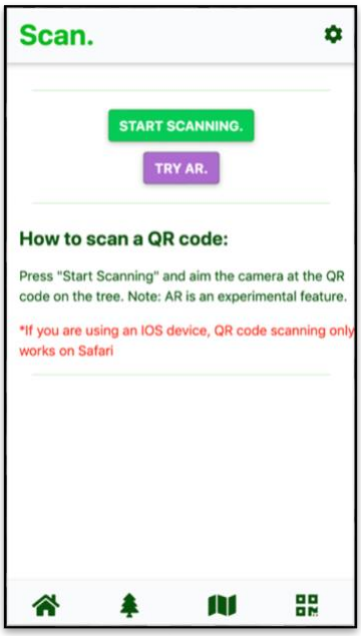


Figure 11 - Scan Page

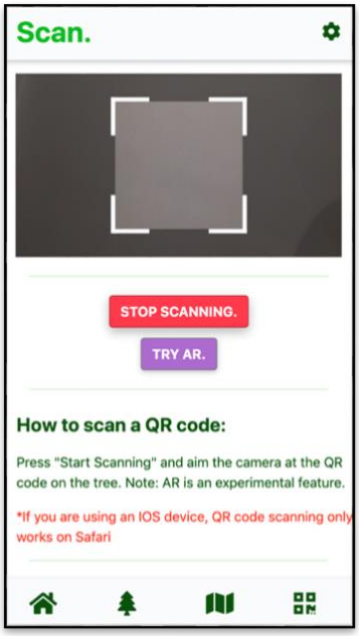


Figure 12 - QR Scanner

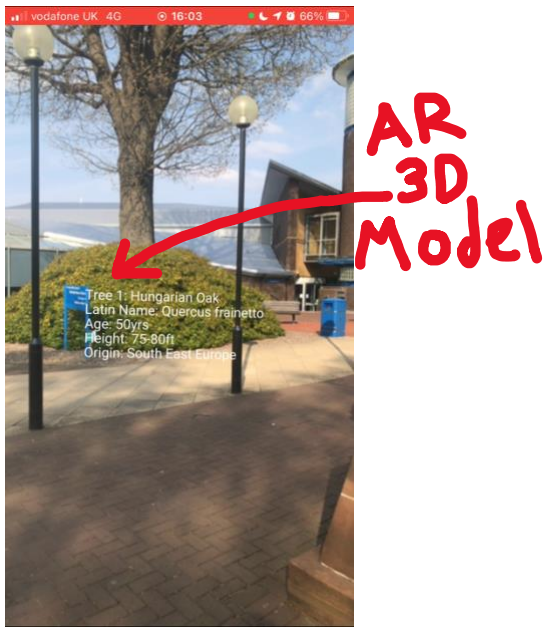


Figure 67 - AR Camera

5.3 Progressive Web-App (PWA) Development

One of the primary development aspects of Trail was ensuring that it was a cross-compatible, downloadable application. To create a progressive web-app that runs locally, a series of strict requirements must be met. Each browser has minor differences in requirements, this application followed Google Chrome's Install Criteria. These requirements were highlighted in Table 5 of Section 2.9 of the Literature review.

5.3.1 Method

This section explains how the PWA was implemented in order to meet the criteria stating the app must be served over HTTPS, contain a Web Manifest File, and have a Service Worker with offline capabilities.

5.3.1.1 Served Over HTTPS

A custom domain was purchased for the application. Heroku, the service the application is hosted on, provides SSL certification to any user on a paid plan. This certificate is required for HTTPS to enable HTML5 geolocation and other features. The minimum paid plan sits at $7 per month and was required to install the PostGIS extension to the PostgreSQL database. Heroku's SSL certificate set-up procedure is highlighted in Appendix 2.3.

Once the certification had been obtained and configured in Heroku, the domain name manager

needed to be accessed. Inside the domain name manager (NameCheap.com) all traffic going to www.thetrailapp.com needed to be redirected to https://www.thetrailapp.com. This can be seen in Appendix 2.4.

5.3.1.2 WebManifest File

A WebManifest file was created based on a template outlined by Google Developers at Web.dev (Google, 2018). The list of minimum manifest file requirements can be seen in Appendix 2.5.

The key requirements of a manifest file, as listen in Appendix 1.8 consist of; the name for the application, a start URL where the application launches from, two home-screen icons (at least) at 192px and 512px to accommodate multiple platforms, and a display option such as *fullscreen*. Other optional aspects consist of a background colour and a theme colour.

Fig.14 below shows the Trail application's Web Manifest file (See Appendix 3.3 for version with icons):

```
{
  "name": "Trail.",
  "short_name": "Trail.",
  "theme_color": "#00b800",
  "background_color": "#ffffff",
  "display": "standalone",
  "orientation": "portrait",
  "Scope": "/",
  "start_url": "https://www.thetrailapp.com/",
  "icons": [
    {
```

Figure 14 - WebManifest File (short)

5.3.1.3 Service Worker/Offline Capabilities

A Service Worker is a JavaScript file running in the background of your browser that caches files, retrieves files from the cache, and intercepts network requests going through your web application. The *fetch* handler serves files from the cache by intercepting network requests when offline.

There are 3 key steps in creating a functioning service worker (Kinlan, 2020):

1. **Registering the service worker:** A service worker must be initially registered to a user's website. The following code does this for the Trail Application (Fig. 15):

```
if ('serviceWorker' in navigator) { //if service workers are suported in the browser then do the following
    navigator.serviceWorker.register('sw.js', {scope: '/'})//registers the service worker file called sw.js and assigns scope over the root directory
    .then(function(registration) {
        console.log('Registration successful, scope is:', registration.scope);
    })
    .catch(function(error) {
        console.log('Service worker registration failed, error:', error);
```

Figure 15 - Registering a Service Worker

2. **Cache static files:** This is the preliminary step for getting the application to work offline. Files listed by the user get added to the cache memory, in step 3 the cache memory will be used to fetch files locally.

3. **Fetching static files from the cache when offline:** Once static files have been stored in the cache memory, a fetch function can be written to fetch files from the cache memory when offline. Fetch functions can either be Cache first or Network first – this choice dictates whether the service worker uses the network by default and then falls back to the cache or vice versa.
   Source-code for caching and fetching files as discussed above can be seen in Appendix 3.4.

5.3.2 Testing

Lighthouse PWA Test: Google chrome has a built-in feature called 'Lighthouse' which is used to test a website's performance, accessibility, and other features. Upon executing the PWA test (used for determining whether the application meets PWA requirements etc..), Trail passed the test and was validated as an installable Progressive Web Application. See Appendix 2.6 for the lighthouse results.

Redirect External Link to Installed PWA: A test was undertaken to determine whether an external link would redirect the user to the installed PWA or simply the webpage on their browser. The *expected results* were that the URL would redirect to the PWA. However, the *actual results* yielded that on Android devices the PWA is automatically launched, while on IOS the browser is opened. I.e., iOS does not support this feature.

5.4 Sourcing User Location Data

Once a successful PWA is created, the next foundational puzzle piece is to obtain the user's current location. The user's location is a key component in creating both an accurate, dynamic map interface and an effective geofencing system. This implementation meets Functional Requirement 2 (AFR2) stating the application must obtain a user's current location and Functional Requirement 3 (AFR3) stating that the user's current location must be displayed on the map interface.

5.4.2 Method

There are two aspects in regard to a user's current location: sourcing and displaying the current location, and tracking/updating that location as a user moves.

To obtain the user's location, the HTML5 Geolocation API was used. This API is a standard across web development and is discussed thoroughly in Section 2.8.1. The following code checks whether the API is supported in the user's browser (Fig. 16).

```
//checks if geolocation is enabled and supported in the users browser
const getCurrentPosition = ({ onSuccess, onError = () => { } }) => {
    if ('geolocation' in navigator === false) {
        return onError(new Error('Sorry, Geolocation is not supported by your browser.'));
    }
    //if the api is supported then call the method to get the location
    return navigator.geolocation.getCurrentPosition(onSuccess, onError);
```

Figure 16 - Check if Geolocation is Enabled

If geolocation is supported, then it calls the method *getCurrentPosition*, which then obtains the latitude and longitude of the user, marks that point on the map using the Google Maps JS API, and pans to that section of the map. Because Trail is a tour guide application, the user's movement needs to be taken into account. The following code shows how a user's location is tracked (Fig. 17). The location is then displayed on the map via the same method as used for getting the current location.

```
//the "watchposition" method that updates every time a user's location changes
const trackLocation = ({ onSuccess, onError = () => { } }) => {
    if ('geolocation' in navigator === false) {
        return onError(new Error('Sorry, Geolocation is not supported by your browser.'));
    }
    return navigator.geolocation.watchPosition(onSuccess, onError, {
        enableHighAccuracy: true,
        timeout: 10000,
        maximumAge: 0
});
```

Figure 17 - Track Location Source Code

5.4.3 Testing

5.4.3.1 Behaviour Across Devices

A test was developed to analyse whether the tracking of location differed in speed and effectiveness between devices. Meticulous tests were taken with both PWA and browser versions of the application using an iPhone 7 Plus (2016) and a Samsung A51 (2019). The tracked location of the device was repeatedly stored in a database along with a date/time stamp. Sample raw result data can be seen in Appendix 2.1. The Golang code to add this information to a database can be seen in Appendix 3.5. By performing some simple calculations, it was possible to deduce the average time in which one location update occurs. The test results yielded as follows (Table 12):

Table 11 - Cross-Device Geolocation Update Tests

Device	Type	Average Update (seconds)
iPhone 7 Plus	Browser (Safari)	1.045
iPhone 7 Plus	PWA	1.117
Samsung A51	Browser (Samsung)	0.549
Samsung A51	PWA	0.631

Samsung demonstrated significantly better results than the iPhone, averaging (across PWA and browser) at one location update being triggered every 0.59 seconds compared to 1.081 seconds by the iPhone. A possible explanation of this could be due to the Samsung device being a newer piece of technology than the iPhone 7 Plus, or the device could have recorded some duplicate data .

5.4.3.2 Noise and Interference

As discussed in Section 2.8.2 it is possible for GNSS to obtain a location accurate up to 3 metres, however, there are issues with the consistency of this accuracy due to interference caused by a variety of factors. To test the accuracy of the implemented system, a stationary location was recorded for 10 seconds, this test was performed in both urban (Fig. 18) and rural (Fig. 19) environments. These results were stored with a time stamp similar to the previous test.

Urban:

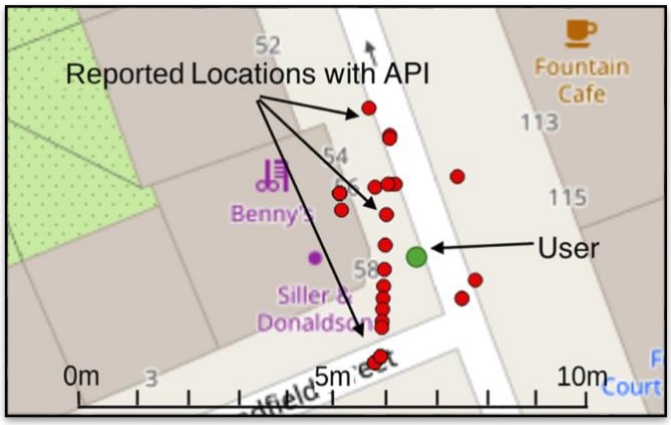


Figure 18 - Urban Positional Accuracy Test

Rural:

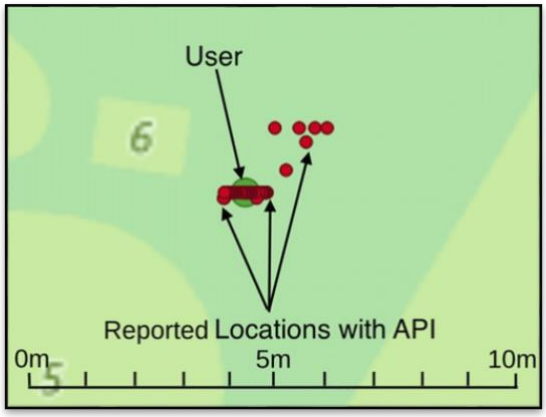


Figure 19 – Rural Positional Accuracy Test

This test showed that location accuracy is better in rural environments than urban environments due to the lack of buildings, and other obstructions that could cause interference. Fortunately, Heriot-Watt Campus is a rural area, so the location accuracy *should* remain solid across the tour route.

5.5 Human-Computer Interaction

Human-computer interaction is a fundamental aspect in developing a successful application, especially when using a variety of technologies. The primary goal of the Trail application is to allow a user to learn about trees on the HWU campus via a range of standard and novel methods. These methods have been split into two segments, push and pull. Push refers to data that is pushed to the user by the system(i.e., the user does not actively pursue the retrieval of said data), this includes spatial triggering, speech synthesis, and haptic feedback. On the contrary, pull refers to data that is pulled from the system by the user, this includes QR code scanning, AR, and interactions with the UI.

The implementation discussed in this section achieves functional requirements AFR4 – The user must be able to view data about trees on the HWU campus, AFR5 – The application must allow for QR technologies, AFR6 – The user should be able to scan a QR code using the app, and AFR7 – the application should notify a user when they are near a tree. It also achieves non-functional requirements NFR6 – The application could notify a user via haptic feedback alerts, and NFR8 – The application should have special accessibility features.

5.5.1 PUSH – Pushing Data to The User

Pushing data to the user creates the main functionality of the tour guide. The pushing system consists of four primary components; spatial triggering to trigger a tree, text to speech synthesis to speak tree data to the user, haptic feedback to alert the user, and text/image display of tree data. While these features work together, the foundation of the push system is the spatial triggering capabilities.

5.5.1.1 Spatial Triggering

To implement a spatial triggering system, a trigger zone known as a geofence first had to be created. These geofences were discussed in Section 2.4.1 of the literature review and are implemented using PostGIS – a spatial extension to the PostgreSQL database.

There are a variety of methods for creating geofences, the method found to be most intuitive was to create a virtual circle around the central point of a tree. In order to achieve this, a "geofence column" was added to the database's tree data table. This column was called "*polygon"* and is of type *GEOGRAPHY*. The geography type is provided by the PostGIS extension and allows the creation of a geographical point, specifically a point on the Earth's surface denoted by latitudes and longitudes. It was for this reason the *GEOGRAPHY* type was chosen over the *GEOMETRY* type - which assumes all data lives on a cartesian plane rather than the Earth's surface.

The column could then be populated with a geofence using the command ***ST_Buffer****(geometry g1, float radius_of_buffer)* where *g1* is the latitude and longitude of the tree and *radius_of_buffer* is the radius size of the geofence. The following line shows the creation of a 6 metre geofence around a tree:

```
UPDATE trees SET longitude=-3.2138, latitude=55.9406, radius=3,
polygon=ST_Buffer(geography(ST_POINT(-3.2138, 55.9406)), 3) WHERE id=1;
```

To check if the user is inside a geofence the **ST_DWithin**(geometry g1, geometry g2, double precision distance_of_srid) command can be used, where g2 is the user's location, g1 is the geofence, and *distance_of_srid* is the circle radius. This is executed in the back end of the system, and so, the primary task to overcome was how the user's location (obtained at the front-end) could be continuously checked in the back end. To solve this, an Ajax POST request was used inside a function called *checkGeofence().* This is called every three seconds when the user's current location is obtained. The function executes a post request to a special URL, where the latitude and longitude of the user is passed in as arguments. The result of the POST request is parsed and can be indexed into to get the returned tree information. *CheckGeofence()* is shown below (Fig. 20):

```
lat=position.coords.latitude;
lng=position.coords.longitude;
var i = 0;
if (!qr){
$.ajax({
            url: "https://www.thetrailapp.com/geofence/" + lat + "/" + lng + "/"
            method: "POST",
            contentType:'application/json&apos',
            dataType: 'json',
            cache: false,
            scriptCharset: 'utf-8',
        }). done (function (jdata) {
           var data = $.parseJSON(jdata);
```

Figure 20 - checkGeofence() function

In the back end, the Golang code uses a POST route handler to handle the post request to the specified URL. This URL is then parsed, and the location is checked with ST_DWithin in the query:

```
SELECT id, treename, latinname, height, age, description, origin, img FROM trees WHERE
ST_DWithin ( geography (ST_Point(longitude,latitude)), geography (ST_Point($1, $2)), 20)
```

The source code in Appendix 3.6 executes this query and sends the results to the front-end as JSON. The returned JSON is passed to a function called *geofence()* that displays the tree data to the user in a pop-up information window (as seen in Section 5.2.4).

An initial test of the triggering mechanism was undertaken on a 60-metre geofence set up around a nearby lamppost. Upon entering the geofence the tree information displayed correctly, however, every three seconds (user's location being checked) the pop-up box would close and re-load. This effect is not desirable, as the information a user is reading will disappear and re-load.

To combat this issue an array called "*visited*" was created and initially set to be empty. When a geofence is triggered, the system checks if the array contains the triggered tree ID, if it doesn't, then it is added to the array and the tree data is displayed. In the case that the array does contain the ID, it means the user is still inside the same geofence and the data is not re-displayed. This code can be seen here (Fig. 21):

```
visited = data.id;
//the following code uses an array to make sure multiple trees are never triggered twice - if the id (
//exists in the array then do not show it, otherwise add it to the array and show the data
if(!visited_array.includes(data.id)){
    visited_array.push(data.id);
    geofence(data);
    autoSpeech(data);
}else{
    console.log("Not in Geofence");
```

Figure 21 - Double Trigger Prevention Code

When a user exits the tour (i.e., leaves the tour page) the visited array data is cleared, this allows for users to undertake the tour multiple times over any period.

This *"visited"* method was tested with one geofence and was successful as the tree did not trigger twice, but in practice a user will be interacting with more than one tree and hence more than one geofence. Section 6.4.1 discusses how this issue was explored and overcome.

5.5.1.2 Haptic Feedback

Once trees could be successfully triggered, the user needed to be alerted when entering a geofence. Haptic feedback (vibration) was used in tandem with the triggering of trees to create a more connected experience, adding a physical response to a user's actions. In order to create this, the HTML5 Vibration API was used as discussed in Section 2.6.2. This API was lightweight and required only the following line of code to implement vibration: `window.navigator.vibrate(3000);`

The code above triggers a vibration of the device for 3 seconds. This length of vibration time was chosen so that a user could differentiate between triggering a tree with the application and getting a text from their device (which as discussed in Section 2.6.2 is typically a shorter vibration).

The vibration was implemented so that it triggered when a geofence was entered, in tandem with the tree information being displayed. Initially, this was tested on an Android device and

functioned correctly, however, upon performing the same test on an iOS device the vibration failed to execute. The following error was displayed in the console:

```
[Error] TypeError: navigator.vibrate is not a function. (In
'navigator.vibrate(3000)', 'navigator.vibrate' is undefined)
```

This error exists because the vibration API has no support on iOS devices. This can cause the Tour page on iOS to become unresponsive. Using an OS detection library this issue could be overcome, the solution can be seen in Section 6.4.2.

5.5.1.3 Speech Synthesis

The haptic feedback system was implemented to notify a user when a tree was triggered without them needing to look at their device, however, when the tree information is displayed, the user will look at their device again regardless. Some feature needed to be put in place that allows a user to learn about these trees while immersed. The solution to this problem is text-to-speech synthesis.

Text-to-speech synthesis allows for tree information stored as string values to be read aloud to the user, this works in tandem with the current text/image display and promotes immersion while allowing further accessibility to the Trail application for users who are visually impaired.

This speech synthesis feature was implemented using the experimental HTML5 Speech Synthesis API via the following function (Fig. 22) called *autoSpeech().* This is called when a geofence is triggered, the tree json data is then passed into this function where it is then read aloud.
(the *'speech'* variable contains what is to be spoken).

```
function autoSpeech(json){
    if (visited != 0){
        var id = json.id;
        var name = json.name;
        var latinname = json.latinname;
        var height = json.height;
        var age = json.age;
        var description = json.description;
        var origin = json.origin;
        if('speechSynthesis' in window){//if speech synth is supported then create a new Utterance string
            var speech = new SpeechSynthesisUtterance('You are approaching tree number' + id + '. This ' + latinname + '. commonly known as the'
            + name + '. is ' + height + ' feet tall. ' + age + 'years old. and originates from ' + origin + '.' + description);
            speech.lang = 'en-GB';//set the speech voice/language
            window.speechSynthesis.speak(speech);//speak the string defined above at default pace
```

Figure 22 - Speech Synthesis AutoSpeech()

5.5.2 PULL – How the User Pulls Data from the System

The pull system supplements the push system by allowing the user freedom to trigger and view tree information whenever they desire. This system is made up of features like QR code scanning, AR exploration, and UI interaction. These features do not need to work together in tandem and hence, the pull system is more disjointed and independent than the push. The primary feature of the pull system allows users to scan a QR code (located on a tree) in order to trigger the tree information box.

5.5.2.1 QR Code Scanner

Scanning a QR code via the Trail application was a primary goal of the system as it was to be implemented both in the thesis version of Trail, and the version that will be created for long term use on campus. The reason behind choosing QR codes as a main interaction method was due to their size, customizability, familiarity (as a result of Covid-19), and the ability to scan them using native camera apps without prior knowledge of the Trail application - creating a more inclusive environment which also leads to the discovery of the app.

As proposed in Section 2.4.2.3 the scanner was implemented using the *html5-qrcode* JavaScript library. The library was imported and the core scanning and handling of QR results was programmed. The full source code that powers the QR system (front-end and backend) can be seen in Appendix 3.7. Once the scanning system was in place the QR codes were created using qrcode-monkey.com (Image of QR code is shown in Appendix 2.8) The URL embedded in the QR code redirects a user to the tour page, passing a variable corresponding to the ID of the scanned tree within it. This URL is handled by Golang at the back end where a query to the database returns the tree data to the user.

To ensure the robustness of the QR system a variety of tests were carried out. The initial test discovered that a small bug occurs when scanning a QR code while inside a geofence. The actual scanning works perfectly, redirecting to the tour page where tree data is displayed, however, as soon as the user allows location permissions on the page, the geofence is triggered and overwrites the tree data box. While this may be a bug, it does not pose any significant issues as the geofence is linked to the same tree as the QR code, hence, the information boxes are the same. However, it cannot be ensured with 100% certainty that this bug may not cause some issues in unforeseen use cases.

Another test was conducted to find out what happens when a user scans a QR code from their native camera app when they have the Trail PWA already installed. Upon completion of this test, results yielded that on Android devices the QR link will open with the installed application while on iOS devices the link will open in a browser window. This is due to iOS' lesser support for Progressive Web Applications (this issue was similarly expressed previously in 5.3.2).

5.5.2.2 Augmented Reality

AR is a potential alternative to scanning a QR code and could provide a more immersive experience for a user. Augmented reality features were experimented with through the use of AR.js, an augmented reality web framework (AR.js org, 2021). The type of AR that was chosen was "Location Based AR" and allows for a model to be rendered through the AR viewer at a coordinate location. In the case of the Trail app, the model was tree data, and the location a tree. The source code (Fig. 23) highlights the creation, and location placement of an AR entity that the user can interact with (for full AR code see Appendix 3.8), the user-end product of this code was shown in UI Section 5.2.6.

```
<a-entity
    gltf-model="/static/ar/tree5.glb"
    look-at="[gps-camera]"
    scale="0.25 0.25 0.25"
    gps-entity-place="latitude: 55.9406; longitude: -3.2138;"
></a-entity>
```

Figure 23 - Augmented Reality Entity

To test the effectiveness of AR for pulling tree data, it was compared against the QR system. A QR code was created for "Tree 5" matching the data of the AR entity shown above. First the QR code was scanned to retrieve the data, the outcome was successful with no issues arising. Secondly, the tree data was viewed using the AR camera, the outcome was successful, however the 3D model initially failed to display correctly, and the page had to be refreshed to fix this issue (analysis in Section 6.5.2).

