Final Year Project Report

Full Unit – Final Report

Offline HTML5 Maps Application

Jack Wearn

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Matthew Hague

Department of Computer Science

Royal Holloway, University of London

March 25, 2021

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 21,367 Words

Student Name: Jack Wearn

Date of Submission: 26/03/2021

Signature: 

Table of Contents

Abstract

Recent developments in the way websites are created and developed has enabled the use of more advanced tools and techniques in the creation of offline web applications. An offline web application is any type of website which maintains (to a degree) the same functionality when there is no Internet connection present for the user. A brief example of such technology is Service Workers, which have developed further on caching to achieve a smoother and more efficient offline experience for users and developers [1]. Combining these tools and techniques with the use of map data, allows for a user to interact with an online map application without the need for a stable Internet connection, which may not always be possible when travelling for example. During the course of this project the map has focuses on the Areas of Outstanding Natural Beauty, otherwise abbreviated as AONB. The reason for this is that when researching what the map application should focus on, there was a clear gap in the market with map applications which offered a tailored experience for these locations within the United Kingdom (UK). Although most of these areas do have their own website, a modern application which houses all of this information would be helpful to get more people informed of them and encourage exploring of these locations, which in turn will deliver more tourists into the area to help with the upkeep of them. Through the use of online documentation, articles and other readings surrounding offline technologies, proof-of-concept programs were created as a way of furthering knowledge of the tool/technology and test how best to achieve the most fluent transition between an online and offline web page. The proof-of-concept programs for the project included a “hello world” website which functions offline, displaying map data from Open Street Map (OSM) onto a web page and a user account system. Future research could delve deeper into the map data more specifically to speed up the process of this transition between online and offline while also ensuring there is no loss of performance when a user loses connection. Within this report, all aspects of the project have been analysed including the proof-of-concept programs, the first term prototype and the end product.

Project Specification

The overall, and most important, aim of this project was to create an offline web application using recent developments, such as HTML5. Open Street Map (OSM) data was used to display the map for the user. This data was also needed to be stored in some way to allow for the user to still access and use it while offline. There were a number of early deliverables which were outlined within the project planning phase, which included the creation of a number of proof-of-concept programs to give a better understanding and more knowledge of the key technologies that was going to work towards building the final product. These proof-of-concept programs included the "hello world" offline HTML5 application, a "to-do list" application using indexedDB, drawing shapes using HTML5 canvas and a web page which was able to load and display OSM data.

During the course of the second term, a number of other proof-of-concept programs were also planned to be created to give a better understanding of some of the more advanced features of the web application. This included a "users accounts" proof-of-concept programs, which tested different languages and techniques to connect a database and have a user be able to create an account and log into that account. Whilst undertaking the development of the project, and all of the proof-of-concept programs, there were reports that had been planned to be created. All of which can be seen in some form within this final report. The reports included basic web development in HTML5, advanced web-based technologies, developing an offline HTML5 application, OSM data representation and Backend Web Development. These reports, which generally became chapters of this report, cover all aspects of the technology that has been used and if they were helpful in the creation of the end product, and if not, why they were not chosen in the end for the final deliverable creation.

The final deliverables of the project were to include a mapping application, which is able to work offline as if it was online, allowing the user to load and display OSM data whilst maintaining functionality. Such functionality includes zooming and panning around the map. As an extension to the project, a user accounts system needed to be setup, which would also allow a user to add their own comments on the AONB locations to give a personal experience for other map users to see when searching through the different locations. The bullet points below outline all of the deliverables for this project:

- Create proof-of-concept programs to better understanding of the key technologies in creating an offline web application using new technology in HTML5.
- Create a fully functioning map application, built around new HTML5 standards, which is able to perform offline and online for the user.
- As an extension to the project, create a user registration and login system which can further enhance the project in the future.
- Also, as an extension, create a user comments system which allows the user to leave their own comment for each of the different AONB locations.