5.5.2.3 User Interface Interaction

To provide a more intuitive method of interaction than QR/AR, and the final feature of the pull system is standard UI interaction. Both the map and the tour pages allow a user to click on any tree on the map to view its corresponding information. This feature was implemented by creating an event listener on the map data that constantly senses for a click. The ability to add a listener to the map data is a feature provided by the Google Maps API which is discussed in the following section (5.6). When a click event is triggered, the information of the clicked-on tree is pulled from a geojson file that stores all of the map layer tree data - this method was found to be simpler than performing a call to the database. The data is then stored in variables and displayed to the user in the same fashion as previously (source code can be viewed in Appendix 3.9).

5.6 Generating Map Data

The ability to display an interactive map was paramount in realising the vision for the Trail application. Interactive maps play a strong role in both the *Map* and *Tour* page, providing the main visual experience for the user. The implementation discussed in this section achieved requirement AFR1 – The user must be able to interact with a map interface. Maps were generated using the Google Maps JavaScript API by defining a new map object with a specified central point (in this case Edinburgh). This map generation code is shown below (Fig. 24):

```
var map = new google.maps.Map(document.getElementById('map2'), {
    center: { lat: 55.909329, lng: -3.319491 },
    zoom: 18,
    mapTypeId: 'satellite',
```

Figure 24 - Map Generation Code

Once an interactive base map had been generated, the locations of trees needed to be marked on the map. To achieve this, geojson files containing the tree information and coordinate locations were created and imported into the map using the function map.data.loadGeoJson(*FILENAME*).

For the maps page, the geojson file was developed with Edinburgh tree data provided by Edinburgh

council (Edinburgh Council, 2018). This dataset was originally 60,000 trees strong and was significantly reduced when implemented in the system to only 10,000 for better load times. A sample of tree geojson file content can be viewed in Appendix 2.9. Once the tree data is loaded, custom tree markers are drawn on the map based on the image URL outlined in the geojson file.

5.7 Directing a User

The ability to direct a user around a tour is the final feature that was implemented in order to complete the Trail application. Initially, the idea was to develop some kind of sat-nav like system with audible direction commands such as: "in 5 metres turn left". However, it was quickly realised by discussing the idea with peers that this would be too overwhelming for the user, especially in a rural environment where the goal is to **supplement** their nature experience. The user is not a computer, they do not need to be told so explicitly which direction to turn on a walking tour which will be physically signposted. A system was created which allows the user to be less interrupted while walking through nature, but still provides enough directing to help the user navigate the tree trail. This system is comprised of three parts; an orientation arrow to show the user which direction they are facing in relation to the map (this arrow replaces the circle location marker), a route path layer on the map that highlights the pathway the user should follow, and a "Lost?" detector which notifies the user when they have left the vicinity of a tree trail.

Orientation Arrow

The development of the orientation arrow involved tacking 2 primary problems: getting the user's device orientation and rotating the location arrow to match the observed orientation. Sourcing the user's orientation required the use of the experimental Device Orientation Web API, which enables the sensing of changes in orientation. This orientation is described via 3 values, alpha, beta, and gamma, where alpha represents the motion of the device around the z axis, beta around the x axis, and gamma around the y axis (Mozilla, 2020).

```
        var alpha = null;
        //Check for iOS property
        if (os.name == "iOS" || os.name == "Mac OS") {
            alpha = event.webkitCompassHeading;
        }
        //non iOS
        else {
            alpha = event.alpha;
        }
        //takes the current icon, rotates it and assigns the new rotated one to the marker.
        var locationIcon = marker.getIcon();;
        locationIcon.rotation = 360 - alpha;
        marker.setIcon(locationIcon);
    }, false);
}else{
    alert("devic
```

Figure 180

This code snippet (Fig. 25) finds alpha and minuses it from 360 (degrees), this is the equivalent to 360° - the angle of rotation from North. The location arrow image is then rotated by this degree, resulting in the arrow facing the correct direction.

Route Path

Following the development of the orientation arrow, the route path was created in order to highlight to a user the exact path of the tree trail (Fig. 26). The primary reason for this was that due to the trail going through forest, any physical pathways are hidden by tree foliage on the satellite map view.

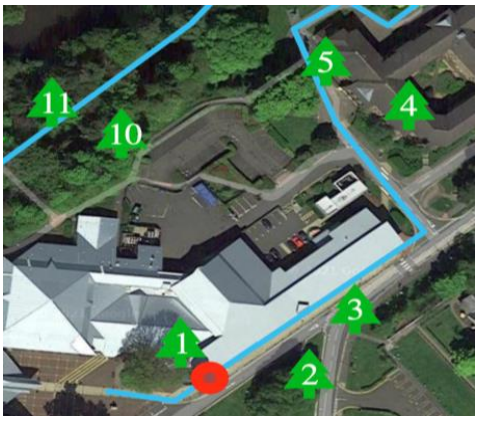


Figure 26 - Blue Route Path

The creation of a route path was done easily through the Maps API Polyline tool. Using this tool, lines can be created between two geographical points on a map and then be displayed as a separate layer to the user.

The following code demonstrates the creation of one of these lines:

```
testingRoute.push( new google.maps.LatLng(55.9407653174704, -3.2129520728739345));
```

Using an array, multiple lines could be created and joined to develop the blue route shown above (Fig. 26), full source code can be viewed in Appendix 3.10.

Lost Detector

The final feature of the direction system is in place to alert a user when they are not within the vicinity of a tree trail.

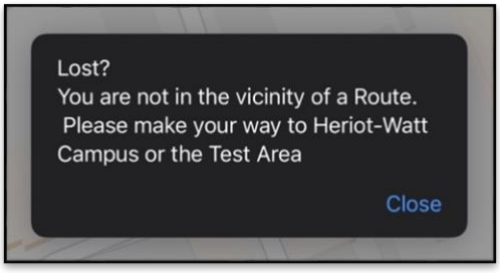


Figure 27 - Lost Detector Alert

This feature satisfies the cases where either a user becomes lost and doesn't know if they are still on the trail, or where a user attempts to undertake a tour when they are not on campus. This alert can be viewed in Figure 27.

In order to implement this feature, a very similar process to creating the tree geofences was taken. A large geofence was created around campus, with its centre point being the centre of the route and the radius set to 300m. A radius of this size is slightly larger than the tour area and was chosen so as the geofence could encapsulate the whole tour while also accommodating for any interference to the user's location. The front-end system sends a user's current location to the backend which detects if a user is out-with the geofence, if they are, then the "Lost?" alert is shown every 4 seconds until they re-enter the tour area. See Appendix 3.11 for Source Code.

Chapter 6: Issues Overcome and System Limitations

The following section discusses the limitations of the features implemented above as well as any issues that were discovered during development and if possible, how these issues were overcome.

6.1 User Interface

Issues overcome

The application must be able to detect when a user is on mobile or desktop and render the correct corresponding version. The application must also scale to fit multiple screen sizes.

To solve the first issue, two classes – *mobhide* and *deskhide* - were created in the main CSS file. These classes exist within a *@media* query, where *mobhide* hides desktop elements when the screen size is < 992px (most likely a mobile device), and *deskhide* hides mobile elements when the screen size is > 992px (most likely a desktop device). Please see Appendix 3.12for source code.

To solve the second issue, HTML's *viewport* meta tag was used. This tag is located in the *<head>* of each page and controls layout and scaling across all mobile devices via a simple set of. The implemented tag is shown below (Fig. 28):

```
<meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1.0, minimum-scale=1.0, user-scalable=no, viewport-fit=cover"/>
```

Figure 28 - Viewport Meta Tag Code

Limitations

While on average a device with a screen size < 992px is most likely a mobile device, there does exist the possibility where a user's desktop device has a small display and thus, renders the mobile page on their desktop machine and vice versa. There are also minor issues when the application is run as a PWA on the iPhone X. Due to the iPhone X's primary control being through gestures via a touch bar located at the bottom of the screen, this can mildly obstruct a user's ability to interact with the bottom navigation bar.

6.2 Progressive Web App

Issues Overcome

The Service Worker was unable to retrieve files, returning a 404 error when it attempted. This was due to the sw.js file being located in the /js folder of the file system and hence its scope was limited. This issue was fixed by moving the sw.js file to the root directory, so as it could have scope over '/'.

Limitations

Occasionally the cache does not clear automatically on the installed PWA and hence, it must be cleared manually for updates to take effect. This could result in users running an old version of the app. Furthermore, PWA install prompts only fire on Android devices. There is no support for automatic prompts on iOS and could result in a user not knowing how to install the app. To combat this, an installation guide is included via an option in settings. iOS has shown to have much less PWA support than Android, with Chrome on iOS having no PWA support at all.

Finally, a key limitation of Progressive Web-Applications is there is no capability to run the application features while a user's device is locked. Background running of the Trail application would provide the user the ability to lock their device and interact with the Tour hands free via solely speech synthesis and spatial triggering. This feature would immerse a user in their environment, similar to how Strava or Nike Run applications function. This ability has been fought for by developers arguing that geolocation could be incorporated into Service Workers (Maher, 2017), however W3C is yet to back and implement the idea.

6.3 User Positioning

Limitations

There are a few key limitations of the implemented geolocation system. As demonstrated by the tests conducted in Section 5.4.3, accuracy in urban environments can be volatile and hence the user cannot rely on the system to be 100% true in portraying their location. Furthermore, due to the application being a web-app and using the HTML5 geolocation API, if a user loses signal on their device or disconnects from the internet, geolocation simply does not function.

6.4 User Information Push

6.4.1 Spatial Triggering

Issues Overcome

A small test site was developed to analyse what would happen in a situation where trees are within overlapping distances. 30-metre geofences were created around three trees (A,B,C) no more than 10-metres apart from one another. The application was then launched and a slow walk past all the trees was performed. The tree A triggered correctly on approach, while tree B and C triggered some time past their true location. To visually understand what was happening here, the walking route was recorded along with the location when each geofence was triggered (Fig. 29).

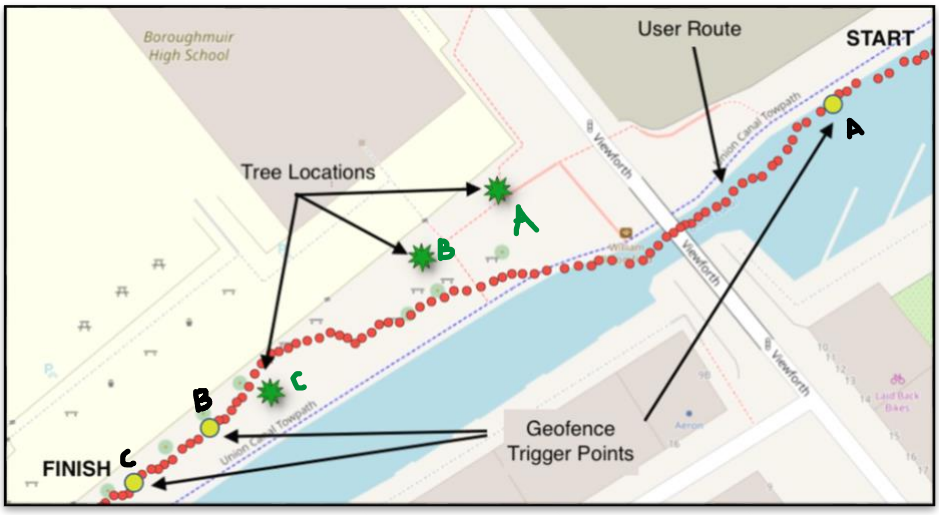


Figure 29 - Geofence Test on Three Trees

The results above show that tree B and C are not being triggered until the user exits the prior geofence they were in. This was happening because the query that checks whether a user is inside a geofence has no knowledge that a tree has already been visited (the front-end does). This means that even though the front-end won't display the data, the back end will keep returning the same tree until the user is no longer inside that geofence. This bug was fixed by passing the ID of the last viewed tree to the backend whenever a geofence is triggered. The ID is then used inside the SQL query to only return data where the id of the new tree does not equal the id of the last viewed tree. This following query now handles the overlapping geofences correctly:

```
SELECT id, treename, latinname, height, age, description, origin, img FROM trees WHERE ST_DWithin ( geography
(ST_Point(longitude,latitude)), geography (ST_Point($1, $2)), 20) AND id != $3
```

The same test was then mapped using the updated query (Fig. 30):

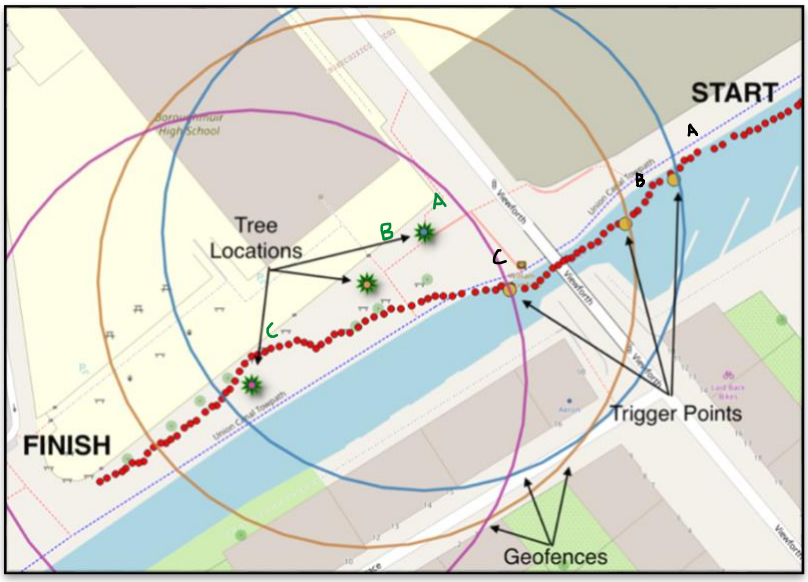


Figure 30 - Updated Query Trigger Test Result

From the mapped result above it is clear that the overlapping issue was solved however, due to the size of the geofences, all trees are being triggered a great distance before they are even seen by a user. Figure 31 below shows the final geofence system with smaller 10-metre geofences.

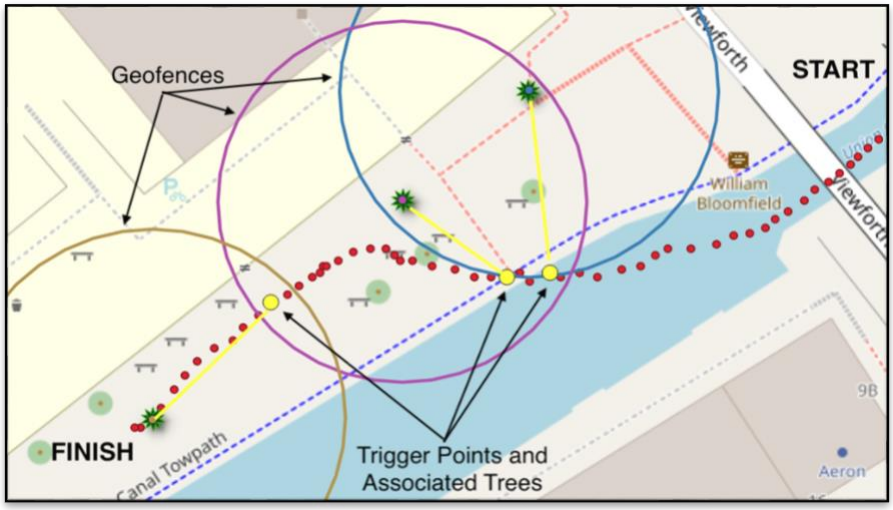


Figure 31 - Final Trigger Test - 10m Geofences

Limitations

Even with the smaller geofences, because of the volatility of location accuracy due to interference there is still scope for trees to trigger earlier, or later than they should. For example, the system could think a user is 3 meters ahead of where they actually are and trigger the tree too far in advance, furthermore, because of the array in place to prevent double triggering, they could not then spatially trigger that tree again if they miss the information - unless they refresh the tour page.

6.4.2 Haptic Feedback

Issues Overcome

Haptic feedback failed to function on iOS devices yielding an error and crashing. To combat this issue while still enabling Android functionality, there needed to be some way to only run the vibration code when a user is not on an iOS device. Luckily, there exists a JavaScript library called "*User Agent Parser*" which can identify the operating system of a user's device (Faisalman, 2021). The following code (Fig. 32) checks a user's device OS and triggers the vibration API only if they are running Android.

```
//if the user not on an IOS/Apple device call the vibrate function
if(os.name != "iOS" && os.name != "Mac OS") { navigator.vibrate(300); }
```

Figure 32 - User Device OS Checker for Haptic Feedback

6.4.3 Speech Synthesis

Issues Overcome

Like the issue discovered with haptic feedback, iOS also has little support for the speech synthesis API. Speech is unable to be triggered automatically by entering a geofence due to Apple's policy that any web media must be triggered via an interaction by the user (e.g., a button press). This policy was designed to prevent malicious websites from flooding a user with media-based advertisements.

To still address the stated accessibility issues an alternative speech trigger was implemented via a button located in the tree information popup box. This button exists on both iOS (to combat the issue) and Android (in order to allow for the re-playing of speech in the event a user misses the initial automatic playing). Fig. 33 shows the source code, and Fig. 34 the button on the tree data box:

```
function read(){
        var idName = $('#tree_title').html(); //add the contents of the data box to variables
        var age = $('#tree_age').html();
        var latinname = $('#latin_name').html();
        var height = $('#tree_height').html();
        var description = $('#tree_info').html();
        var origin = $('#tree_origin').html();
        //if the user not on an IOS/Apple device call the vibrate function
        if(os.name != "iOS" && os.name != "Mac OS") { navigator.vibrate(100); }
        if('speechSynthesis' in window){//if speech synth is supported then create a new Utter
            var speech = new SpeechSynthesisUtterance('Tree ' + idName + ', This ' + latinname
            + ', is ' + height + ' tall. ' + age + 'years old, and originates from ' + origin
            speech.lang = 'en-GB';//set the speech voice/language
            textToSpeech.speak(speech);//speak the string defined above at default pace
```

Figure 33 – Source Code for Speech Synth via Button

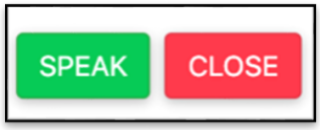


Figure 34 - Speak Button

Limitations

A key limitation arises in the event that a user traverses between trees at a fast pace. When geofences are entered in rapid succession the Speech Synthesis feature misleads the user by speaking information about e.g., tree 1, when in reality the user has already triggered tree 2. This is a result of speech taking a long time to read all of the data about a tree, and so the data for the next tree gets queued behind the current data until it's complete. As stated previously, there are also evident limitations when using an iOS device due to the lack of support for automatic execution of speech synthesis.

6.5 User Information Pull

6.5.1 QR Code Scanner

Issues Overcome

There are a variety of security vulnerabilities exposed to the system such as, when a user scans a QR code the system automatically redirects the user to the URL it holds. URLS can be malicious and redirect a user to a harmful webpage. To combat this, a common regex expression was used to make sure the scanned URL was a legitimate URL (Appendix 3.7) along with the following line of code that permits only Trail domain URLs redirect: `if (qrCodeMessage.includes('thetrailapp'))`

6.5.2 Augmented Reality

Limitations

Web AR is an experimental feature and in many cases the AR entity glitches, travelling across the screen in explosive movements. This behaviour could be a result of the AR system failing to pinpoint the coordinate location of the entity and causing it to periodically adjust its position (See Appendix 2.10 for example). Overall, AR can create a semi-functional immersive experience for learning about trees but unfortunately, due to its exhibited unreliability the QR system proves to be a better primary choice for the Trail application.

6.5.3 UI Interaction

Limitations

There is one notable issue of the UI interaction, this is on the *map* page where clustering is used. Because of clustering, the standard click event listener would not work as there is a prior event listener checking for clicks on cluster icons. To combat this, a listener was created when the tree markers are added to the map rather than after, however, this system was unable to successfully pull the geojson data and yields a *null pointer exception*. A theory as to why this occurs is that since the system clusters the tree markers it fails to see the markers as individual entities and hence, attempts to pull tree data of an object that does not exist in the geojson file. The next section goes on to explain how map data is generated and why clustering was chosen to be used despite this issue

6.6 Map Data

Issues Overcome

Accessing the *Map* page demonstrated awful load times due to the number of trees being individually rendered. To solve this issue clustering was used. Clustering is a library function that works with the Maps API, it groups nearby trees together displaying the number of trees in the group inside a coloured circle icon. The icon and number change depending on the zoom layer of the map, i.e., if a user is zoomed in to 100%, they can view individual trees or groups of two, whereas a zoom level of 50% would show clusters of 100 or so trees. This feature was shown in section 5.2.5, the source code for creating these clusters can be seen below (Fig. 34):

```
var markerCluster = new MarkerClusterer(map, markers,{
    imagePath: 'https://developers.google.com/maps/documentation/javascript/examples/markerclusterer/m'});
});
```

Figure 34 - Creating Marker Clusters

Limitations

While clustering (on the web client side) did improve the performance of the *Map* page it is still sub-optimal with load times ranging between 2 and 6 seconds depending on the age of the user's device. As discussed in the previous section the clustering also gave way to problems regarding the UI interaction with the map. It would be possible to improve performance by creating map raster tiles on the server in advance to be delivered to the user, or alternatively the number of Edinburgh trees could be reduced, however, with that the scope of the *Map* page is also reduced.

6.7 User Direction System

Limitations

The direction system provides the necessary tools in order to keep a user on track and aware of the route. Despite this, there are limitations, such as the orientation arrow not functioning on iOS devices, and the lost detector alerts having the ability to queue behind one another if they are not dismissed, leading to a cumbersome task of closing all of said alerts before resuming the tour.

Chapter 7: System Self Evaluation – Taking the Tour

Due to Covid-19, the number of study participants willing to test the physical aspects of the application was immensely small (only 1 user stated they would participate in physical testing and didn't show up). This is both due to government COVID-19 guidelines and general comfort of the participants. In order to still evaluate the main use-case of the tour (spatial triggering), a small test site was created where an 'in house' system evaluation could take place. This test site was located in central Edinburgh to prevent the unadvised use of public transport that would have occurred if one had travelled to the Trail route on Heriot-Watt Campus. Fig. 35 outlines the test site:

Figure 35 - Test Site Trees

7.1 Lighthouse Tests

Before beginning the testing procedures at the site, it was valuable to gain some statistics on the overall system rating/performance. To do this, a web tool called Lighthouse was used. Lighthouse is a community-maintained tool designed to help improve the quality of a webpage. Upon running a test, 4 separate audits are performed on that webpage: Performance, Accessibility, Best Practices, and SEO (search engine optimisation). Fig. 36 details the results of running the Lighthouse test on the Trail application:

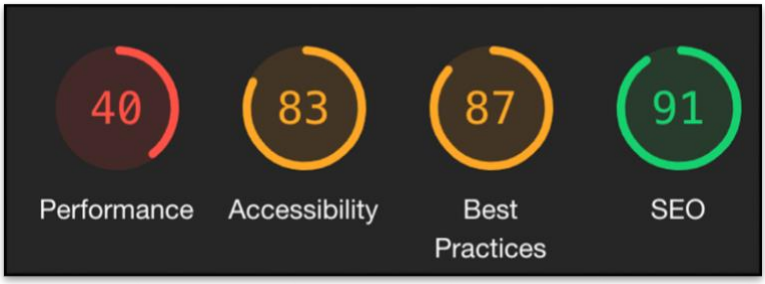


Figure 36 - Lighthouse Test Results

Accessibility, Best Practices, and SEO all yield extremely positive scores of over 80, meaning that they have been achieved well. There is a low score of 40 returned by Performance, upon inspecting the result, Lighthouse details the issues and possible solutions to the problem, stating unused JavaScript and CSS files are slowing down page load by 4.2 seconds. This is a result of using the MDB UI framework as all files that provide the features of MDB must be imported regardless of them being used. Secondly, as discussed in Section 5.6, the Edinburgh tree dataset results in much longer load times. An example of performance improvements suggested by Lighthouse is located in Appendix 4.1.

The following scenarios have further evaluated the system by performing tasks and recording the expected and actual outcomes along with possible solutions to any problems that have arisen. These tests were undertaken 3 times for consistency and executed using an Android device in order to be compatible with all Tour features.

7.2 Undertaking Tour – Normal

The Expected Outcome is that trees will trigger in the order 5,6,7. Each trigger will activate a vibration to the device along with speech synthesis initialising to tell the user about the tree. **The Actual Outcome** was as expected however there was some overlap in speech between tree 5 and 6 due to increased walking pace. **A Possible Solution** to this issue was discussed Section 6.4.3 with Speech Synthesis limitations.

7.3 Undertaking Tour – Backwards

The Expected Outcome is that Trees will trigger in the order 7,6,5. Each trigger will activate a vibration to the device along with speech synthesis initialising to tell the user about the tree. **The Actual Outcome** was as expected.

7.4 Leaving Tour Area

The Expected Outcome is that upon leaving the tour area the "Lost?" pop-up will appear telling the user that they have left the trail route. This pop-up will trigger every 4 seconds until a user re-enters a tour area. **The Actual Outcome** was as expected.

7.5 Entering the Tour Half-Way

The Expected Outcome is that tree number 6 will trigger as normal, along with the vibration and speech. Then depending on the route the user chooses, either tree 5 or tree 7 will trigger. **The Actual Outcome** was as expected.

7.6 Losing Phone Signal/Internet Connection

The Expected Outcome is that the page will fail to load and display an error stating that there is no internet connection (due to the Maps API needing internet). **The Actual Outcome** was an alert that stated "Error: Position Unavailable" and an unresponsive map. This is a result of the geolocation system failing *before* a map fetch request yields the expected error. **An Improvement** could be to add "check your internet connection" after the position alert as the alert is rather vague, stating only "Position unavailable" with no reason why. This fix will improve the clarity of the error to a user.

Chapter 8: User Evaluation

8.1 Introduction and Explanation of Tests

In order to assess the effectiveness and completeness of the implemented Trail application, a study involving 13 participants was carried out between the 22nd of March and the 12th of April 2021. This section describes the evaluation study for the Trail application, where users were given a variety of tasks to test application features. These tasks were paired with a series of online questionnaires that were completed pre-test and post-test to obtain important information such as user demographics and both qualitative and quantitative opinions. The results of the questionnaires are analysed in depth and relevant conclusions are drawn.

To recruit participants, a survey was sent out over social media platforms. Participants who registered interest were sent consent forms via email, or direct message. This consent form outlined the purpose of the study, personal data being collected, and GDPR guidelines. For consent form, see Appendix 4.2. All participants were given a participant ID to keep track of forms and maintain pseudo-anonymity. Once recruited, a pre-questionnaire is sent to gather demographics data. This questionnaire consists of quantitative numerical questions (e.g., age) and quantitative binary questions (e.g., YES/NO). While this data will be anonymised, it will provide valuable insights into the overall demographics of participants. An example questionnaire can be seen in Appendix 4.3.

The participant then undertakes the evaluation tasks along with a post-questionnaire via an online survey. These questions are made up of binary systems, text based open-ended questions, and Likert scale responses where a user can agree with a statement to varying degrees. These surveys were designed in accordance with best practices outlined by Survey Monkey. These best practices state to focus on closed-ended, unbiased questions that are not repeated. Answers should be optional, and Likert scales should have a balanced range of choices (Monkey, 2021). Example evaluation tasks and post-task questions can be seen in Appendix 4.4.

8.2 Updated Strategy Due to Covid-19

Initially, the evaluation strategy was orchestrated to test physical aspects of the application by inviting a participant to the HWU Campus, whereby they could undertake the complete on-site tour. However, as a result of the ongoing Covid-19 pandemic this original strategy is no longer viable. Current Covid-19 guidelines state that the population should not take public transport and avoid travelling completely unless deemed essential, hence preventing the on-campus evaluation.

To combat this issue, the physical aspects of the application will be evaluated at the test site created for Chapter 7: Self System Evaluation. This test site is located in the centre of Edinburgh and should be

a walkable distance for test participants. In order to promote safety for participants, the physical tour is optional. The initial demographics survey document was updated to include the following questions: "*Would you feel comfortable testing the physical aspects of the application? (yes/no)"* and "*If yes, are you within walking distance from the test site (marked on map – See Appendix 2.11 for map) located along the canal in Fountainbridge Green EH11 1BF ? (yes/no)".* If a user stated "*yes*" to both of those questions they were sent a set of physical testing tasks along with the documents outlined in the previous section. This questionnaire can be seen in Appendix 4.5.

8.3 Results

8.3.1 Pre-Questionnaire (demographics) Analysis

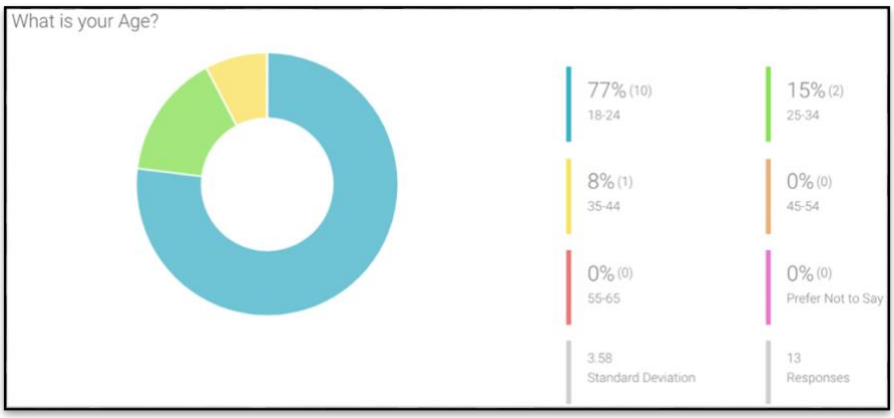


Figure 37 - Age Demographics Result (Pie Chart)

Fig. 37 above shows the majority of participants were between the ages of 18 and 24 with 77% of users falling under this category. Age range increases marginally, capping at the 35-44 age bracket and hence, there is no data recorded to analyse the behaviour of users over 45 years old. This is an important factor as the younger generation are generally more familiar with using modern/emerging technologies and could mean the obtained results are biased. Despite only having one participant in the 34-44 age range, it was seen from their evaluation that they did indeed yield more negative responses, however, due to the singular nature of this event it cannot be proven with significance.

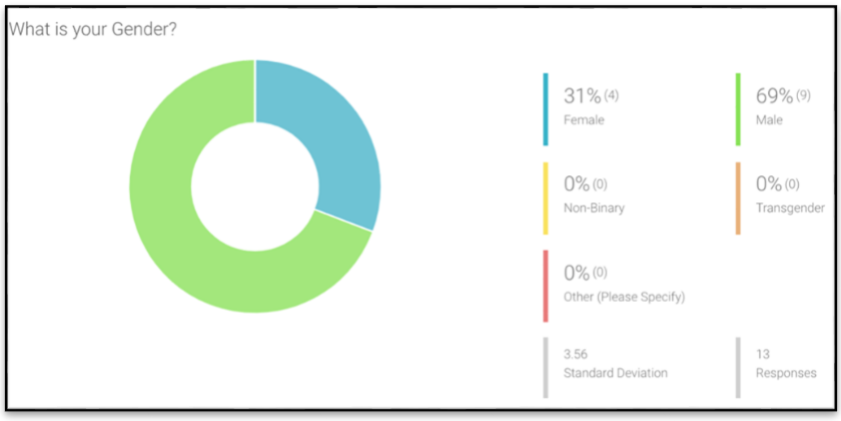


Figure 38 - Gender Results (Pie Chart)

Fig. 38 (left) demonstrates the gender split between participants in the study with 69% male to 31% female. It would have been desirable to have had a more even split in order to gather a more equal, accurate results.

Fig. 39 below describes the device (iOS or Android) that the participant used while undertaking the evaluation study. Looking at the results it is evident that majority (83%) of participants were using an iPhone (iOS) device. Based on the testing carried out in prior sections, iOS was shown to have limited support for some features of the PWA and hence, some of the results reflect this - namely when the user was asked to add the application to their home screen.

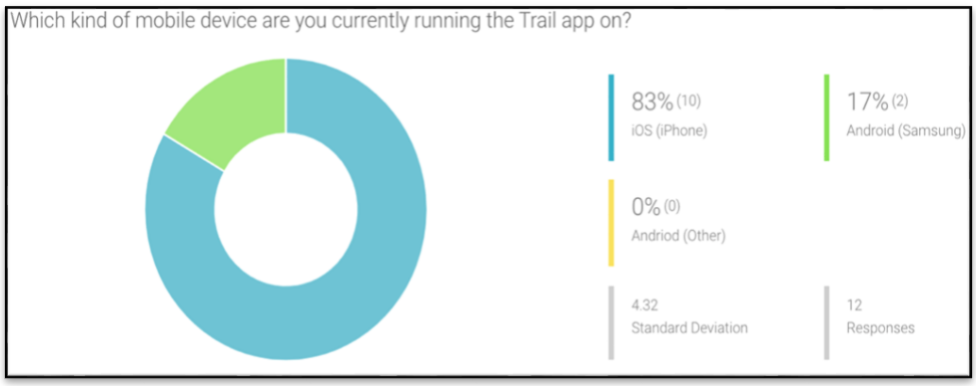


Figure 39 - User Device Results (Pie Chart)

Furthermore, Fig. 40 continues to explain user device demographics by outlining the browser that participants primarily use on their device. 66% of the Android users chose “Samsung Internet Browser” whereases 70% of iOS users chose Safari. It is clear from this result that regardless of the device, the user’s browser preference leans towards the default browser installed on their device. This is beneficial when regarding iOS devices as it was demonstrated that the “Chrome” app on iOS has the least support overall for progressive web apps (23% of participants still primarily use iOS Chrome).

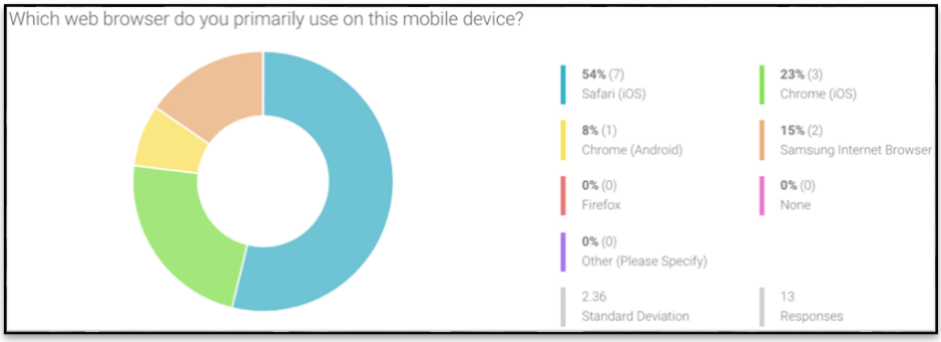


Figure 40 - User Browser Results (Pie Chart)

Fig. 41 and 42 detail the results of the Covid-19 mitigation that was put in place, requesting if users would be comfortable participating in the physical aspects of the study. According to the results over 60% of participants (Fig. 41) stated that they were comfortable to undertake physical aspects of the tour, however, 92% of these participants (Fig. 42) were not within walking distance of the test site and hence due to Covid-19 guidelines at that time, unfortunately could not undertake that segment of the study. It would have been greatly desirable to have obtained more physical test participants in order to fully evaluate those aspects of the system by a third party.

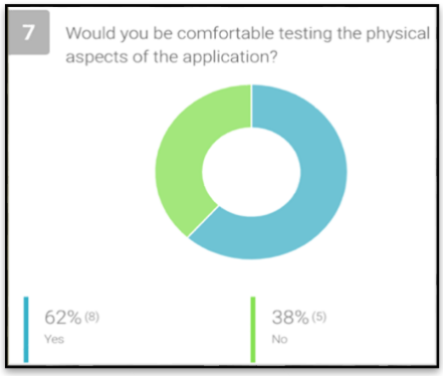


Figure 41 - User Physical Testing Check (Pie Chart)

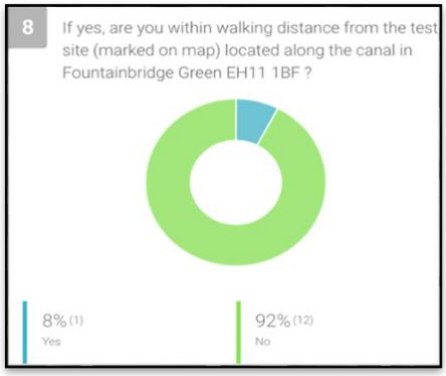


Figure 42 - Walking Distance Test Site (Pie Chart)

8.3.2 Evaluation Task Question Analysis

The following section discusses the outcomes of the evaluation tasks and analyses any comments or criticism given by participants. Fig. 43 below shows the Likert Scale results of task 1, where a user was asked to scan a tree QR code using the application's QR code scanner. The results are extremely positive with 77% scoring the experience a 9 or 10 for "easiness".

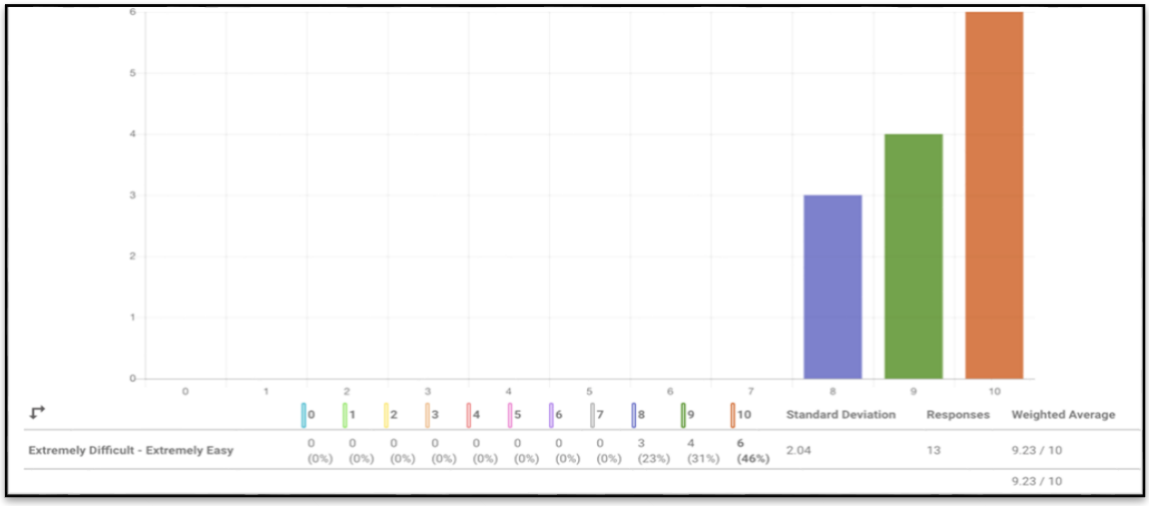


Figure 43 - QR Code Scanning Likert Scale Result (Bar Chart)

It is valuable to note that 23% of participants rated the experience at an 8 in terms of the ease of use. This was primarily due to some users not being able to identify the QR code logo on the nav bar and hence, it took them longer to figure out how to find the scanner. Some optional text responses to the QR scanning experience are shown below (Full responses shown in Appendix 4.6):

- ***"Very easy to use, fast response"***
- ***"I didn't know what to look for, I wouldn't have known what the QR icon looked like"***
- ***"Extremely easy to find and navigate"***

The second task in the evaluation study required a user to navigate to the "Map" page and locate "Tree 1" on the Heriot-Watt Campus. The Likert Scale results are displayed in Fig. 44 below, stating the "easiness" of the task via a rating from 1-10.

Upon analysing the bar chart one can see that the results are much more spread than the prior task, with the two highest rated points on the scale being 7 (31%) and 10 (31%). While the average rating sits at 7.97 there is one response of 1, stating that the experience was extremely difficult.

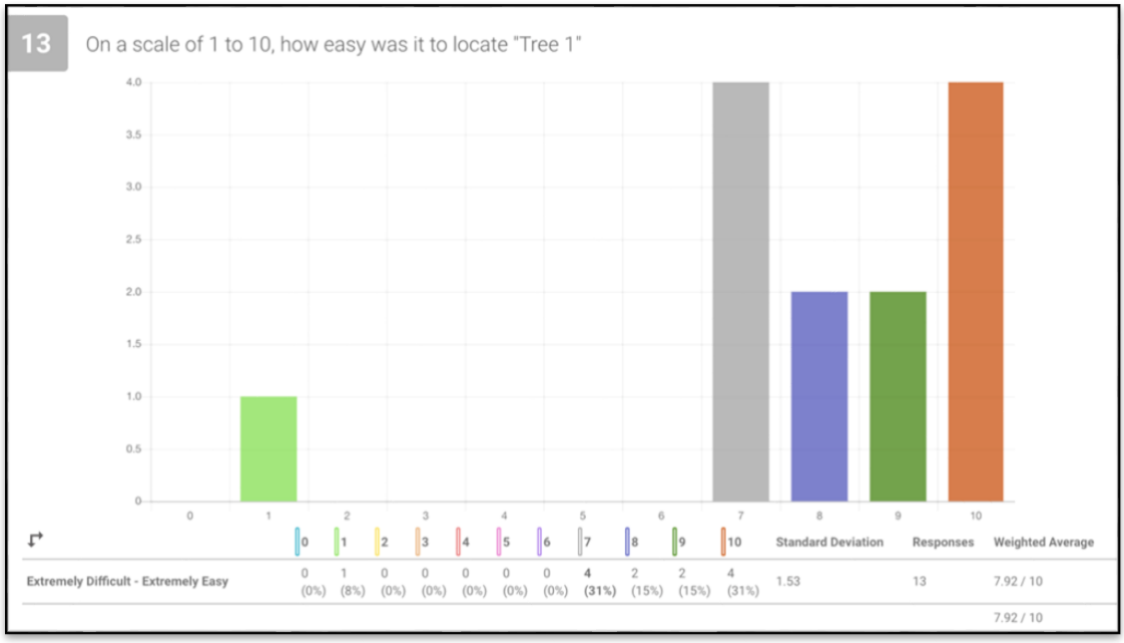


Figure 44 - Locating Tree 1 Likert Responses (Bar Chart)

This disparity between ratings can be explained by some of the participant criticism made in the optional response box that was linked to this task (Full comments shown in Appendix 4.7).

- ***"Incorrect tree icon loaded, no number 1?"***
- ***"Trees did not appear on map"***
- ***"Map was slow"***
- ***"Map took a long time to load"***

Many of the responses address a limitation that was discussed in section 6.6 highlighting the slow load times of the map interface due to the large Edinburgh tree dataset. There is also one occurrence where some trees did not load correctly for users (hence the rating of 1). This occurred due to the user having accessed the app previously during development stages and because of this, their cache memory stored an old version of the trees (this issue was discussed in section 6.2).

The third task led to the evaluation of the speech synthesis feature by asking users to locate "Tree 1" and get the information to be read aloud to them via the "SPEAK" button. The results demonstrated that participants on average found the feature to be moderately useful with 66% of votes falling between 7 and 9 (where 0 is "not useful at all" and 10 is "extremely useful"). This result can be viewed in Appendix 4.8. The reasoning behind this response is made clear through the comments given by participants, stating: "Great for people with reading difficulty", "Useful for disabilities but do not see myself using it", and "mindful of people with additional support needs". It is evident from these remarks that the speech system is more attractive for users with additional needs than general users. There was also criticism about the monotone quality of the computerized voice, however, this is a limitation of the speech synthesis API and could perhaps be improved upon in the future through the use of more advanced speech synth technology. The full participant comments can be seen in Appendix 4.9.

In regard to beginning a tour, 92% of participants stated that the experience was extremely simple due to the large "BEGIN THE TOUR" button located on the homepage (See Appendix 4.10 for Likert results). Furthermore, participants found it very intuitive to identify their current location on the map via the large red marker or arrow (depending on device), with 100% of participants scoring the clarity of their location with a rating of 9 or 10 (See Appendix 4.11).

As mentioned in the demographic's results section, some participants found it fairly complicated to add the application to their device's home screen. Adding the PWA to the home screen is not crucial for usability, however it is a primary reason as to why a PWA implementation was chosen and hence, the process should not be complicated. Fig. 45 below shows the Likert scale results of the "add to home screen" process, where a rating of 0 means the process was extremely complex and 10, extremely easy. The figure demonstrates a large spread between ratings ranging from as low as 4 to as high as 10. Only 15% of users rated the experience a 10, and it is no coincidence that these users were also using an Android device (meaning they had automatic install prompts on the webpage).

Overall, it was demonstrated that perhaps participants who had installed PWAs before or were more "tech savvy" found the process fairly intuitive, however, participants who had never used a PWA – it can be estimated that this is a large number of the general public since PWAs are still an emerging technology – found the process to be notably more complex.

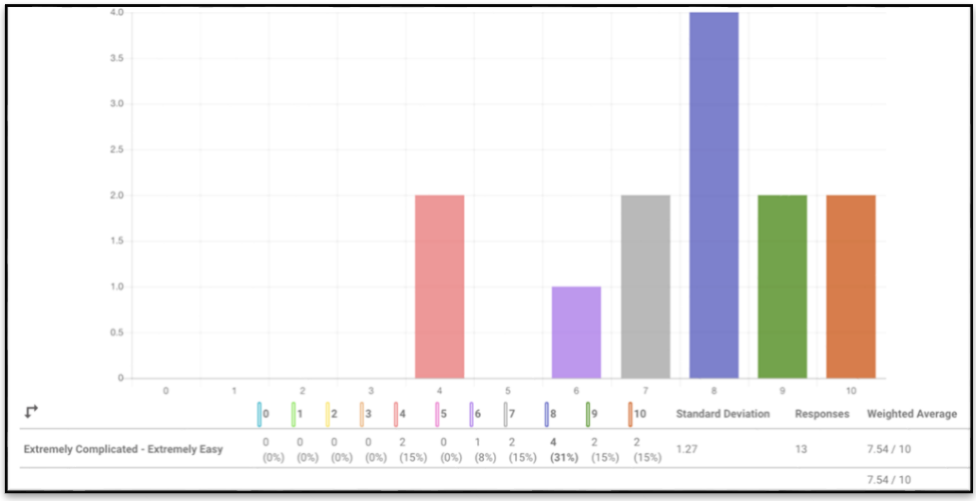


Figure 45 - Adding PWA to Home Screen Likert Results (Bar Chart)

8.3.3 Post-Questionnaire - User-Interface Analysis

Once a participant had completed the tasks above, they were given a post questionnaire that contained a variety of Likert scales where a user could express agreement to various user interface focussed questions. This section aims to deduce the end-user's average opinion of Trail's UI.

Overall, the user interface of the system proved to yield extremely positive results. Fig. 46 outlines the results of the first question, where participants were asked whether the application was easy to navigate. 85% of the participants strongly agreed with this statement (rating a 9 or 10) yielding a total average rating for navigation at 9.15/10.