Chapter 1: **Introduction**

1.1 Why this Project is Needed

Although a number of the Areas of Outstanding Natural Beauty (AONB) locations within the United Kingdom (UK) have their own websites, there is a need for a centralised location for all of this information. It is for this reason that the project is needed as it will give people a place to be able to view them all along with the location of them on a map. By having a web application with this information, it will educate more people on the different locations within the UK, but also encourage people to visit somewhere new. Especially coming off the back of the national lockdowns, the map application is even more vital as people will be looking to find new places to visit and explore the outdoors more.

1.2 Personal Project Motivation

The main motivation behind this project is to encourage more people to explore the many places within the UK. From personal experience, a large number of people are unaware of what an AONB is and the significance of these locations. Some people may also not realise the places they are able to explore within their own country, county or even town. By having a web application with all of this information and exact locations showing on a map, it may encourage more people to explore the areas they may not have realised existed. Another major motivation behind the project is to give people a chance to use the map without internet connection. Whilst visiting a number of these AONB locations, Wi-Fi is usually difficult to come across, or Wi-Fi signals are usually very weak. By having a map which can function well offline, a person will be able to better navigate a new area that they are visiting.

Another personal project motivation is that travelling and exploring new places is a personal passion. Therefore, the map application can serve as a personal "checklist" of locations that can be explored and give a better understanding of what can be done in the surrounding areas and landmarks that could be discovered.

1.3 Project Aims and Objectives

The aims of this project are to have a fully functional mapping application, which can work both online and offline. Once this has been achieved, then the aim of the project is to extend on what is already a part of the foundation to a more social experience. This will include the allowance of creating user accounts and having some form of social interaction. This could include comments on the different location's information web pages, or a forum style area in which different users could communicate together regarding the different AONB locations.

The objectives of this project can be broken down into term one and term two milestones. For term one, these can be seen from the project planning phase as set out in the milestones section. This will be covered more extensively later on within this chapter, along with a more thorough breakdown of the term two milestone objectives also. However, a brief outline of all of the required objectives include:

- Gain a better understanding, through the use of reports and proof-of-concept programs, of how modern web applications are creating. This includes more advanced topics of modern web development such as offline capabilities and registering a user.

- Create a fully functioning map application, which can work both online and offline, capable of displaying the different AONB locations.

1.4 Survey of Related Literature

In order to undertake this project, background knowledge and reading was needed to ensure that the project run and progressed as smoothly as possible. A number of helpful resources were discovered and used whilst undertaking this project, to either better understand the feature or coding paradigm that is being used or created. These resources can be categorised into a number of different source material such as: websites, reports, blogs, books, surveys and research papers. This subsection will go into some more details of the related literature and discuss the usefulness when undertaking this project.

Online web-based documentation was vital in the creation of a web application. For example, the "MDN web docs" by Mozilla [2] have a number of in-depth tutorials and documentation on how to use, create and maintain different aspects of a web application. An example of this is their documentation for Caching. Not only did the information assist in the creation of a proof-of-concept application to better understand caching, but also gave an insight into newer and better techniques such as service workers, which are more widely accepted in modern web development for offline web applications. By having this documentation at hand, it gave an opportunity to better understand the capabilities and limitations of the technologies (caching and service workers) and draw upon a better conclusion as to which would be better suited for this project.

Another example of an extremely useful online resource was W3Schools [3]. The resource offers an extensive array of online tutorials for most, if not all, web development requirements. For example, W3Schools have a range of tutorials including JavaScript, HTML5/HTML5 Canvas, jQuery and CSS. This resource proved helpful not only in being able to better understand how web applications are created, but also to put theory into practice in the creation of both the proof-of-concept programs and the prototypes and finished product.