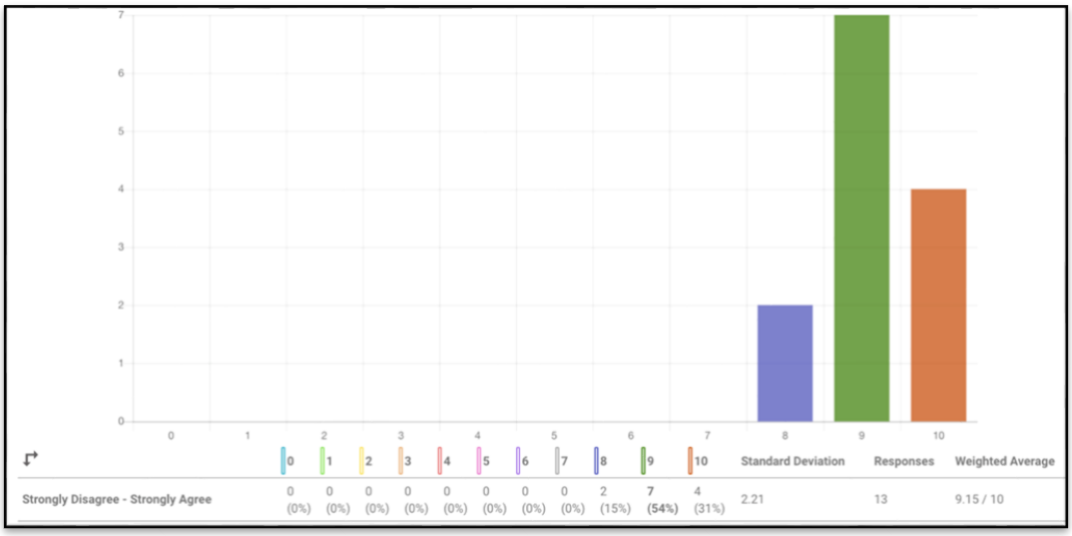


Figure 46 - Easiness of Navigation Likert Results (Bar Chart)

Participants were also asked to agree with the statement "The UI is aesthetically appealing" which yielded a similarly high average agreement at 9.46/10 (See Appendix 4.12 for results).

Results begin to lean towards ratings of 7 as questions start focussing on button function and placement. Fig. 47 below outlines the results of a question asking a user whether they immediately understood the function of each button. Fig. 48 details the results of asking whether participants felt buttons were easy to find. By analysing these two figures it is evident that while the UI as a whole is effective, individual buttons need some amendments to improve clarity.

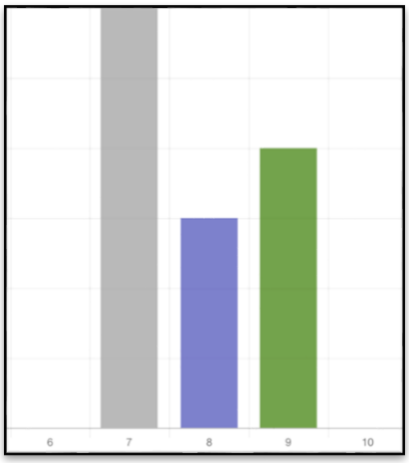


Figure 47 - Button Function Clarity Likert Result

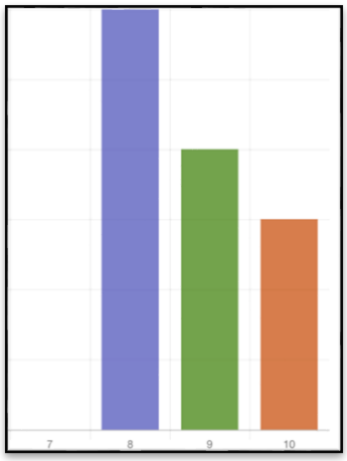


Figure 48 - Button Placement Likert Result

8.4 Study Conclusions and Additional Insights

To collect valuable feedback and gain further insight into participant's reasoning for their answers, a series of concluding text-based questions were asked. The first of these questions helped to understand the most positive aspects of the application by asking participants what their favourite feature was. Majority of participants particularly enjoyed the QR or AR technology, some of the comments are listed below:

- ***"I loved the use of AR and QR scanning with the camera"***
- ***"The facts that cycle through the home screen"***
- ***"The ability to switch between satellite mode and also the colour coded tree clusters"***

- ***"The augmented reality"***
- ***"Scanning QR codes"***
- ***"The speaking tool and how it can be downloaded as an app"***
- ***"I thought the QR code scanner was rather nifty"***

Participants were then asked what they would change about the application to understand the most negative aspects of the system. The main comments desire the inclusion of information that can make it easier to install the application as well as reducing the frequency of the "Lost?" alert. These comments are listed below:

- ***"A little extra styling, maybe a splash of colour"***
- ***"More consistency across different browsers"***
- ***"Dark mode, or ability to turn off the lost alert thing"***
- ***"I had to google how to install the app, some instructions would be nice"***
- ***"Add help for installing as a home-screen application"***
- ***"Maybe on dashboard tell a user how to install the application"***
- ***"Reduce the frequency of the lost pop-up"***
- ***"Add some text under menu buttons and maybe change Map to Tree Map"***

Finally, participants were asked to leave any additional comments regarding the application or the evaluation study. These comments include:

- ***"Some lag on map but besides that a very easy to use application!"***
- ***"Great app, amazing to learn about the trees on campus"***
- ***"I had accessed app during development so I needed to clear my cache so updates would show"***
- ***"Overall, really useful"***
- ***"Looking forward to doing the tour on campus!"***
- ***"I like how there is a prompt to install the application"***
- ***"I found it hard to find the QR page because I didn't recognise the logo"***

The open-ended text-based questions have provided invaluable information and an insight into the mind of the user. There are a variety of limitations that as a developer were overlooked such as the difficulty of installing the application or the annoyance of the "Lost?" alert.

Overall, the system performed incredibly well yielding very positive results, however, as demonstrated in this section there are issues that need amending, and hence, Section 9.3 will propose extensions to the system and future work that could be executed to improve upon these.

Chapter 9: Summary and Future Work

9.1 Achievements

9.1.1 Meeting of Aims and Objectives

The primary aim of this project was to develop a dependable and robust mobile application that allows a user to get information about specific trees around the Heriot-Watt Campus in Edinburgh. The application was envisioned to provide an interactive tour experience via the use of QR, spatial, and speech synthesis technologies to create a go-to tool for tree exploration.

This aim can be considered to have been mostly met, as the final application did allow for the user to learn about trees on the Heriot-Watt campus through the use of these technologies. However, there were a variety of issues such as the low iOS PWA support or the complications with cache memory not clearing, that directly countered the dependability and robustness of the application.

All research-based objectives were achieved through the in-depth literature review, successfully comparing similar systems, and producing the necessary findings in order to develop a user-friendly application that makes use of QR, spatial, speech synth and AR technology effectively.

9.1.2 Completed Implementation of All Planned Features (and more!)

The application was successfully developed as an installable PWA through the creation of a WebManifest.json file which outlined the application's main properties and icons, along with a Service Worker file that was used to handle file requests. This in turn provided some offline capabilities which met Google Chrome's PWA requirements. Furthermore, a user can scan tree QR codes using the applications QR scanner that was developed with the assistance of the *html5qrcode* library (handled using Golang and JavaScript functions). Alongside this, users can effectively spatially trigger nearby trees by walking into specially created geofences, their device will then successfully vibrate (haptic feedback) and then speak aloud all necessary tree information (Speech Synthesis). Spatial triggering was developed using PostGIS (a new learning curve) and the HTML5 geolocation API. Augmented Reality was also implemented using the AR.js library and was able to display tree information to a user, but unfortunately demonstrated adverse effects, as discussed in prior sections.

9.1.3 Successful System Evaluation and Usability Study

Unfortunately, due to Covid-19 guidelines and a no-show participant, the spatial triggering system could not be externally tested by users and hence, was tested internally to still evaluate the system. Despite the drawbacks of Covid-19, a successful usability study was executed in order to evaluate all non-physical aspects of the system and proved to yield extremely valuable results, many of which gave way to possible extensions and future work that could be done to the application.

9.2 Main Limitations

The limitations of the Trail application were discussed in depth in Section 6, and hence, Table 18 below will only summarise these concisely.

Table 12 - Main Limitations

Feature	Limitations
PWA	1. Occasionally the cache does not clear automatically on the installed PWA and hence, it must be cleared manually for some updates to take effect. This could result in user's running a version of the application that is out of date. 2. There is no capability to run the application's geolocation features while a user's device screen is locked. 3. iOS has very little support, namely the lack of an automatic install prompt.
User Positioning	1. Accuracy in urban environments can be volatile and hence the user cannot rely on the system to be 100% true in portraying their location. 2. If the user loses signal/internet connection, the lose geolocation abilities.
Spatial Triggering	1. Because of the volatility of location accuracy due to interference, there is scope for trees to trigger earlier or later than they should. Resulting in confusion to the user as the incorrect tree could be triggered.
Speech Synthesis	1. When geofences are entered in rapid succession the Speech Synthesis feature can mislead the user by still speaking information about e.g., tree 1, when in reality the user has already triggered and is viewing tree 2. 2. iOS does not support automatic playing of Speech. 3. Limited to monotone characteristics of current computerized voices.
AR	1. Web AR is an experimental feature and in many cases the AR entity glitches, travelling across the screen in explosive movements.
Map Data	1. While clustering (on the web client side) did improve the performance of the *Map* page it is still sub-optimal with load times ranging between 2 and 6 seconds depending on the age of the user's device. 2. Because of the clustering, the standard click event listener does not work as there is a prior event listener checking for clicks on the cluster icons. Meaning specific Edinburgh tree data could not be viewed so long as clusters are used.
Direction System	1. The orientation arrow fails to function on iOS devices. 2. The lost detector alerts on both devices have the ability to queue behind one another if they are not dismissed, leading to a cumbersome task of closing all of said alerts before resuming the tour.

9.3 Possible Extensions and Future Work

The usability study results, along with the identified limitations of features, and the exciting ideas about other possible new features for Trail (that a developer dreams about while undertaking a project) have all demonstrated that while the application was a largely a success, there is still so much scope for extension. This section will detail some future work that could be done to the Trail application, both to extend it, and improve it.

9.3.1 Group Nearby Trees Together in One Geofence

As highlighted in the limitation sections, there was an evident issue when it came to triggering trees that were nearby each other and had overlapping geofences. This issue resulted in trees being occasionally triggered before or after they should be. One way of fixing this would be to implement "group geofences" that encapsulate multiple trees e.g., 3 trees inside one geofence. This geofence would then trigger a summary of data for all 3 trees rather than specific data (which could be obtained via further UI interaction or QR code scanning). A virtual route line could also be created, the user is then snapped to that route using PostGIS functions, keeping them on the exact trail trajectory. An alternative solution is to return all trees within a 30m geofence and store them in order of distance. The stored tree data would then be announced to the user at specific locations e.g., Tree 2 data is displayed when walking between Tree 1 and 2.

9.3.2 User Interface Amendments - Factors Proposed by Study Participants

It was evident from results that there were a variety of features that needed improved upon to better the user experience. The primary issues highlighted were: The difficulty to install the app, the ambiguity of some menu bar icons, and the frequency of the "Lost?" pop-up. If there was more time to develop the application, these issues would have been fixed immediately after the evaluation took place. Issue one would be solved via install instructions located on the home screen that would only show to iOS users, issue two would be solved by adding labels underneath the menu icons, and issue three would be solved by changing the pop-up function to only run every 10 seconds instead of 4.

9.3.3 Ability to Obtain Statistical Data of Tours

If there had been more time for development, one feature that would had been extremely useful would be the ability to gain insight into how users interact with a tour. This could be simply from recording how long people spend inside each geofence, in order to deduce what trees are the most popular etc. As a version of this application is intended to actually be used on campus, this feature would provide Robbie Fraser and the University with valuable information on their client-base.

9.3.4 Future Large-Scale Ideas

1. The ability to select between multiple tours or select the length of the tour e.g., 10min, 1hr.
2. Personal user accounts with the ability to favourite trees and add their own trees.
3. Experimentation with AI to recognise trees by scanning their bark instead of QR codes.
4. Ability to filter map data e.g., show all trees from Spain or all Oak trees in HWU + Edinburgh.
5. Scale up Trail so it is community run and becomes a hosting platform for all kinds of tours.

References

agup, 2013. *How Accurate is HTML5 Geolocation, really? Part 2: Mobile Web.* [Online] Available at: https://www.andygup.net/how-accurate-is-html5-geolocation-really-part-2- mobile-web/ [Accessed 5 11 2020].

Alireza Sahami Shirazi, N. H. T. D. M. P. ,. D. W. A. S., 2014. *Large-Scale Assessment of Mobile Notifications,* Toronto: ACM.

Anand, 2018. *Can the Web NFC api be used in Progressive Web Applications?.* [Online] Available at: https://stackoverflow.com/questions/51134691/can-the-web-nfc-api-be-used- in-progressive-web-applications/51142826
[Accessed 29 10 2020].

Anna Chang, S. M., 2015. Improving reading rates and comprehension through audio-assisted extensive reading for beginner learners. *System,* Volume 52, pp. 92-102.

Apple, 2020. *Building an NFC Tag-Reader App.* [Online]
Available at: https://developer.apple.com/documentation/corenfc/building_an_nfc_tag-reader_app
[Accessed 28 10 2020].

AR.js org, 2021. *AR-js-org / AR.js.* [Online]
Available at: https://github.com/AR-js-org/AR.js
[Accessed 19 2 2021].

Babich, N., 2019. *The 4 Golden Rules of UI Design.* [Online]
Available at: https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/
[Accessed 24 10 2020].

Bellan J. M., &. S. G., 1998. Actual and virtual reality: Making the most of field trips. *Social Education,* Issue 62, p. 35–40.

Bloom, L. B., 2020. *Virtual Tours To Take During Coronavirus.* [Online]
Available at: https://www.forbes.com/sites/laurabegleybloom/2020/04/27/ranked-worlds- 15-best-virtual-tours-coronavirus/#6fa3362e6709
[Accessed 15 10 2020].

BlueBite, 2020. *How Does NFC Work?.* [Online]
Available at: https://www.bluebite.com/nfc/how-does-nfc-work
[Accessed 28 10 2020].

BlueBite, 2020. *The State of NFC in 2020.* [Online]
Available at: https://medium.com/@BlueBite/the-state-of-nfc-in-2020-2a512e83774b
[Accessed 16 11 2020].

Bours, A., Cetin, E. & Dempster, A., 2014. Enhanced GPS interference detection and localisation. *Ellectronic Letters,* 50(19), pp. 1391-1393.

Broersma, C., 2020. *How to Create Effective CTA Buttons for Your Website.* [Online] Available at: https://www.cazbah.net/boosting-business-by-building-better-buttons/ [Accessed 24 10 2020].

Busby, S., 2017. *Print2Life- Layar App.* [Online]
Available at:
https://play.google.com/store/apps/details?id=com.layar.print2life&hl=en&gl=US&showAllReviews=true
[Accessed 20 11 2020].

Cöster, P. P. E. F. S., 2003. GeoNotes: A Location-Based Information System for Public Spaces. In: *Designing Information Spaces: The Social Navigation Approach* . s.l.:Springer, London, pp. 151-173.

Chang, F.-H., 2015. Carbon Dioxide Concentrations and Temperatures within Tour Buses under RealTime Traffic Conditions. *Carbon Dioxide Concentrations and Temperatures within Tour Buses under Real-Time Traffic Conditions,* p. 1.

Ching-yin Law, S. S., 2010. QR Codes in Education. *Journal of Educational Technology Development and Exchange,* 3(1), pp. 85-90.

Choudhary, S. R., 2020. *How to scan QR codes with Android phones without an app.* [Online] Available at: https://blog.beaconstac.com/2019/03/how-to-scan-qr-codes-with-android-phones/#:~:text=Android%209%20and%20Android%2010,see%20a%20URL%20pop%2Dup.&text=To%20activate%20Google%20Lens%20to,app%20and%20click%20on%20more. [Accessed 27 10 2020].

Cliffe, A., 2017. A review of the benefits and drawbacks to virtual field guides in today's Geoscience higher education environment. *International Journal of Educational Technology in Higher Education ,* Volume 14.

College, Y.-N., 2020. *Smart devices should space out vibrations to maximize user alert benefits.* [Online]
Available at: https://www.sciencedaily.com/releases/2020/06/200602151321.htm [Accessed 20 11 2020].

Cornelis, L., 2020. *Push Notifications for Progressive Web Apps (PWA) Explained.* [Online] Available at: https://www.pushpro.io/blog/pwa-push-notifications-for-progressive-web- apps [Accessed 7 11 2020].

Coskun, V., Ozdenizci, B. & Ok, K., 2015. The Survey on Near Field Communication.. *Sensors (Basel),* 15(6), pp. 1-10.

Crenna, L., 2014. *Historypin.* [Online]
Available at: https://www.commonsensemedia.org/app-reviews/historypin
[Accessed 20 10 2020].

Criteo, 2020. *Mobile App Engagement: The Pros and Cons of Push Notifications and Email.* [Online]
Available at: https://www.criteo.com/blog/push-notifications-and-email-apac/
[Accessed 20 11 2020].

Denso, 2010. *What is a QR code?.* [Online]
Available at: https://www.qrcode.com/en/about/#featurePage1
[Accessed 25 10 2020].

Denso, 2019. *Error Correction Feature.* [Online]
Available at: https://www.qrcode.com/en/about/error_correction.html [Accessed 25 10 2020].

Denso, 2019. *Readable from any direction in 360.* [Online] Available at: https://www.qrcode.com/en/about/#featurePage5 [Accessed 25 10 2020].

Dghaidy, B., 2013. *What is your review of Layar?.* [Online]
Available at: https://www.quora.com/What-is-your-review-of-Layar
[Accessed 20 11 2020].

Dykes, J., 2000. An approach to virtual environments for visualization using linked geo-referenced panoramic imagery. *Computers, Environments and Urban Systems,* 24(2), pp. 127-152.

DZone, 2017. *How to Do Simple Geofencing With PostGIS.* [Online]
Available at: https://dzone.com/articles/how-to-do-simple-geofencing-with-postgis-1
[Accessed 2 11 2020].

Edinburgh Council, 2018. *Trees.* [Online]
Available at:
https://data.edinburghcouncilmaps.info/datasets/4dfc8f18a40346009b9fc32cbee34039_39?geometry=-3.212%2C55.950%2C-3.181%2C55.954&selectedAttribute=Height
[Accessed 12 4 2021].

Emile Morse, M. L. K. A. O., 2002. Testing visual information retrieval methodologies case study: Comparative analysis of textual, icon, graphical, and "spring" displays. *Journal of the American Society for Information Science and Technology ,* 53(1), pp. 1-22.

Enge, P. K., 2011. *The Global Positioning System: Signals, measurements, and performance,* Lincoln: Ganga-Jamuna Press.

Faisalman, 2021. *ua-parser-js.* [Online]
Available at: https://github.com/faisalman/ua-parser-js
[Accessed 20 2 2021].

Fletcher, S., 2002. Fieldwork education and technology: A GEES perspective. *Fieldwork education and technology: A GEES perspective,* Volume 4, pp. 17-19.

Fotaflo, 2018. *ANDROID QR CODE IMPROVEMENTS.* [Online]
Available at: https://www.fotaflo.com/product-updates/android-qr-code-improvements
[Accessed 20 11 2020].

GeoPandas, 2021. *GeoPandas 0.9.0.* [Online]
Available at: https://geopandas.org/
[Accessed 9 4 2021].

Gibbs, S., 2015. *Google Maps: a decade of transforming the mapping landscape.* [Online]
Available at: https://www.theguardian.com/technology/2015/feb/08/google-maps-10-anniversary-iphone-android-street-view [Accessed 20 10 2020].

Google, 2019. *Getting Started with Infographics.* [Online]
Available at: https://developers.google.com/chart/infographics/docs/overview
[Accessed 27 10 2020].

Google, 2020. *Display live data on your site.* [Online]
Available at: https://developers.google.com/chart
[Accessed 27 10 2020].

Google, 2020. *Maps Embed API.* [Online]
Available at: https://developers.google.com/maps/documentation/embed/get-started
[Accessed 4 11 2020].

Google, 2020. *MY MAPS.* [Online]
Available at: https://www.google.com/maps/about/mymaps/
[Accessed 4 11 2020].

Google, 2020. *My Maps - Embed this map.* [Online]
Available at:
https://www.google.com/maps/d/edit?hl=en&mid=19nWUrSIbzzyBa13LxIAP9qG9gtUs8Kwv&ll=55.906882477387384%2C-3.3219935158248037&z=16
[Accessed 4 11 2020].

Google, 2020. *Provide contextual experiences when users enter or leave an area of interest.* [Online]
Available at: https://developers.google.com/location-context/geofencing
[Accessed 2 11 2020].

Google, 2018. *Add a web app manifest.* [Online]
Available at: https://web.dev/add-manifest/
[Accessed 19 03 2021].

goQR, 2020. *QR code API: command "create-qr-code" (generate QR code, QR code generator).* [Online]
Available at: http://goqr.me/api/doc/create-qr-code/
[Accessed 27 10 2020].

goQR, 2020. *QR code API: command "read-qr-code" (read / scan QR code, QR code reader).* [Online]
Available at: http://goqr.me/api/doc/read-qr-code/
[Accessed 27 10 2020].

Griffin-Shirley N, B. D. A. P., 2017. Survey on the Use of Mobile Applications for People who Are Visually Impaired. *Journal of Visual Impairment & Blindness,* 111(4), pp. 307-323.

Hall, C., 2020. *What is Google Lens and what can it do?.* [Online]
Available at: https://www.pocket-lint.com/apps/news/google/141075-what-is-google-lens- and-how-does-it-work-and-which-devices-have-it
[Accessed 27 10 2020].

Heriot-Watt, 2020. *Virtual tours.* [Online]
Available at: https://www.hw.ac.uk/virtual-tours/?l=edinburgh&t=live&c=tour_34&o=0
[Accessed 15 10 2020].

HistoryPin, 2020. *About.* [Online]
Available at: https://about.historypin.org/about/
[Accessed 20 10 2020].

Image+, 2018. *Benefits of Good UI Design.* [Online]
Available at: https://www.image-plus.co.uk/2018/08/27/benefits-of-good-ui-design/
[Accessed 24 10 2020].

insideGNSS, 2009. *What About Vector Tracking Loops?.* [Online] Available at: https://insidegnss.com/what-about-vector-tracking-loops/ [Accessed 7 11 2020].

Jayce, D., 2019. *Can I scan QR codes without internet?.* [Online]
Available at: https://www.quora.com/Can-I-scan-QR-codes-without-internet [Accessed 20 11 2020].

Kantra, S., 2020. *16 Surprising Things You Can Do with Google Maps.* [Online]
Available at: https://www.techlicious.com/tip/surprising-things-you-can-do-with-google- maps/
[Accessed 20 10 2020].

Kenneth Christiansen, Z. K. F. B., 2020. *Web NFC explained.* [Online]
Available at: https://github.com/w3c/web-nfc/blob/gh-pages/EXPLAINER.md
[Accessed 29 10 2020].

Kenneth Rohde Christiansen, Z. K. B., 2020. *Web NFC,* s.l.: Web NFC Community Group .
Koshman, S., 1996. Usability testing of a prototype visualization-based information retrieval system. *PROCEEDINGS OF THE ASIS ANNUAL MEETING,* Volume 33, pp. 278-279.

Kinlan, P., 2020. *Adding a Service worker and offline to your web app.* [Online]
Available at: https://developers.google.com/web/fundamentals/codelabs/offline
[Accessed 19 03 2021].

Kościelniak, K., 2021. *Progressive Web Apps: Advantages and Disadvantages.* [Online]
Available at: https://brainhub.eu/library/progressive-web-apps-advantages-disadvantages/
[Accessed 14 4 2021].

Layar, 2020. *ABOUT LAYAR.* [Online] Available at: https://www.layar.com/about/ [Accessed 20 11 2020].

Lee, A., 2020. *40 Examples of Progressive Web Apps (PWAs) in 2020.* [Online]
Available at: https://www.tigren.com/examples-progressive-web-apps-pwa/
[Accessed 13 4 2021].

LePage, P., 2020. *What does it take to be installable?.* [Online]
Available at: https://web.dev/install-criteria/
[Accessed 17 03 2021].

Lindsey, R., 2020. *Climate Change: Atmospheric Carbon Dioxide.* [Online]
Available at: https://www.climate.gov/news-features/understanding-climate/climate-change-atmospheric-carbon-dioxide
[Accessed 15 10 2020].

Li-Ta Hsu, S.-S. J. P. D. G. N. K., 2015. Multipath mitigation and NLOS detection using vector tracking in urban environments. *GPS Solutions,* 19(2), p. 249–262.

Litherland, S., 2012. Virtual field sites: Losses and gains in authenticity with semantic technologies. *Technology, Pedagogy and Education,* 21(2), pp. 213-230.

Localysts, 2015. *The Inside View: How Consumers Really Feel About Push Notifications.* [Online]
Available at: https://uplandsoftware.com/localytics/resources/blog/the-inside-view-how-consumers-really-feel-about-push-notifications/#:~:text=Over%2050%25%20of%20app%20users,which%20is%20not%20a%20surprise.&text=It%20also%20means%20testing%20and,when%20users%20are%20m [Accessed 20 11 2020].

Locatify, 2020. *What is The Automatic Tourist Guide?.* [Online]
Available at: https://locatify.com/automatic-tourist-guide/
[Accessed 20 10 2020].

Love, C., 2018. *Progressive Web Application Development by Example.* s.l.:Packt.
Maher, R., 2017. *Background Geolocation via Service Workers #13.* [Online]
Available at: https://github.com/w3c/geolocation-api/issues/13
[Accessed 22 3 2021].

Lu-miao Liu, W. L. J.-j. D., 2017. *Haptic technology and its application in education and learning.* Pattaya, Thailand, IEEE.

M. Mareli, S. R. B. S. P. K. O. a. A. P., 2013. *Experimental evaluation of NFC reliability between an RFID tag and a smartphone.* Pointe-Aux-Piments, Mauritius, IEEE.

Mall, S., 2016. *Advantages of QR Code: 5 real reasons to use them.* [Online] Available at: https://scanova.io/blog/blog/2016/02/16/advantages-of-qr-code/ [Accessed 20 11 2020].

Matthias Wildemeersch, J. F.-G., 2010. *Radio Frequency Interference Impact Assessment on Global Navigation Satellite Systems,* s.l.: Luxembourg: Publications Office of the European Union.

Microsoft, 2012. *8 reasons you want NFC on your phone.* [Online]
Available at: https://blogs.windows.com/devices/2012/12/07/eight-reasons-you-want-nfc/
[Accessed 28 10 2020].

Mikayilov, J. M. S. M. J. &. A. M., 2019. Re-evaluating the environmental impacts of tourism: does EKC exist?. *Environmental science and pollution research international,* 26(19), pp. 19389-19402.

Minhaz, 2020. *html5-qrcode/examples/html5/.* [Online]
Available at: https://github.com/mebjas/html5-qrcode/tree/master/examples/html5
[Accessed 27 10 2020].

Minhaz, 2020. *mebjas / html5-qrcode.* [Online] Available at: https://github.com/mebjas/html5-qrcode [Accessed 27 10 2020].

Monkey, S., 2021. *Writing good survey questions.* [Online]
Available at: https://www.surveymonkey.co.uk/mp/writing-survey-questions/
[Accessed 25 3 2021].

Moth, D., 2013. *85% of consumers favour apps over mobile websites.* [Online]
Available at: https://econsultancy.com/85-of-consumers-favour-apps-over-mobile-websites/
[Accessed 17 3 2021].

Mozilla, 2020. *Detecting device orientation.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/API/Detecting_device_orientation
[Accessed 13 4 2021].

Mozilla, 2020. *<audio>: The Embed Audio element.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/audio
[Accessed 3 11 2020].

Mozilla, 2020. *Describing vibrations.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/API/Vibration_API
[Accessed 8 11 2020].

Mozilla, 2020. *PositionOptions.enableHighAccuracy.* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/API/PositionOptions/enableHighAccuracy
[Accessed 7 11 2020].

N.Bidargaddi, T., 2018. Predicting which type of push notification content motivates users to engage in a self-monitoring app. *Preventetive medicine reports,* Volume 11, pp. 267-273.

National Oceanic and Atmospheric Administration Office of Oceanic and Atmospheric Research, 2020. *Global Monitoring Laboratory - Carbon Cycle Greenhouse Gases.* [Online]
Available at: https://www.esrl.noaa.gov/gmd/ccgg/trends/
[Accessed 15 10 2020].

NFCW, 2020. *NFC phones: The definitive list.* [Online]
Available at: https://www.nfcw.com/nfc-phones-list/
[Accessed 28 10 2020].

NHS, 2018. *Blindness and vision loss.* [Online] Available at: https://www.nhs.uk/conditions/vision-loss/#:~:text=In%20the%20UK%2C%20there%20are,to%20come%20to%20terms%20with. [Accessed 3 11 2020].

NHS, 2018. *Dyslexia.* [Online]
Available at:
https://www.nhs.uk/conditions/dyslexia/#:~:text=It's%20estimated%20up%20to%201,successful%20at%20school%20and%20work.
[Accessed 3 11 2020].

NOAA, 2020. *Global Monitoring Laboratory - Carbon Cycle Greenhouse Gases.* [Online]
Available at: https://www.esrl.noaa.gov/gmd/ccgg/trends/
[Accessed 15 10 2020].

OpenStreetMap, 2020. *About OpenStreetMap.* [Online]
Available at: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap [Accessed 4 11 2020].

OpenStreetMap, 2020. *Frameworks.* [Online]
Available at: https://wiki.openstreetmap.org/wiki/Frameworks#Webmaps [Accessed 4 11 2020].

Oppong, T., 2019. *4 Pros & Cons of Push Notifications For Your Elearning Company.* [Online]
Available at: https://alltopstartups.com/2019/09/04/4-pros-cons-of-push-notifications-for-your-elearning-company/ [Accessed 8 11 2020].

Patel, P., 2018. *Smartphones are warming the planet far more than you think.* [Online] Available at: https://www.anthropocenemagazine.org/2018/04/the-energy-hogging-dark- side-of-smartphones/#:~:text=Making%20a%20phone%20accounts%20for,equivalent%20between%202010%20and%202020.
[Accessed 15 10 2020].

Pavitra, D., 2019. *What are some advantages of push notifications?.* [Online]
Available at: https://www.quora.com/What-are-some-advantages-of-push-notifications
[Accessed 20 11 2020].

PCplus, 2011. *Layar review.* [Online]
Available at: https://www.techradar.com/uk/reviews/pc-mac/software/home-and-reference-software/layar-968308/review
[Accessed 20 11 2020].

Per Persson, F. E. F. S. C., 2003. GeoNotes: A Location-Based Information System for Public Spaces. *Designing Information Spaces: The Social Navigation Approach,* pp. 151-173.

Perea, H., 2019. *Button Design essentials for your Web UI.* [Online]
Available at: https://swapps.com/blog/button-design-essentials-for-your-web-ui/#:~:text=Buttons%20not%20only%20make%20your,proper%20message%20to%20your%20audience.
[Accessed 24 10 2020].

P, M., 2016. *How does google map location works?.* [Online]
Available at: https://stackoverflow.com/questions/37894741/how-does-google-map- location-works#:~:text=If%20you%20run%20the%20Google,sensor%2C%20WIFI%2C%20network%20 etc.
[Accessed 20 10 2020].

pokusew, 2020. *pokusew / nfc-pcsc.* [Online]
Available at: https://github.com/pokusew/nfc-pcsc#basic-usage
[Accessed 28 10 2020].

PostGIS, 2020. *About PostGIS.* [Online]
Available at: https://postgis.net/
[Accessed 2 11 2020].

PostgreSQL, 2020. *What is PostgreSQL?.* [Online]
Available at: https://www.postgresql.org/about/
[Accessed 2 11 2020].

Praveen, A., 2015. *Are all big websites written in HTML and CSS?.* [Online]
Available at: https://www.quora.com/Are-all-big-websites-written-in-HTML-and-CSS
[Accessed 3 11 2020].

precisionMicrodrives, 2020. *What Is Haptic / Tactile Feedback?.* [Online]
Available at: https://www.precisionmicrodrives.com/haptic-feedback/introduction-to- haptic-feedback/
[Accessed 8 11 2020].

QRCode Monkey, 2020. *QR Code API with Logo and Design.* [Online]
Available at: https://www.qrcode-monkey.com/qr-code-api-with-logo
[Accessed 20 11 2020].

Rajendra Singh, H. B., 2013. QR Codes in Print Advertising: Elucidating Indian Vogue Using Content Analysis. *Management & Marketing,* 8(2), pp. 353-368.

ResearchArticles.com, 2019. *Textual Presentation of Data.* [Online]
Available at: http://researcharticles.com/index.php/textual-presentation-of-data/
[Accessed 3 11 2020].

RFwireless, 2020. *Advantages of NFC | disadvantages of NFC.* [Online] Available at: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-NFC.html
[Accessed 16 11 2020].

Rhonda Hadi, A. V., 2020. Good Vibrations: Consumer Responses to Technology-Mediated Haptic Feedback. *Journal of Consumer Research,* 47(2), pp. 256-271.

Sabella, R. R., 2016. *The 5 NFC Tag Types.* [Online]
Available at: https://www.dummies.com/consumer-electronics/5-nfc-tag-types/
[Accessed 28 10 2020].

Samantha Abelinde, L. T. L. F. M. E. S. M. C. F. Q., 2014. *Benefits of Audio Media.* [Online]
Available at: https://audiomediaet101.weebly.com/benefits-of-audio-media.html [Accessed 3 11 2020].

Scottish Government, 2020. *Coronavirus (COVID-19): what you can and cannot do.* [Online]
Available at: https://www.gov.scot/publications/coronavirus-covid-19-what-you-can-and- cannot-do/pages/seeing-friends-and-family/
[Accessed 15 10 2020].

Selle, H., 2015. *Is there someway to make "My maps" in google maps available offline with location markers?.* [Online]
Available at: https://android.stackexchange.com/questions/31961/is-there-someway-to- make-my-maps-in-google-maps-available-offline-with-locatio
[Accessed 4 11 2020].

smith, M., 123. aople colours. *fruits,* 2(11), pp. 342-3434.

Software Brothers, 2020. *PROGRESSIVE WEB APPS – PROS, CONS, COST AND POPULAR TECHNOLOGIES USED FOR DEVELOPMENT.* [Online]
Available at: https://softwarebrothers.co/blog/progressive-web-apps-pros-cons-cost-and-popular-technologies-used-for-development/
[Accessed 14 4 2021].

sproutQR, 2020. *QR Code Minimum Size: How Small Can a QR Code Be?.* [Online] Available at: https://www.sproutqr.com/blog/qr-code-minimum-size#toc-qr-code- minimum-size
[Accessed 20 11 2020].

Steve Fletcher, D. F. K. M. &. G. R., 2002. Fieldwork Education and Technology: A GEES Perspective. *Planet,* 7(1), pp. 17-19.

Stott T., L. K. C. P. N. A., 2014. Using Interactive Virtual Field Guides and Linked Data in Geoscience Teaching and Learning. In: *Geoscience Research and Education.* s.l.:Springer, Dordrecht, pp. 163-188.

Suhua Tang, Y. Y. R. Z. S. O., 2015. Efficient Geo-Fencing via Hybrid Hashing: A Combination of Bucket Selection and In-Bucket Binary Search. *ACM Transactions on Spatial Algorithms and Systems,* 1(2), pp. 1-20.

Suschevich, A., 2020. *PWA: Pros and Cons.* [Online]
Available at: https://medium.com/the-innovation/pwa-pros-and-cons-28edb4728e40
[Accessed 12 4 2021].

Syzonenko, A., 2019. *Buttons on the web: placement and order.* [Online] Available at: https://uxdesign.cc/buttons-placement-and-order-bb1c4abadfcb [Accessed 20 11 2020].

Tankovska, H., 2020. *Standalone and embedded consumer AR mobile applications worldwide 2016-2022 Published by H. Tankovska, Sep 1, 2020 The statistic shows the number of mobile augmented reality applications worldwide in the consumer market from 2016 to 2022. In 2016, the in.* [Online]
Available at: https://www.statista.com/statistics/608967/mobile-ar-applications-installed- base-worldwide/
[Accessed 20 11 2020].

Taylor, P., 2009. Introduction. In: *Text-to-Speech Synthesis.* Cambridge: Cambridge University Press, pp. 1-10
The Frankline Institute, 2020. *WHAT IS AUGMENTED REALITY?.* [Online]

Available at: https://www.fi.edu/what-is-augmented-reality
[Accessed 20 11 2020].
Thomas Verheyde, A. B. C. M. F.-X. M., 2020. *Analyzing Android GNSS Raw Measurements Flags Detection Mechanisms for Collaborative Positioning in Urban Environment.* Tampere, Finland, HAL.

ultraleap, 2020. *What is Haptic Feedback?.* [Online]
Available at: https://www.ultraleap.com/company/news/blog/what-is-haptic-feedback/
[Accessed 8 11 2020].

Vasishta, J., 2020. *Six Virtual Tours to Escape the Coronavirus Lockdown.* [Online]
Available at: https://www.dirt.com/feature/virtual-tours-coronavirus-lockdown-1203317376/
[Accessed 15 10 2020].

Vedat Coskun, B. O. a. K. O., 2015. The Survey on Near Field Communication. *Sensors,* Volume 15, pp. 13348-13405.

VWO, 2017. *What do recipients find annoying about push notifications?.* [Online]
Available at: https://www.businessofapps.com/marketplace/push-notifications/research/push-notifications-statistics/
[Accessed 20 11 2020].

w3, 2008. *Web Content accessability guidelines.* [Online]
Available at: https://www.w3.org/TR/WCAG20/#seizure-does-not-violate
[Accessed 16 11 2020].
w3c, 2020. *GRAPHICS.* [Online]
Available at: https://www.w3.org/standards/webdesign/graphics
[Accessed 3 11 2020].

w3c, 2020. *HTML5 Geolocation AP.* [Online]
Available at: https://www.w3schools.com/html/html5_geolocation.asp [Accessed 5 11 2020].

W3schools, 2020. *Bootstrap 4 Get Started.* [Online]
Available at: https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp
[Accessed 24 10 2020].

W3schools, 2020. *HTML <button> Tag.* [Online]
Available at: https://www.w3schools.com/tags/tag_button.asp [Accessed 24 10 2020].

w3schools, 2020. *HTML Introduction.* [Online]
Available at: https://www.w3schools.com/html/html_intro.asp [Accessed 3 11 2020].

Wakadev, 2019. *NFC TOOLS - IOS.* [Online]
Available at: https://www.wakdev.com/apps/nfc-tools-ios.html
[Accessed 28 10 2020].

Warcholinski, M., 2021. *Progressive Web Apps: Advantages and Disadvantages.* [Online]
Available at: https://brainhub.eu/library/progressive-web-apps-advantages-disadvantages/
[Accessed 14 4 2021].

WebNFC Community Group, 2020. *Web NFC Draft Community Group Report.* [Online]
Available at: https://w3c.github.io/web-nfc/
[Accessed 28 10 2020].

White, E., 2020. *Why you Need Haptic Feedback in your App.* [Online]
Available at: https://www.bairesdev.com/blog/your-app-needs-haptic-feedback/
[Accessed 20 11 2020].

White, S. K., 2017. *What is geofencing? Putting location to work.* [Online]
Available at: https://www.cio.com/article/2383123/geofencing-explained.html#:~:text=Geofencing%20is%20a%20location%2Dbased,location%2C%20known%20as%20a%20geofence.
[Accessed 2 11 2020].

WHO, 2010. *global data on visual impairment.* [Online]
Available at: https://www.who.int/blindness/publications/globaldata/en/
[Accessed 20 11 2020].

Wikitude, 2019. *Travel company TUI tests the use of AR in its activities.* [Online]
Available at: https://www.wikitude.com/showcase/tui-tests-the-use-of-augmented-reality- in-its-activities/
[Accessed 20 11 2020].

wwcdt, 2020. *NFC.* [Online]
Available at: https://whatwebcando.today/nfc.html
[Accessed 16 11 2020].

Yu-Hsuan Lin, S.-H. L. L.-L. H.-Y. C., 2013. Prevalent Hallucinations during Medical Internships: Phantom Vibration and Ringing Syndromes. *PLoS ONE,* 8(2), p. e65152.

Zanetis, J., 2010. The Beginner's Guide to Interactive Virtual Field Trips. *Learning & Leading with Technology,* 37(6), pp. 20-23.

1. Background Appendices

1.0 Near Field Communication (NFC)

NFC works using a reader (normally a smartphone) and a tag(a small signal transmitter); these tags are read/write and can contain from 96 to 4,098 bytes of data depending on the tag type (BlueBite, 2020). Tags range from Type 1 (simplest and slowest) to Type 5 (more features), with Tag Type 4 offering the most flexibility and memory out of the five (Sabella, 2016). These tags are commonly manufactured as stickers consisting of an antenna + Integrated Circuit chip, plastic substrate, adhesive package and release liner (BlueBite, 2020).

The figure below from *"The Survey on Near Field Communication"* details how a smartphone interacts with these NFC tags (Vedat Coskun, 2015):

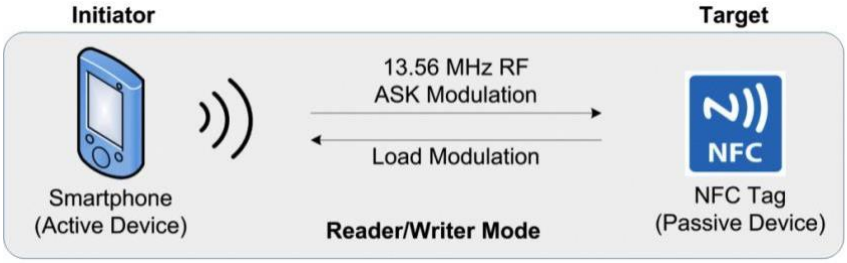
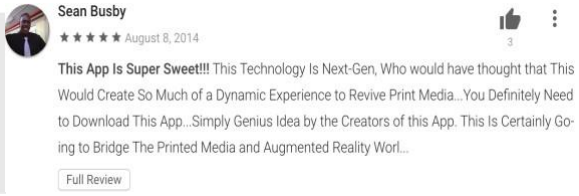


Figure 49- NFC Smartphone Interaction, The Survey on Near Field Communication

NFC functions using magnetic induction, i.e. the reader will emit an electric charge creating a magnetic field. This is used to power the NFC Tag (passive) which converts it into electrical impulses to transfer the data. A read is performed once the devices are in range of each other (WebNFC Community Group, 2020).

1.1 Layar Reviews

(PCplus, 2011)



(Busby, 2017)

(Dghaidy, 2013)

Figure 50 - Layar User App Reviews

1.2 Push Notifications Background

To implement push notifications in a PWA environment a developer must use what is known as Web Push Notifications (Cornelis, 2020) , these notifications require a 3rd party push notification service such as Google Chrome's Push Notification system or PushPro. The following figure provided by PushPro[9] details how web push notifications arrive to a user's device:

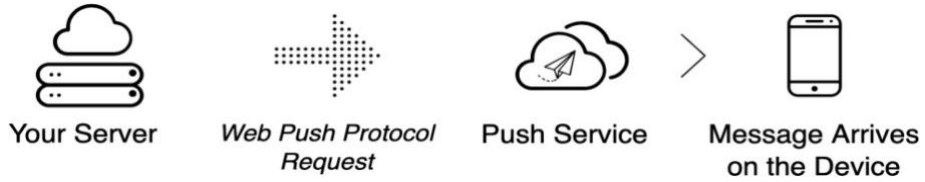


Figure 51 - Push Notification Device Interaction

Some key advantages and disadvantages of push notifications are described in Table 3:

Table 13 - Pros and Cons of Push Notifications

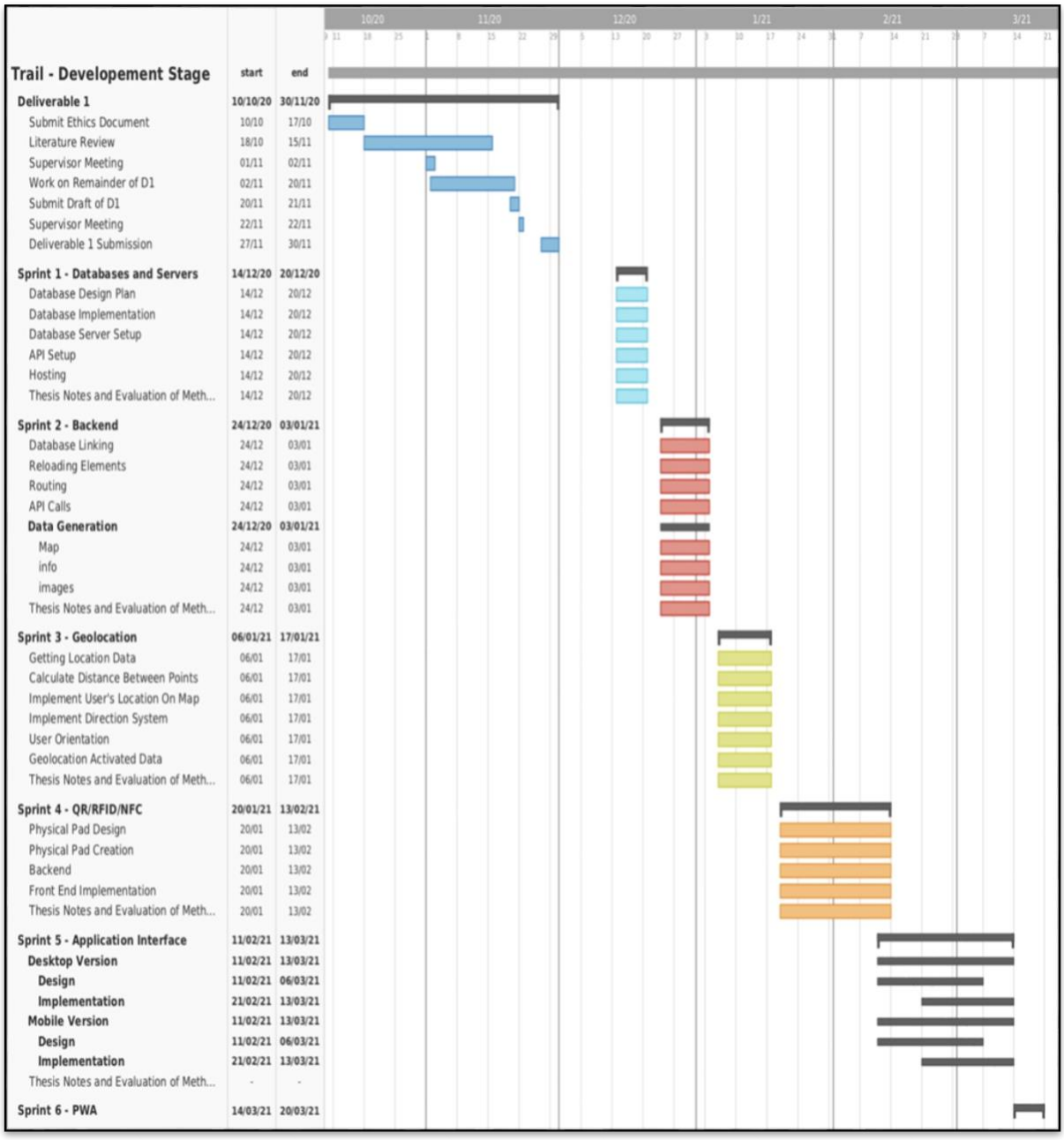
Advantages	Disadvantages
Promotes user engagement.	Statistics show 50% of users find push notifications annoying (Localysts, 2015)
Users can control what and how they see in notifications (Pavitra, 2019)	Users must opt in to use push notifications, minimising possible effectiveness (Criteo, 2020).
Through tracking notification interaction times, it is possible to gain an insight into user behaviour (Pavitra, 2019)	It is easy to dismiss and ignore a push notification due to their small size (Criteo, 2020).
Cleaner and easier than email or SMS	A survey conducted by VWO found that 49.1% of participants found push notifications distracting (VWO, 2017)

Overall, push notifications play a vital part in a user's interaction with an application and are a necessity to improving the use of a system. With the Trail application, it would be beneficial to use push notifications however, they should be kept simple, spread out, and a user should be able to opt-in and opt-out whenever they wish.