Other than the online web documentation and large quantity of web-based tutorials, other forms of literature were reviewed to better understand the theory and application of web developments. The most common form of literature reviewed were research papers, which used research to help give a better understanding of different web development technologies. In order to find these research papers, a number of sources were used. This included Google Scholar, IEEE Xplore and Scopus by Elsevier. When looking for specific papers, search terms such as "offline web application" and "HTML5 Development" were used and only papers from the last decade were considered, as anything older than this may use older technology which may now be obsolete for the purpose of this project. A number of research papers also "dived deeper" into some of the topics which will be used throughout this project, such as the idea and implementation of offline web applications.

1.5 Milestones Summary

Milestones are planned timeframes for different aspects of the project. These milestones ran throughout both the first and second term of the project and encapsulated the required tasks to be completed and the timeframe of which they will need to be finished by.

For term one milestones, they were set out at the beginning of the project during the project planning phase. During this time, a project plan document was created, which highlighted all of the required millstones that had been agreed. The table below outlines the milestones which were set during the project planning phase during term one.

Figure 1. Table showing the milestones set out during the project planning phase of the project, during term one.

Task set out during project planning phase	Outlined Dates
Create basic web development proof of concept program and corresponding report.	28/09/2020 - 30/09/2020
Create a proof-of-concept program for OpenStreetMap and Leaflet JS, along with the corresponding report	29/09/2020 - 02/10/2020
Setup GitHub ready for use for the project files	01/10/2020
Begin the creation of the map application	05/10/2020 - 06/10/2020
Begin to learn what is required for a webpage to be able to be viewed offline and how to make this work with OSM/Leaflet JS	07/10/2020 -14/10/2020
Create a report on offline webpages	09/10/2020 -16/10/2020
Begin to implement the required OSM/Leaflet JS data into the project files	19/10/2020 - 21/10/2020
Continue to work on the project and learning more advanced Leaflet JS to style the map, markers, etc...	22/10/2020 - 30/10/2020
Gather map Latitude & Longitude points for the different points of interest and set them up in a file for use (potentially GeoJSON) + (Create a report of GeoJSON if used)	28/10/2020 - 09/11/2020
Research and implement (into project/proof of concept program) how to create layered maps (user can change look of map for road networks, outdoor trails, etc....)	05/11/2020 - 13/11/2020
Convert file of points of interest locations into markers, boundaries, etc... into project	16/11/2020 - 18/11/2020
Create a report on Dijkstra's algorithm, explaining how it works and how it could be used for navigation	19/11/2020 - 25/11/2020
Create proof of concept programs and report for route planning	24/11/2020 - 30/11/2020
Begin to implement the route planning aspects to the main project	01/12/2020 - 07/12/2020

Create a proof-of-concept program for offline web pages, implementing different ways	01/12/2020 -08/12/2020
Work on the project to get it to work offline (basic starting of it)	02/12/2020 -11/12/2020

Whilst undertaking the project creation phases, during both term one and term two of the project, a project diary was kept keeping track of how the project was progressing and what milestones had been achieved. The benefit of keeping track of this information using the diary meant that if specific milestones take more or less time than expected, it is easier to identify and make adjustments to the projects schedule to accommodate for either the increased or decreased amount of time available for the remaining tasks. On average, the diary was updated once per week, to keep track of what had been completed during the week and act as a recap for the entirety of it. However, on some of the busier weeks where a number of tasks were achieved, multiple diary entries were committed to ensure that no details of the achieved milestones were missed.

During the first term of the project, there were a number of milestones which were not achieved to the timeframe that was outlined within the project plan documentation. The milestone was either achieved earlier than what had been planned for or required more time to be completed. As like with the overall project, these details were added to the weekly diary entries, of which a passage can be seen below. This passage was committed to the diary on the 31st of October 2020, which is after the date that was set out in the milestones section of the project plan.