[9] PushPro is a company specializing in created 3rd party web push notification software for PWA's (Cornelis, 2020)

2. General Appendices

2.1 Gantt Chart



Trail - Developement Stage	start	end
Deliverable 1	10/10/20	30/11/20
Submit Ethics Document	10/10	17/10
Literature Review	18/10	15/11
Supervisor Meeting	01/11	02/11
Work on Remainder of D1	02/11	20/11
Submit Draft of D1	20/11	21/11
Supervisor Meeting	22/11	22/11
Deliverable 1 Submission	27/11	30/11
Sprint 1 - Databases and Servers	14/12/20	20/12/20
Database Design Plan	14/12	20/12
Database Implementation	14/12	20/12
Database Server Setup	14/12	20/12
API Setup	14/12	20/12
Hosting	14/12	20/12
Thesis Notes and Evaluation of Meth...	14/12	20/12
Sprint 2 - Backend	24/12/20	03/01/21
Database Linking	24/12	03/01
Reloading Elements	24/12	03/01
Routing	24/12	03/01
API Calls	24/12	03/01
Data Generation	24/12/20	03/01/21
Map	24/12	03/01
info	24/12	03/01
images	24/12	03/01
Thesis Notes and Evaluation of Meth...	24/12	03/01
Sprint 3 - Geolocation	06/01/21	17/01/21
Getting Location Data	06/01	17/01
Calculate Distance Between Points	06/01	17/01
Implement User's Location On Map	06/01	17/01
Implement Direction System	06/01	17/01
User Orientation	06/01	17/01
Geolocation Activated Data	06/01	17/01
Thesis Notes and Evaluation of Meth...	06/01	17/01
Sprint 4 - QR/RFID/NFC	20/01/21	13/02/21
Physical Pad Design	20/01	13/02
Physical Pad Creation	20/01	13/02
Backend	20/01	13/02
Front End Implementation	20/01	13/02
Thesis Notes and Evaluation of Meth...	20/01	13/02
Sprint 5 - Application Interface	11/02/21	13/03/21
Desktop Version	11/02/21	13/03/21
Design	11/02/21	06/03/21
Implementation	21/02/21	13/03/21
Mobile Version	11/02/21	13/03/21
Design	11/02/21	06/03/21
Implementation	21/02/21	13/03/21
Thesis Notes and Evaluation of Meth...	-	-
Sprint 6 - PWA	14/03/21	20/03/21

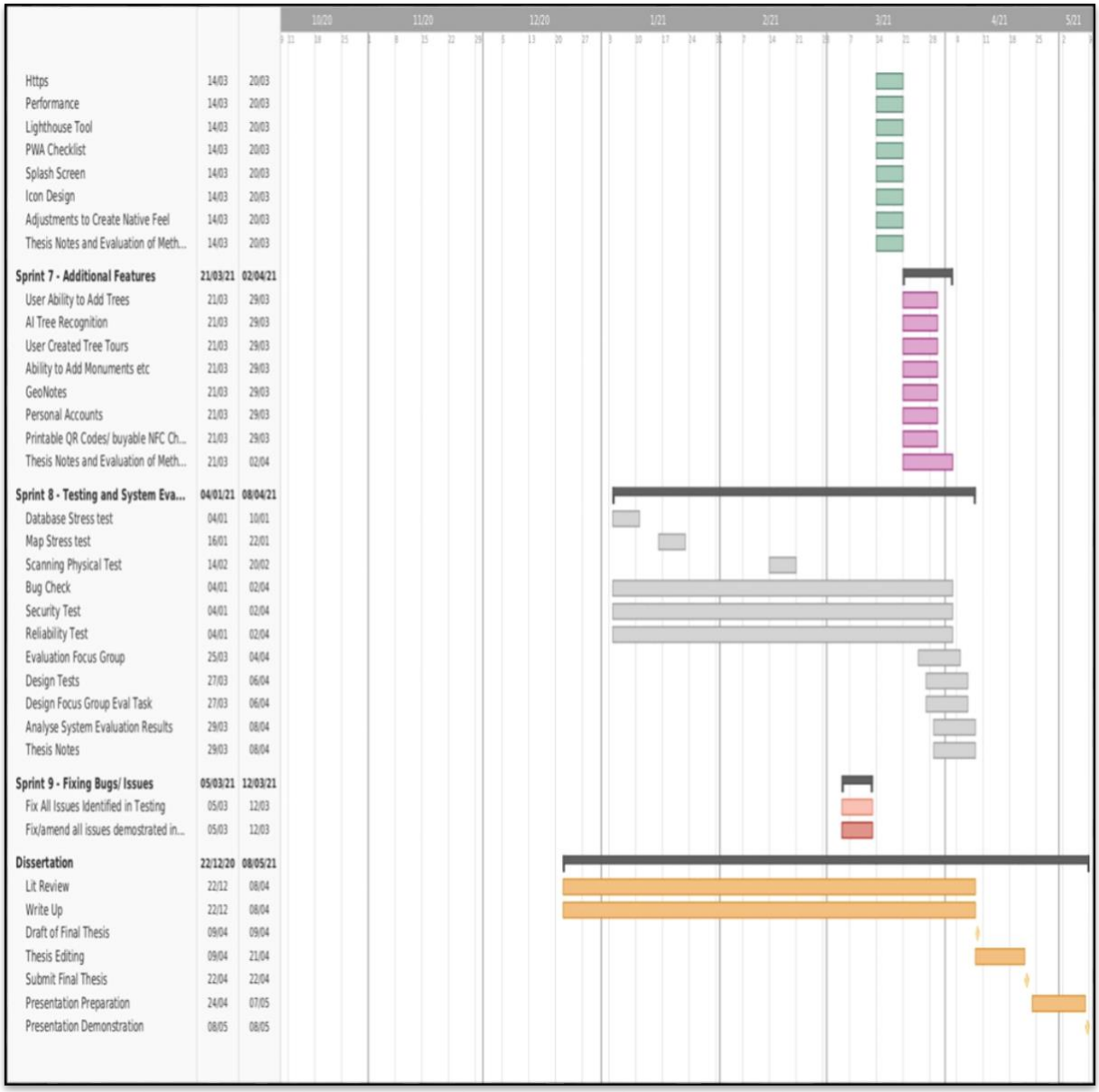


Figure 52 - Project Gantt Chart

2.2 Kanban Example

The following Kanban board was created last year using GitLab's Issue Board feature.

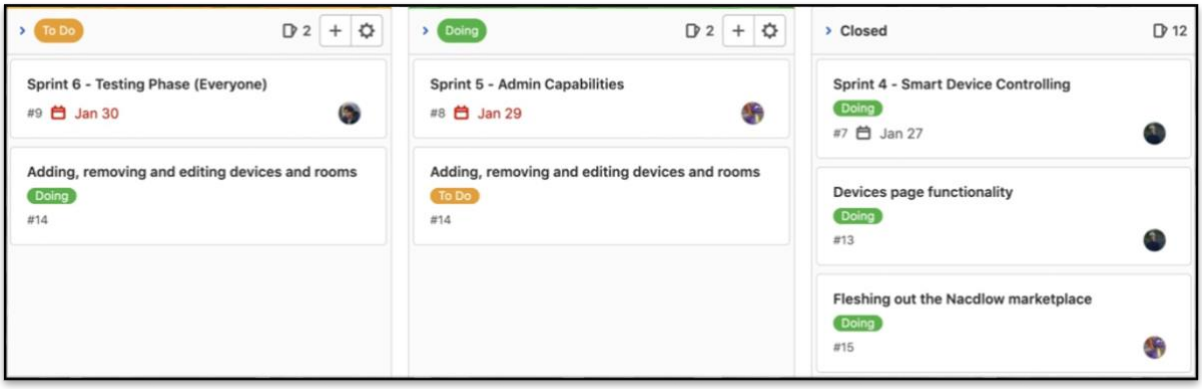


Figure 53 - Kanban Board Example

2.3 Heroku SSL Certification

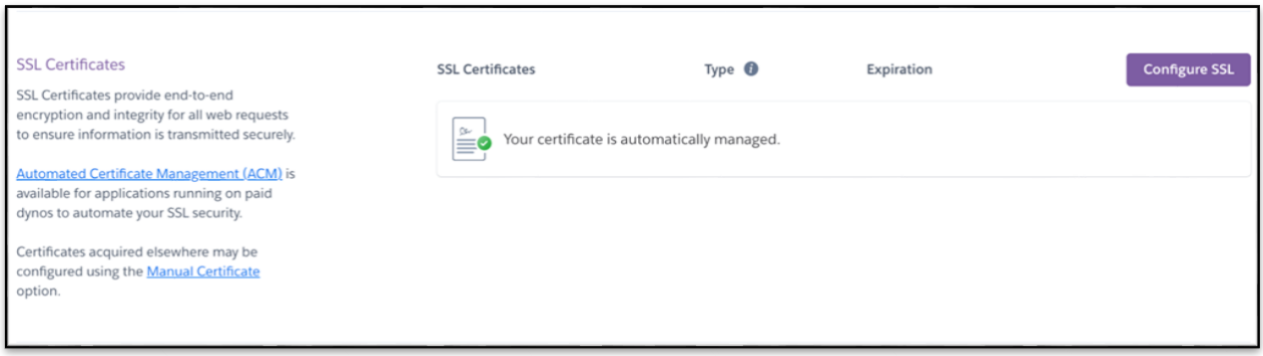


Figure 54 – Heroku SSL Certificate

2.4 Domain Name HTTPS Redirect

Figure 55 - Domain Name HTTPS Redirect

2.5 Web Manifest Requirements

- Includes a Web App Manifest that includes:
  - `short_name` or `name`
  - `icons` - must include a 192px and a 512px icon
  - `start_url`
  - `display` - must be one of `fullscreen`, `standalone`, or `minimal-ui`
  - `prefer_related_applications` must not be present, or be `false`

Figure 56 - PWA Web Manifest Requirements by Web.dev

2.6 PWA Lighthouse Test

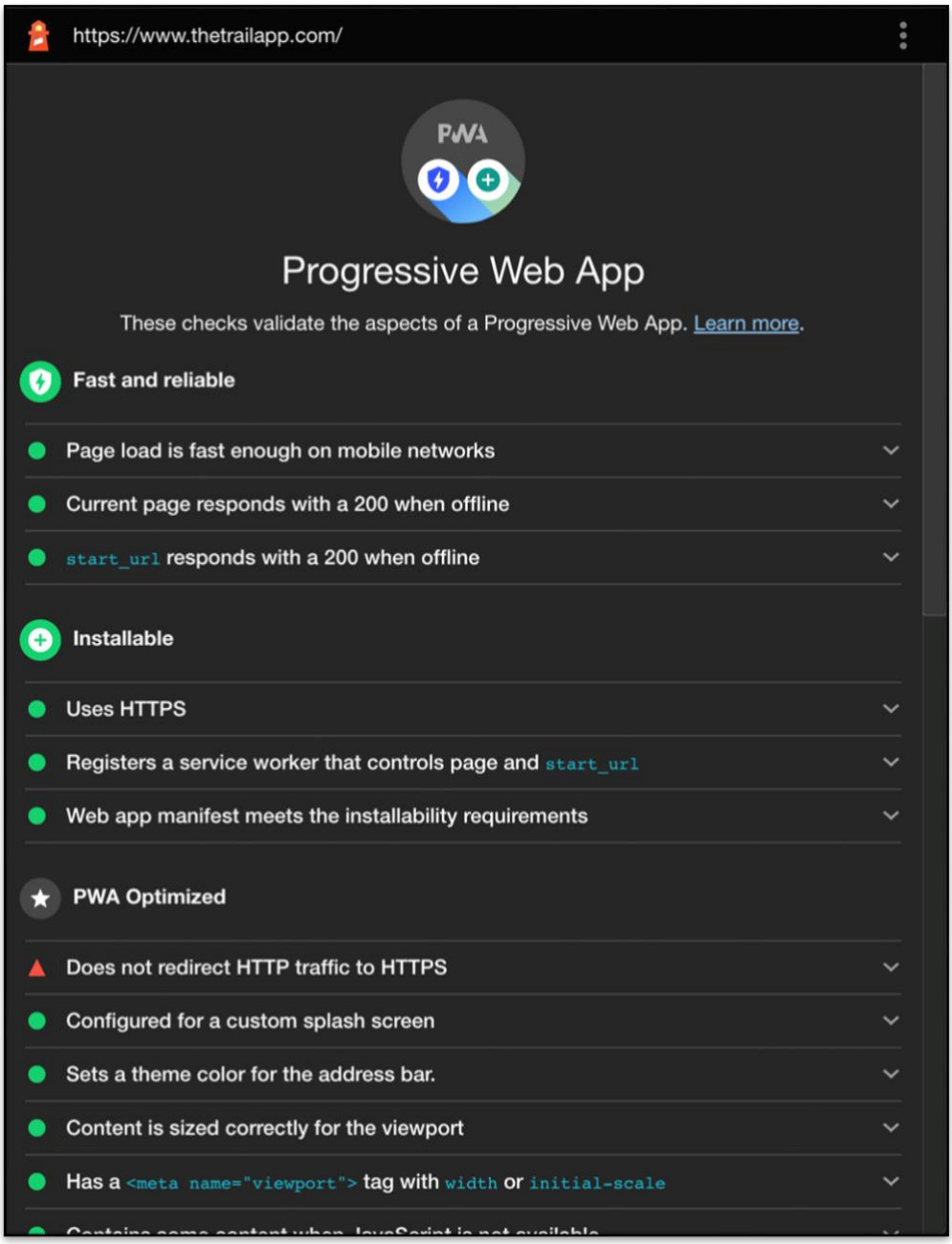


Figure 57 - PWA Lighthouse Test Results

2.7 Geolocation Cross-Device Test Sample

tick	lat	long
2021-01-18 17:25:04.239552	55.94365	-3.2114856
2021-01-18 17:25:04.349052	55.943653	-3.2114851
2021-01-18 17:25:04.472816	55.94365	-3.2114854
2021-01-18 17:25:04.667406	55.943653	-3.2114606
2021-01-18 17:25:06.683254	55.94364	-3.2114635
2021-01-18 17:25:07.480799	55.943626	-3.2114463
2021-01-18 17:25:08.486564	55.943623	-3.211447
2021-01-18 17:25:09.471482	55.943604	-3.211423
2021-01-18 17:25:10.481747	55.943596	-3.2114134
2021-01-18 17:25:11.484343	55.943577	-3.2114055
2021-01-18 17:25:12.476772	55.943558	-3.2113957
2021-01-18 17:25:13.523054	55.943546	-3.2113965
2021-01-18 17:25:14.485913	55.94352	-3.2113628
2021-01-18 17:25:15.471461	55.943504	-3.2113585
2021-01-18 17:25:16.508514	55.943478	-3.2113566
2021-01-18 17:25:17.471484	55.943466	-3.211361
2021-01-18 17:25:18.47647	55.94346	-3.2113562
2021-01-18 17:25:19.470213	55.943447	-3.2113254
2021-01-18 17:25:20.482198	55.943443	-3.2113085
2021-01-18 17:25:21.516577	55.943428	-3.2113023
2021-01-18 17:25:22.496457	55.943417	-3.2112963
2021-01-18 17:25:23.479542	55.943394	-3.2112937
2021-01-18 17:25:24.498956	55.943375	-3.211287
2021-01-18 17:25:25.470764	55.94336	-3.2112854
2021-01-18 17:25:26.477213	55.943344	-3.2112737
2021-01-18 17:25:27.470564	55.94333	-3.2112691
2021-01-18 17:25:28.489482	55.943306	-3.2112465
2021-01-18 17:25:29.47849	55.943287	-3.211248
2021-01-18 17:25:30.484325	55.94327	-3.2112348
2021-01-18 17:25:31.470594	55.943268	-3.2112145
2021-01-18 17:25:32.471872	55.94326	-3.2112029
2021-01-18 17:25:33.42767	55.943253	-3.211152
2021-01-18 17:25:34.392984	55.943226	-3.2112098
2021-01-18 17:25:35.332824	55.94318	-3.2112098
2021-01-18 17:25:36.279539	55.943165	-3.2112029
2021-01-18 17:25:37.244725	55.943146	-3.2111773
2021-01-18 17:25:38.225921	55.943123	-3.2111979
2021-01-18 17:25:39.221653	55.943123	-3.2111835
2021-01-18 17:25:40.222305	55.943123	-3.2111583
2021-01-18 17:25:41.229224	55.9431	-3.2111433
2021-01-18 17:25:42.226378	55.94309	-3.2111177
2021-01-18 17:25:43.264725	55.943073	-3.2111156
2021-01-18 17:25:44.229392	55.94306	-3.2110982
2021-01-18 17:25:45.227415	55.943054	-3.2110918
2021-01-18 17:25:46.247852	55.94304	-3.211073
2021-01-18 17:25:47.220981	55.94302	-3.211068
2021-01-18 17:25:48.238416	55.94302	-3.2110608
2021-01-18 17:25:49.267777	55.942997	-3.211054
2021-01-18 17:25:50.22277	55.942978	-3.2110448
2021-01-18 17:25:51.2247	55.942963	-3.2110426
2021-01-18 17:25:52.230395	55.942955	-3.2110412
2021-01-18 17:25:53.245504	55.942947	-3.2110348
2021-01-18 17:25:54.238565	55.94294	-3.2110367
2021-01-18 17:25:55.248137	55.942924	-3.211005
2021-01-18 17:25:56.239798	55.942913	-3.2109685
2021-01-18 17:26:01.466683	55.94289	-3.2108004
2021-01-18 17:26:02.222318	55.94289	-3.210771
2021-01-18 17:26:15.097578	55.942966	-3.2103992

2.8 QR Code Creation using QRmonkey

Figure 58 - QR Code Creation for Tree 1

2.9 Tree Geojson File Content (HWU Trees)


```
{
  "type": "Feature",
  "properties": {
    "Id": "2",
    "CommonName": "Silver birch",
    "LatinName": "Abies nordmanniana",
    "Height": "20ft",
    "Age": "",
    "Description": "Some description about silver burches, this is a group of them",
    "Origin": "Europe and Eurasia",
    "icon": "/static/img/tree2.svg"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      -3.319175129045629,
      55.90929208976373
    ]
  }
},
{
  "type": "Feature",
  "properties": {
    "Id": "3",
    "CommonName": "Nordmann fir",
    "LatinName": "Abies nordmanniana",
    "Height": "",
    "Age": "",
    "Description": "CHRISTMAS TREE at roundabout",
    "Origin": "Mountains south and east of the Black Sea",
    "icon": "/static/img/tree3.svg"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      -3.3189412418215265,
      55.90951134603756
```


Figure 59 - HWU Tree Geojson Sample

2.10 Augmented Reality Bug/Limitation

Figure 60 - AR Bug - Model Loading Incorrectly

2.11 Evaluation Test Site Location Map

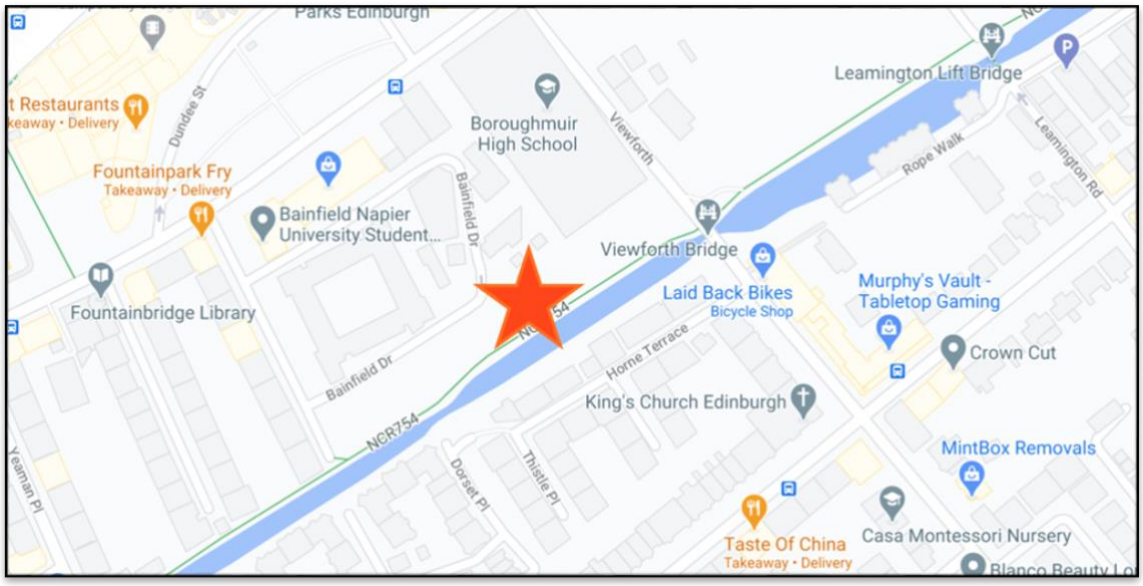


Figure 61 - Physical Test Site Location

3. Source Code Appendices

3.1 Dynamic Greeting JavaScript

```
<script>
    //CODE TO GENERATE DYNAMIC GREETING
    var myDate = new Date();//gets current date
    var hrs = myDate.getHours();//gets the current time
    var greet;

    if (hrs < 12)
        greet = 'Good Morning'; //if it is before 12PM then it is morning
    else if (hrs >= 12 && hrs <= 16)//if it is between 12PM and 4PM it is afternoon
        greet = 'Good Afternoon';
    else if (hrs >= 16 && hrs <= 24)//if it is after 4PM and before 12am is is evening
        greet = 'Good Evening';

    document.getElementById('greeting').innerHTML = greet; //display the result (desktop)
    document.getElementById('greeting2').innerHTML = greet; //display the result (mobile)
</script>
```

Figure 62 - Dynamic Greeting Source Code

3.2 Random JSON Fact Selector Code

```
//HANDLES OBTAINING AND DISPLAYING A RANDOM FACT (every 20 seconds)
function fact() {
    //open a new ajax request and return the resultant data
    var json = (function () {
        var json = null;
        $.ajax({
            'async': false,
            'global': false,
            'url': "/static/facts.json", //URL to the local json file containing the facts
            'dataType': "json",
            'success': function (data) {//if the data is retrieved then run this function and store the data
                json = data;
            }
        });
        return json;
    })();
    let randomIndex = Math.floor(Math.random() * 25) //generate a random number between 0 and the number of facts (25)
    document.getElementById('fact').innerHTML = json[randomIndex]; //get the fact at that random index and display it (mobile)
    document.getElementById('fact2').innerHTML = json[randomIndex];//get the fact at that random index and display it (desktop)
}

fact();//initial function call

//call the function every 20 seconds (update fact every 20 seconds)
setInterval(function () {
    fact()
}, 20000)
```

Figure 63 - JSON Fact Selector Source Code

```
{} facts.json > ...
{
    "0": " Earth has more than 60,000 known tree species.",
    "1": " Earth has more than 60,000 known tree species.",
    "2": " Since 1969 over a quarter of a million trees have been planted on the Heriot-Watt University Campus",
    "3": " Trees didn't exist for the first 90 percent of Earth's history.",
    "4": " Before trees, Earth was home to fungi that grew 26 feet tall.",
    "5": " The first known tree was a leafless, fern-like plant from New York.",
    "6": " Some trees emit chemicals that attract enemies of their enemies.",
    "7": " Trees in a forest can 'talk' and share nutrients through an underground internet built by soil fungi.",
    "8": " Most tree roots stay in the top 18 inches of soil, but they can also grow above ground or dive a few hundred feet deep.",
    "9": " A large oak tree can consume about 100 gallons of water per day, and a giant sequoia can drink up to 500 gallons daily.",
    "10": " The oldest trees on the Heriot-Watt Campus are the native or naturalised hardwoods of beech, ash, sycamore and particularly the Riccarton Sweet Chestnut (also known as the Spa
    "11": " Adding one tree to an open pasture can increase its bird biodiversity from almost zero species to as high as 80.",
    "12": " In recent years, Heriot-Watt University has achieved an environmentally sustainable campus.",
    "13": " Foxes, rabbits, hares, swans, hedgehogs, moles, weasels, stoats and grey squirrels are all thriving in the Heriot-Watt Campus.",
    "14": " Trees that grow in humid places or near large bodies of water have broad, big leaves. Trees that grow in dry environments have small, hard leaves, which reduce water loss.",
    "15": " Bark of trees that grow in the shade is often thin, while bark of trees that grow in sunny places is thicker.",
    "16": " Cottonwood seeds are able to stay airborne for days, which is much longer than any other type of seed.",
    "17": " The most poisonous tree in the world is the manchineel tree, which is native to Florida. If eaten, its fruit can kill a person.",
    "18": " A bristlecone pine tree nicknamed Methuselah is believed to be the oldest tree in the world. Its exact location is kept a secret to protect it from vandals and tourists.",
    "19": " Ancient pagan cultures, such as the Celts, believed that benevolent and helpful spirits lived in trees. Knocking on tree trunks roused a spirit for protection, which led to th
    "20": " There are are 3.04 trillion trees on Earth, or about 422 for each person.",
    "21": " Since the start of human civilization, about 11,700 years ago, the total number of trees on Earth has fallen by around 46%.",
    "22": " About 15 billion trees are lost each year due to deforestation, forest management, and changes in land use.",
    "23": " The Royal Botanic Garden Edinburgh (RBGE) has donated 21 trees of eight species to Heriot-Watt University including the rare Catacol whitebeam (Sorbus pseudomeinichii).",
    "24": " During the Second World War, Riccarton became an army base and afterwards, it became a resettlement camp for ex-prisoners of war then headquarters of the Royal Artillery 3rd A
    "25": " The woodland surrounding the Lawn on Heriot-Watt Campus still contains several specimen trees from the collection and some of the exotic species are dated between 100 and 150
}
```


Figure 64 - Tree Facts

3.3 manifest.webmanifest file and Icon


```
{} manifest.webmanifest > ...
{
  "name": "Trail.",
  "short_name": "Trail.",
  "theme_color": "#00b800",
  "background_color": "#ffffff",
  "display": "standalone",
  "orientation": "portrait",
  "Scope": "/",
  "start_url": "https://www.thetrailapp.com/",
  "icons": [
    {
      "src": "/static/img/icon-72.png",
      "sizes": "72x72",
      "type": "img/png",
      "purpose": "any maskable"
    },
    {
      "src": "/static/img/icon-128.png",
      "sizes": "128x128",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/static/img/icon-144.png",
      "sizes": "144x144",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/static/img/icon-152.png",
      "sizes": "152x152",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/static/img/android-chrome-192x192.png",
      "sizes": "192x192",
      "type": "image/png",
      "purpose": "any maskable"
    },
    {
      "src": "/static/img/android-chrome-512x512.png",
      "sizes": "512x512",
      "type": "image/png",
      "purpose": "any maskable"
    }

  ]
}
```


Figure 65 - Web Manifest File

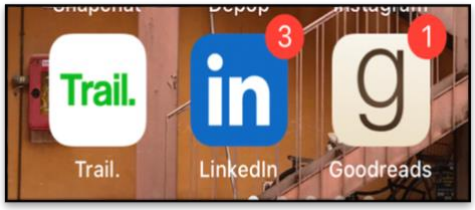


Figure 66 - Trail Icon on iOS

3.4 Service Worker.js Code – Caching and Fetching Files

```
3   self.addEventListener('install', function(event) { //when the SW is registered this install function is executed
4       event.waitUntil(//opens the cache with the name defined on line 34 and caches the list of static files
5           caches.open(CACHE_NAME)
6           .then(cache => {
7               return cache.addAll(cacheurls);
8           })
9       );
10  });
11
12  self.addEventListener('activate', function(event) { //handles the activation of the sw (after registering)
13      console.log("NEW Service Worker Activated :)");
14    });
15
16
17    //cache first fetch function
18  self.addEventListener('fetch', event => {//waits till a file is requested
19      console.log('[Trail - ServiceWorker] Fetch event fired.', event.request.url);
20      event.respondWith(
21          caches.match(event.request).then(function(response) {//if the file is in the cache then it retrieves it
22              if (response) {
23                  console.log('[Trail - ServiceWorker] Retrieving from cache...');
24                  return response;
25              }
26              console.log('[Trail - ServiceWorker] Retrieving from URL...');//if it is not in the cache then it uses the network
27              return fetch(event.request).catch(function (e) {
28                alert('You appear to be offline, please try again when back online');
29              });
30          })
31      );
32  });
33
34    const CACHE_NAME = 'Trail-cache-v3'; //defines the name of the cache
35    const cacheurls = [ //an array consistic of the static files to be cached
36        './',
37        '/static/edi.geojson',
```

Figure 67 - Service Worker Source Code

3.5 Geolocation Test Code

```
//this function is used for testing geolocation updates
router.GET("/location/:lat/:long", func(c *gin.Context) {
    lat := c.Param("lat")
    long := c.Param("long")
    if _, err := db.Exec("CREATE TABLE IF NOT EXISTS ticks (tick timestamp, lat real, long real)"); err != nil {
        c.String(http.StatusInternalServerError,
            fmt.Sprintf("Error creating database table: %q", err))
        return
    }
    //this line adds the current timestamp and the latitude and longitude into the database
    if _, err := db.Exec("INSERT INTO ticks VALUES (now(),$1, $2)", lat, long); err != nil {
        c.String(http.StatusInternalServerError,
            fmt.Sprintf("Error incrementing tick: %q", err))
        return
    }
})
```

Figure 68 - Geolocation Test Code

3.6 Geofence Back-end Code

```golang
router.POST("/geofence/:lat/:long/:visited", func(c *gin.Context) {
    lat := c.Param("lat")
    long := c.Param("long")
    visited := c.Param("visited")

    //check if the lat and long is inside the goefence
    rows, err := db.Query("SELECT id, treename, latinname, height, age, description, origin, img
    FROM trees WHERE ST_DWithin ( geography (ST_Point(longitude,latitude)), geography (ST_Point($1, $2)), 20) AND id != $3 limit 1", long, lat, visited)
    if err != nil { //throws error if unsuccessful
        c.String(http.StatusInternalServerError,
            fmt.Sprintf("Error reading trees: %q", err))
        return
    }
    defer rows.Close() //keeps query result open
    //define variables for query result to go
    var name string
    var latinname string
    var height int
    var age int
    var description string
    var origin string
    var img string
    var id string
    var success bool = false

    //loop through the results putting the values into the variables defined above
    for rows.Next() {
        if err := rows.Scan(&id, &name, &latinname, &height, &age, &description, &origin, &img); err != nil {
            c.String(http.StatusInternalServerError,
                fmt.Sprintf("Error scanning trees: %q", err))
            return
        }
        success = true
    }
    //stores variable values in a json object
    treeJson := TreeJson{
        Id:          id,
        Name:        name,
        Latinname:   latinname,
        Height:      height,
        Age:         age,
        Description: description,
        Img:         img,
    }

    js, err := json.Marshal(treeJson) //encodes the json
    if success == true && id != visited { //returns the json to the front end
        c.JSON(200, string(js))
    } else {
        c.JSON(200, "null")
    }
})
```

Figure 69 - Geofence Backend Code (Golang)

3.7 QR Scanner Source Code – Front-end and Backend

```
function validURL(str) {
var pattern = new RegExp('^(https?:\\/\\/)?'+ // the protocol
    '((([a-z\\d]([a-z\\d-]*[a-z\\d])*)\\.)+[a-z]{2,}|'+ // the domain name
    '((\\d{1,3}\\.){3}\\d{1,3}))'+ // OR ip address
    '(\\:\\d+)?(\\/[-a-z\\d%_.~+]*)*'+ // port and path
    '(\\?[;&a-z\\d%_.~+=-]*)?'+ // query
    '(\\#[-a-z\\d_]*)?$','i'); // fragment locator
return !!pattern.test(str);
}
function startScanning() {
    console.log("Scan beginning")
    var results = document.getElementById('scanned-result');//define QR variables
    var lastMessage;
    var codesFound = 0;
    function onScanSuccess(qrCodeMessage) {//if a QR code is found
        var button = document.getElementById('start');
        if (lastMessage !== qrCodeMessage) {//checks if the last scanned item is not the same as the current one
            lastMessage = qrCodeMessage;//updates the last scanned message with the current one
            if (validURL(qrCodeMessage)) {//checks the QR code content using the REGEX function to make
            //sure it is a URL
                stopScanning();//stop scanning if it is valid
                scanning = false;
                button.disabled = false;
                button.innerHTML = "Start Scanning";
                button.setAttribute('class','btn btn-success');
                //This code makes sure that users cannot scan malicious URLS
                //i.e. the QR code can only be for the trail app domain
                if (qrCodeMessage.includes("thetrailapp")){
                window.location.href = qrCodeMessage;//redirects the user to that page
                }else{
                    alert("Sorry, you can only scan QR codes designed for the Trail application");
                }
            }
        }
    }
```

Figure 70 - QR Scanner Front-end Code

```
//The following section of code handles the event in which a user scans a QR code of a specific tree (variable is passed in ? url param)
treeNum := c.Query("id")
fmt.Println("Tree ID is ?", treeNum)

//if the variable value is not null then pull all tree info corresponding to that id from database
if treeNum != "" {
    rows, err := db.Query("SELECT treename, latinname, height, age, description, origin, img FROM trees WHERE id = $1", treeNum)
    if err != nil {
        c.String(http.StatusInternalServerError,
            fmt.Sprintf("Error reading trees: %q", err))
        return
    }
    defer rows.Close()

    var name string
    var latinname string
    var height int
    var age int
    var description string
    var origin string
    var img string
    //loop through the data recieved from the SELECT statement and store in the specific variables above
    for rows.Next() {
        if err := rows.Scan(&name, &latinname, &height, &age, &description, &origin, &img); err != nil {
            c.String(http.StatusInternalServerError,
                fmt.Sprintf("Error scanning trees: %q", err))
            return
        }
    }
    //serve the tour page, passing in the tree info variables
    c.HTML(http.StatusOK, "tour.html", gin.H{"navtitle": "Tour.",
        "qr":          true,
        "id":          treeNum,
        "treename":    name,
        "latinname":   latinname,
        "height":      height,
        "age":         age,
        "description": description,
        "origin":      origin,
        "img":         img,
    })
} else { //if the QR code has no variable assigned to it just load the tour page
    c.HTML(http.StatusOK, "tour.html", gin.H{"navtitle": "Tour."})
}
```

Figure 71 - QR Scanner Backend Handling (Golang)

3.8 Augmented Reality Full Page Code

```
<!-- Import AR.js Library Scripts-->
<script src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
<script src="https://unpkg.com/aframe-look-at-component@0.8.0/dist/aframe-look-at-component.min.js"></script>
<script src="https://raw.githack.com/AR-js-org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>


<body style="margin : 0px; overflow: hidden;">

    <!-- Creates an AR "scene" where objects and text can be added-->
    <a-scene
        vr-mode-ui="enabled: false"
        embedded
        arjs="sourceType: webcam; debugUIEnabled: false;"

    >

    <!--defines some text that is displayed via AR at a specific geolocation-->
        <a-text
        value="Tree 5: Rowan
        Latin Name: Sorbus aucuparia
        Age: 3yrs
        Height: 10ft
        Origin: Native"
        look-at="[gps-camera]"
        scale="1.5 1.5 1.5"
        gps-entity-place="latitude: 55.94371572996293; longitude: -3.211351372713154;"
        ></a-text>

        <a-entity
        gltf-model="/static/ar/adore.glb"
        look-at="[gps-camera]"
        scale="0.5 0.5 0.5"
        gps-entity-place="latitude: 55.894139976579304; longitude: -3.1551469589806995;"
    ></a-entity>

        <a-entity
        gltf-model="/static/ar/tree5.glb"
        look-at="[gps-camera]"
        scale="0.25 0.25 0.25"
        gps-entity-place="latitude: 55.9406; longitude: -3.2138;"
    ></a-entity>

    <a-marker preset="hiro" id="anchor">
        <a-entity
        gltf-model="/static/ar/tree5.glb"
        scale="0.25 0.25 0.25"
        position="15 -5 0"
    ></a-entity>
    </a-marker>

        <a-camera gps-camera rotation-reader> </a-camera>
    </a-scene>

<script>
 </script>
 </body>
```

Figure 72 - Augmented Reality Source Code

3.9 Map Click Event Code

```
// Handles click event on a tree marker, pulls matching tree data from json and renders it in a pop-up modal on screen
map.data.addListener('click', function (event) {
    var tree_id = event.feature.getProperty('Id');//pulls data from geojson
    var tree_name = event.feature.getProperty('CommonName');
    var latin_name = event.feature.getProperty('LatinName');
    var tree_age = event.feature.getProperty('Age');
    var tree_height = event.feature.getProperty('Height');
    var tree_description = event.feature.getProperty('Description');
    var tree_origin = event.feature.getProperty('Origin');
    var tree_img = event.feature.getProperty('Src');
    $('#tree_title').html(tree_id + ": " + tree_name);
    $('#tree_img').attr("src", tree_img);
    $('#tree_age').html(tree_age);
    $('#latin_name').html(latin_name);
    $('#tree_height').html(tree_height);
    $('#tree_info').html(tree_description);
    $('#tree_origin').html(tree_origin);
    $('#myModal').modal('show');
});
```

Figure 73 - Map Click Event Source Code

3.10 Route Path Source Code

```
//creates a route on the map for the tree trail on campus
var route = new google.maps.MVCArray();
route.push( new google.maps.LatLng(55.90934378036286, -3.3200149819795204));
route.push( new google.maps.LatLng(55.90931070579712, -3.3196716592317315));
route.push( new google.maps.LatLng(55.9098836129535, -3.31838160597292));
route.push( new google.maps.LatLng(55.91025165524692, -3.318836591534194));
route.push( new google.maps.LatLng(55.91056910722812, -3.3190569477858607));
route.push( new google.maps.LatLng(55.910668491360795, -3.318790110129424));
route.push( new google.maps.LatLng(55.91058165087567, -3.318569753877696));
route.push( new google.maps.LatLng(55.91079199749353, -3.318130762907456 3));
route.push( new google.maps.LatLng(55.91115479449, -3.3185129432816702));
route.push( new google.maps.LatLng(55.91117698674892, -3.318714362668015));
route.push( new google.maps.LatLng(55.91058701704161, -3.3195850717041306));
route.push( new google.maps.LatLng(55.91011244014375, -3.3205811970731434));
route.push( new google.maps.LatLng(55.90954824221835, -3.32181505061713));
route.push( new google.maps.LatLng(55.90948593734672, -3.3216774250089998));
route.push( new google.maps.LatLng(55.90947703664342, -3.3215636192086158));
route.push( new google.maps.LatLng(55.909221882279844, -3.3221538213827024));
route.push( new google.maps.LatLng(55.90912842415683, -3.3221855811409484));
route.push( new google.maps.LatLng(55.90873678765671, -3.323082794330757));
route.push( new google.maps.LatLng(55.908917771677196, -3.3237179894956927));
route.push( new google.maps.LatLng(55.908672030887786, -3.326161712301503));
route.push( new google.maps.LatLng(55.90884349491923, -3.326229296153743));
route.push( new google.maps.LatLng(55.90927414542232, -3.3241128544651755));
route.push( new google.maps.LatLng(55.90951139991695, -3.3243867469141146));
route.push( new google.maps.LatLng(55.909914131095896, -3.323006613510477));
route.push( new google.maps.LatLng(55.90973868436378, -3.3227931908191923));
route.push( new google.maps.LatLng(55.90944560680215, -3.3234974857004302));

//defines the polyLine attributes
var routeOptions = {
    path: route,
    strokeColor: '#35bef0',
    strokeWeight: 6,
    strokeOpacity: 1,
};
var polyLine = new google.maps.Polyline(routeOptions);
//draws the line
polyLine.setMap(map);
```

Figure 74 - Route Path Creation Code

3.11 Lost Detector Source Code – Front end and Backend

```
$.ajax({//send location to backend
    url: "https://www.thetrailapp.com/route/" + lat + "/" + lng,
    method: "POST",
    contentType:'application/json&apos',
    dataType: 'json',
    cache: false,
    scriptCharset: 'utf-8',
}). done (function (jdata) {
    counter++;//use a counter to display the data every 5 seconds
   var data = $.parseJSON(jdata);
   if(data == false && counter%5 == 0){//if the user is not inside the geofence alert them
       alert("Lost? \nYou are not in the vicinity of a Route.\n Please make your way to Heriot-Watt Campus or the Test Area");
   }
```

Figure 75 - Lost? Detector Front-end Code

```
//this function checks whether a user is within the geofence of a route - if not it tells them they are not near a route
router.POST("/route/:lat/:long", func(c *gin.Context) {
    lat := c.Param("lat")
    long := c.Param("long")
    result := false
    var id int
    rows, err := db.Query("SELECT id FROM boundary WHERE ST_DWithin ( geography (ST_Point(longitude,latitude)), geography (ST_Point($1, $2)),
    if err != nil {
        c.String(http.StatusInternalServerError,
            fmt.Sprintf("Error reading trees: %q", err))
        return
    }
    defer rows.Close()
    for rows.Next() {
        if err := rows.Scan(&id); err != nil {
            c.String(http.StatusInternalServerError,
                fmt.Sprintf("Error scanning trees: %q", err))
            return
        }
        result = true
    }
    c.JSON(200, result)
```

Figure 76 - Lost Detector Backend Code

3.12 Mobhide and Deskhide CSS Media Query

```
/*Deals with hiding and showing elements based on the size of the screen for mob and desk usage */
    @media (max-width: 992px) {
     .mobhide {
       display: none !important;
     }
   }

    @media (min-width: 992px) {
      .deskhide {
        display: none !important;
      }
    }
```

Figure 77 - Mobhide and Deskhide CSS

4. Evaluation Study Appendices

4.1 Lighthouse Performance Improvements

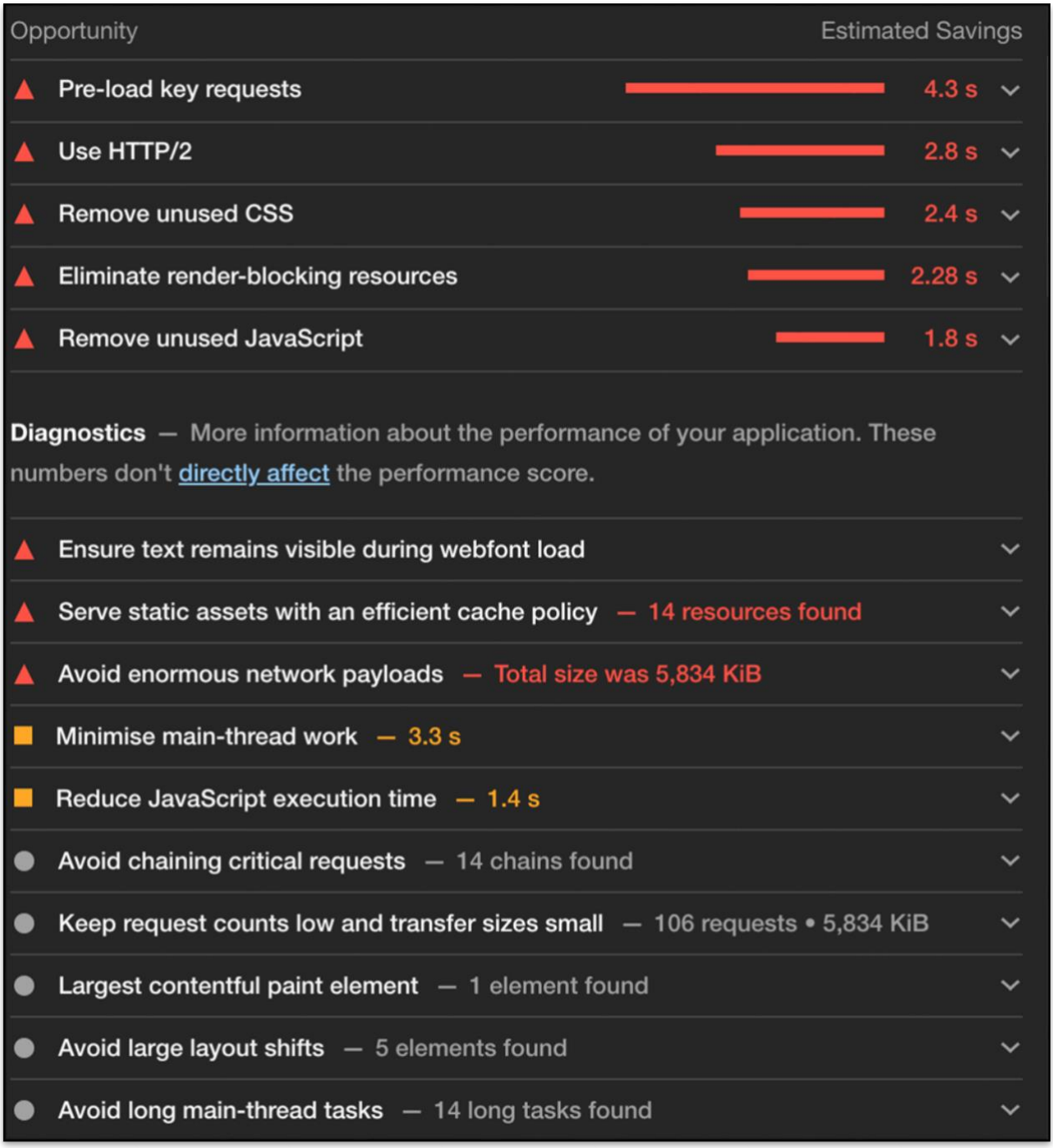


Figure 78 - Lighthouse Test Performance Output

4.2 Consent Form

TRAIL USABILITY STUDY

Consent Form

Heriot-Watt University

CONSENT TO ACT AS A SUBJECT IN AN EXPERIMENTAL STUDY

PRINCIPAL INVESTIGATOR: [MOLLICA, RUARIDH]

DESCRIPTION: The purpose of this study is to acquire feedback from a range of participants about the general usability of the application

PERSONAL DATA COLLECTED: *Age, Gender*
There are minimal risks for you to participate in this study. All personal information will be kept in accordance with the provisions of the GDPR. Your participation will not affect how well you do in your courses (if you are a student) or affect your relationship with the university in any way.

You are free to decline to participate in this study. Should you decide to participate, you are free to end your participation at any time. Such a decision by you will not adversely affect or alter your status with the university in any way. You are also free to withdraw 7 days after the study (please email **rm141@hw.ac.uk**). If you withdraw, your data will be removed and destroyed.

PARTICIPATION VOLUNTARY CONSENT: *I certify that I have read the preceding and that I understand its contents. Any questions I have pertaining to the research have been answered satisfactorily by the team. My signature below means that I have freely agreed to participate in this study.*

.

_____ _____

_____

Date **Subject Signature**
Investigator Initials **Participant Code [_ _ _]**

✂ .....

INVESTIGATOR'S CERTIFICATION: I certify that I have explained to the above individual the nature and purpose, the potential benefits, and possible risks associated with participation in this research study, have answered any questions that have been raised, and have witnessed the above signature.

_____ _____

_____

Date **Subject Signature**
Investigator Initials **Participant Code [_ _ _]**

Figure 79 - Study Consent Form

4.3 Pre-Questionnaire

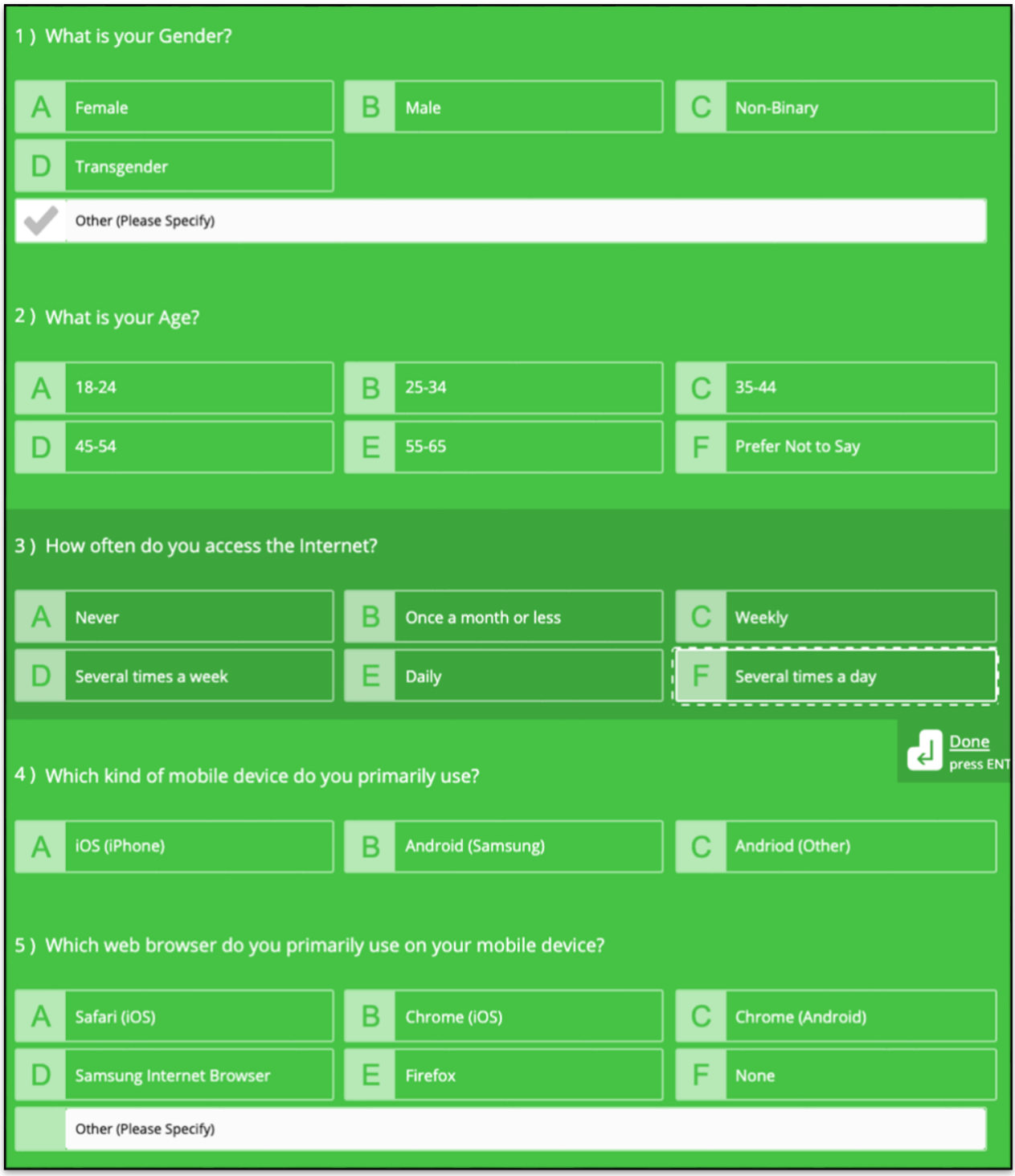


Figure 80 - Demographics Survey Questions

4.4 Tasks and Post-Questionnaire

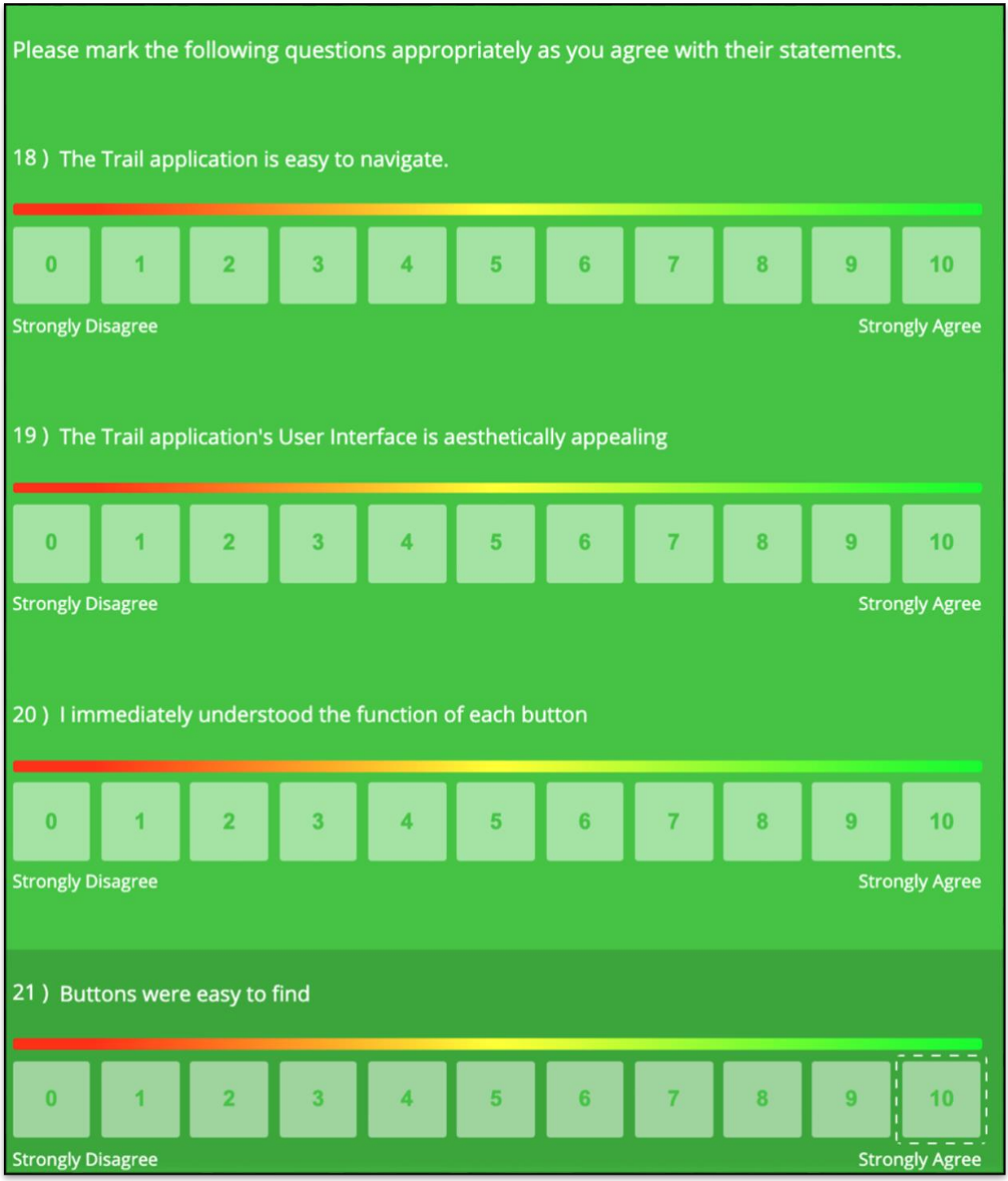


Figure 81 - Study Tasks and Post Questionnaire

4.5 Physical Tasks Questionnaire Example

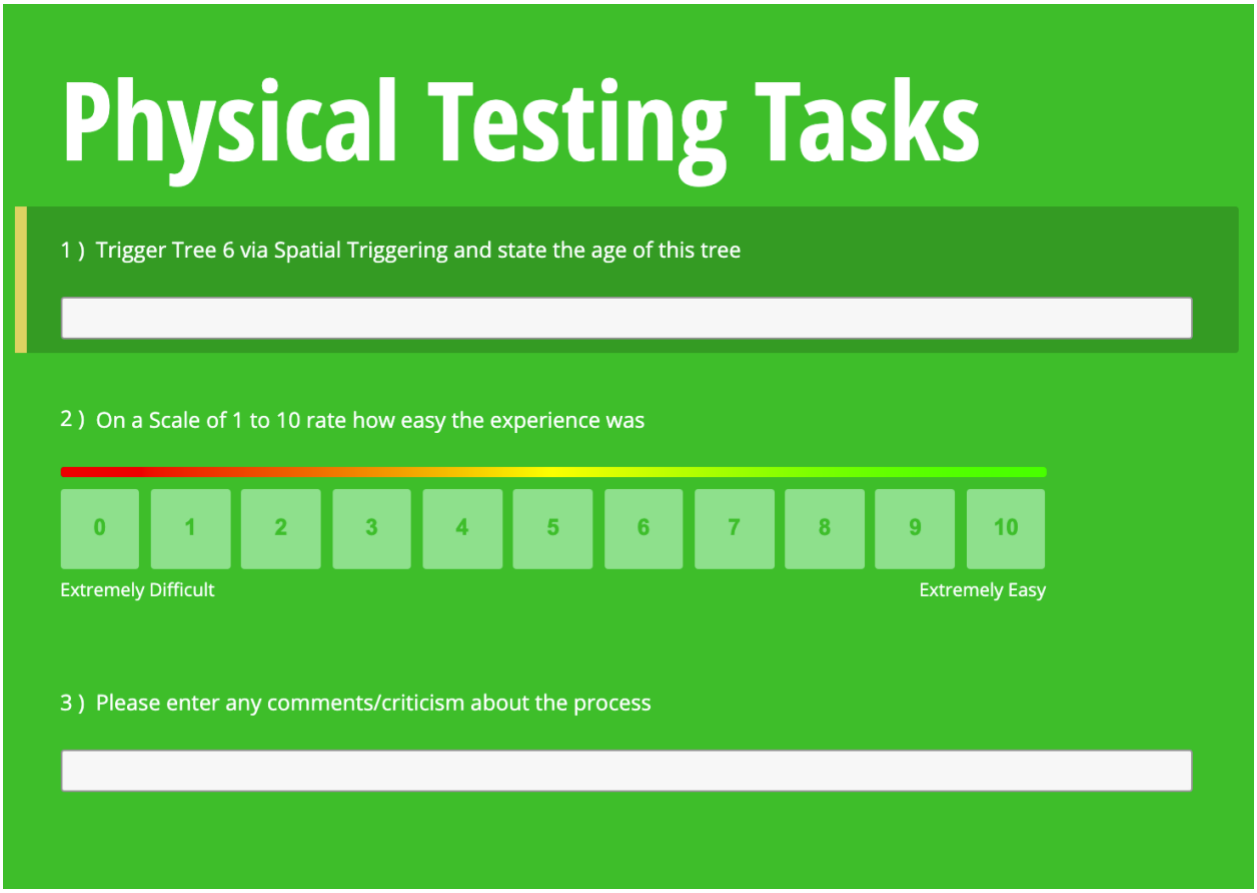


Figure 82 - Physical Tasks Survey

4.6 QR Scanning Comments

Very easy to use, fast response
had to go to incognito mode
high tech stuff
easy
If i didnt know what to look for, i wouldnt have known what the QR icon looked like
extremely easy to find and navigate.
cool QR code!
it was easy
N/A
Scanning was easy but it took a second to find the scan page
Page was easy to find because of the QR code icon. Nice big 'start scanning button'.

Figure 83 - QR Scanning Optional Comments/Criticism

4.7 Finding Tree 1 Comments

Map works smoothly and the numbers are easy to read
Simple
Incorrect tree icon loaded - no number 1? Also clusters didnt show (duck duck go browser)
trees did not appear on the campus
map was slow
My phone screen is small so things felt a little cramped with all the trees on the map
map took a long time to load
Map was a bit slow
N/a
map was a bit slow to load
Initially i didnt understand the clusters were for trees. Finding Tree 1 was okay

Figure 84 - Comments from "Find Tree 1" Task

4.8 SPEAK Button Likert Scale Results

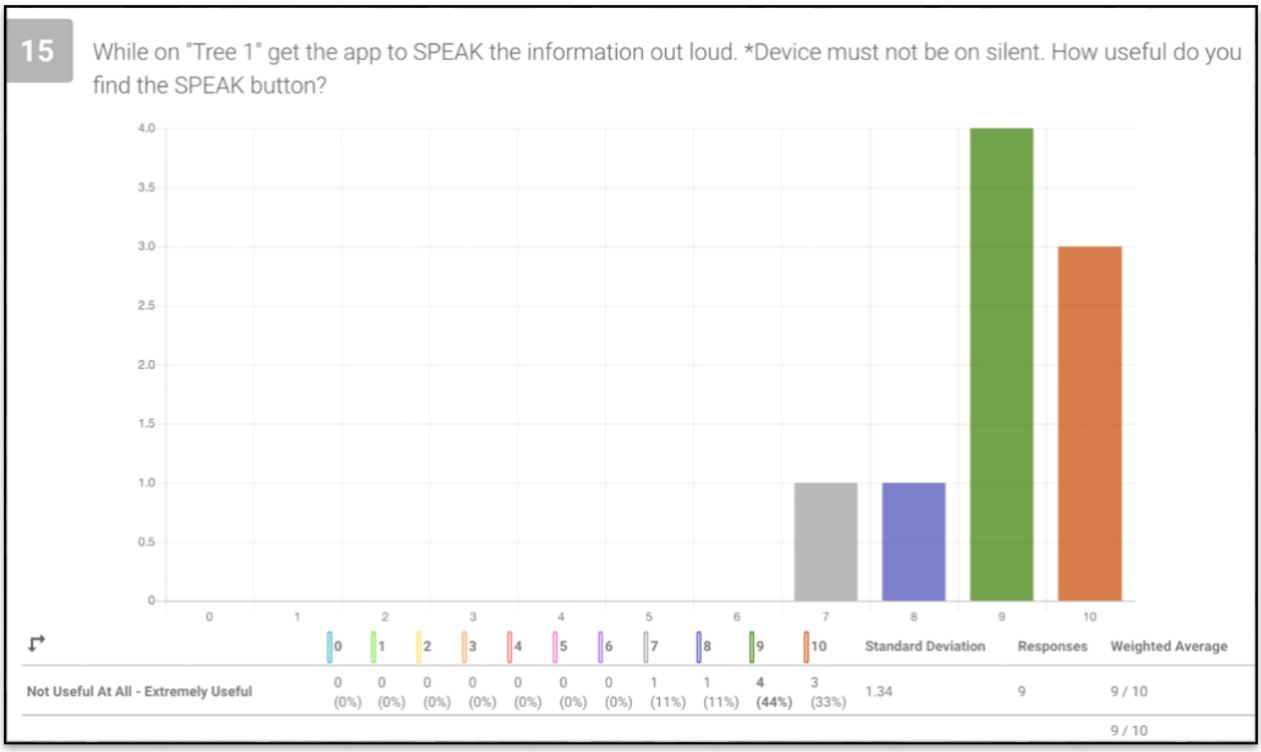


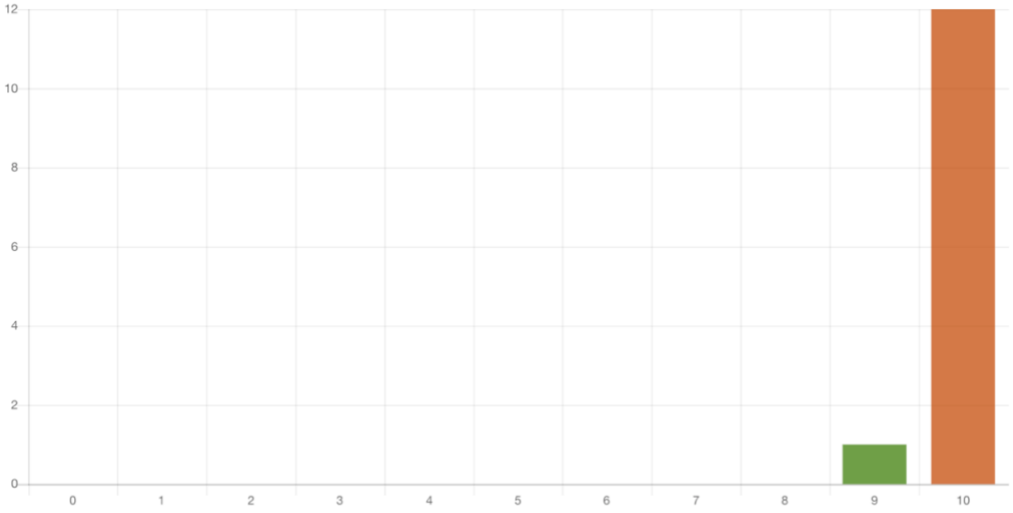
Figure 85 - SPEAK Button Task Results

4.9 SPEAK Button Comment/Criticism Results

Figure 86 - SPEAK Button Optional Comments

4.10 Begin the Tour Task Likert Results

17 From the Home Page, please begin a tour. On a scale of 1 to 10, how easy was it to begin a tour?

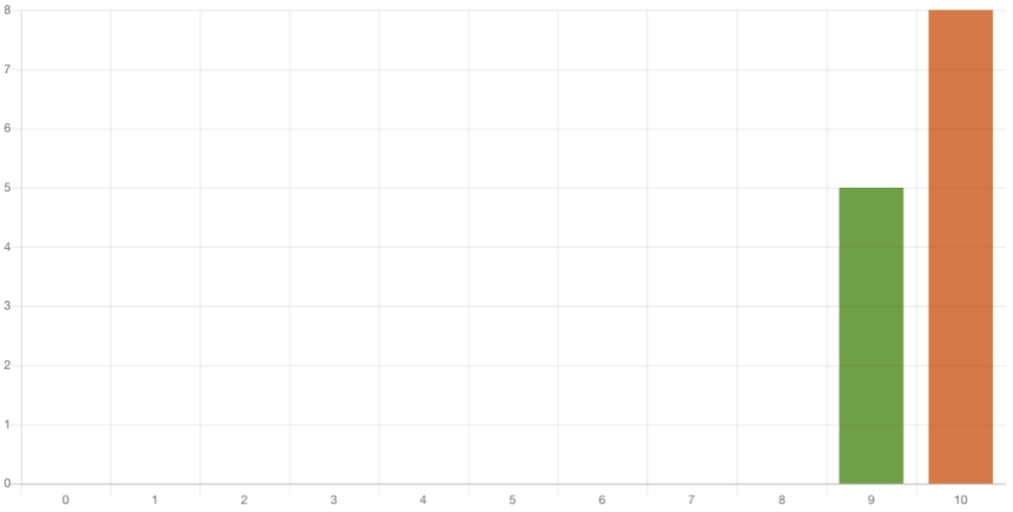


	0	1	2	3	4	5	6	7	8	9	10	Standard Deviation	Responses	Weighted Average
Extremely Difficult - Extremely Easy	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (8%)	**12 (92%)**	3.43	13	9.92 / 10

Figure 87 - Beginning the Tour Task Results

4.11 Current Location Clarity Likert Results

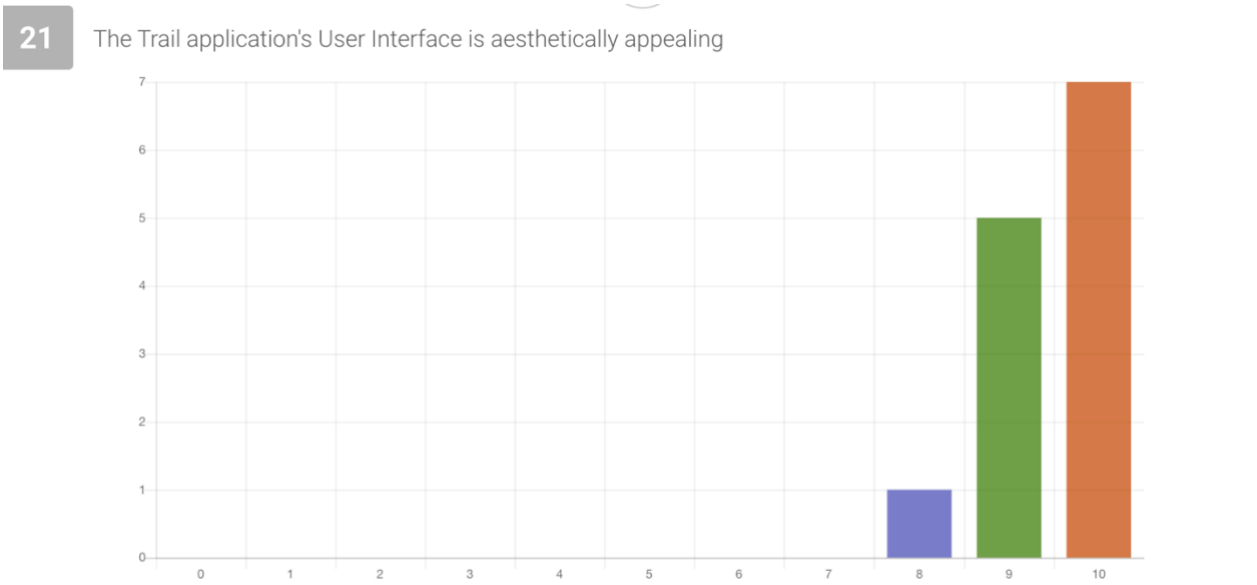
18 When on the Tour Page, is it clear where your current location is? On a scale of 1 to 10 how clear is it to you where your current location is?



	0	1	2	3	4	5	6	7	8	9	10	Standard Deviation	Responses	Weighted Average
Extremely Unclear - Extremely Clear	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	5 (38%)	**8 (62%)**	2.59	13	9.62 / 10
														9.62 / 10

Figure 88 - Clarity of User Location Results

4.12 Aesthetics of Application UI Likert Results



	0	1	2	3	4	5	6	7	8	9	10	Standard Deviation	Responses	Weighted Average
Strongly Disagree - Strongly Agree	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1 (8%)	5 (38%)	**7 (54%)**	2.33	13	9.46 / 10
														9.46 / 10

Figure 89 - Aesthetic Appeal of User Interface Likert Results