> *"Continued working on the code for the actual project. Have begun working on a proof-of-concept program for the offline aspect of the map. This will continue into next week. Continued work on my project's reports, including an evaluation report of different mapping technologies and which I may choose to use in the end. October 31, 2020" [4] [Appendix B].*

The full project diary [4] can be found in Appendix B, which shows all of the entries into the diary that have been made throughout the course of the project. A rough outline of the term two milestones was also created during the project planning phase; however, this did somewhat change by the end of the first term of the development of the project. The table below outlines what was created in the project planning phase.

Figure 2. Table showing the milestones set out during the project planning phase of the project, during term one for term two milestones to be achieved.

Task set out during project planning phase	Outlined Dates
Continue creating actual projects map application – initial features (displaying the map, adding some markers and locations)	11/01/2021 - 18/01/2021
Begin implementing more locations and other ways of displaying the data on the map application	15/01/2021 - 25/01/2021
Implement some extended features (including user login/sign up)	25/01/2021 - 08/02/2021
Continue working on the map application, fixing bugs and starting to create the final report based on the proof-of-concept reports	08/02/2021 - 26/03/2021

Overall, the milestones which were set during the project planning phase were accurate to the scale of the project. However, having not created a project of this magnitude the second term milestones that were set out at the beginning of the first term could have been better. They did leave a lot of time to implement some of the more advanced and extended features of the project. However, the idea of creating the report was incorrect as this was worked on throughout the duration of the project between both terms (i.e., interim review and final project report).

Chapter 2: **Web Development**

2.1 Introduction

This chapter of the report covers all the aspects of web development, ranging from the more basic web development technologies, such as HTML, CSS and JavaScript, to the advanced features of modern web development, which includes aspects such as Caching and Service Workers for offline web applications. This chapter will also cover aspects surrounding User Interface (UI) design.

2.2 HTML, CSS, JavaScript and Other Basic Web Development Technologies

There are a number of standard programming languages which are required to get a basic website created. These languages include HTML, CSS and JavaScript. These languages can be further extended to deliver more advanced websites. However, for the purpose of this report, only the basics of these will be covered.

HTML (HyperText Markup Language) is the foundation programming language for web development. It communicates with a web browser to display the content contained within the HTML document. This is done by defining sections of the content using tags, so the web browser knows what to display. HTML5 is the latest version of the HTML programming language, published by The World Wide Web Consortium (W3C) [5]. In order to create a website/HTML document, there are a number of required tags that need to be included. The code snippet below shows how a HTML document can be laid out with the required tags:

```
<!DOCTYPE html>
<html>
  <head lang="en">
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0,
      maximum-scale=1.0, user-scalable=no" />
  </head>
  <body>
    <!-- Body of webpage goes here -->
  </body>
</html>
```

CSS (Cascading Style Sheet) is used to change the styling of the HTML elements. For example, changing the colours, background or font family. Styling with CSS can be done in multiple different ways, including inline, in-head or in an external file. It was first introduced by Håkon Wium Lie and Bert Bos in 1994. They saw a need for changing the look of a website as the web had become a platform for electronic publishing and by the end of 1995 the W3C had adopted the CSS specification as a work item with the goal of making it into a recommendation for everyone to use [6]. The code snippet below shows how a specific tag with an ID and class within a HTML document can be styled using in-head or an external CSS file. As you can see, to style a class you use the dot-notation, whereas for a tag, which has an ID, you use the hashtag notation:

```
/* Styling based on class */
.form-inline{
    width: 50%;
    display: flex;
    justify-content: center;
    padding: none;
}
```

```
/* Styling based on ID */
#submitbtn {
    background-color: white;
    border: white;
    border-radius: 5px;
    padding: none;
}
```

To style an element in-line, you would need to specify the style required when creating the tag in the HTML document. This is done by adding the "style=" option within the opening of the tag. This could look like the code snippet below, which makes the text green in colour through the styling:

```
<p style="color: green">This is a paragraph</p>
```

JavaScript is a scripting programming language which can be used in conjunction with HTML and also CSS for changing the functionality of a web page. JavaScript was first introduced on December 4 in 1995 and since then has become recognised world-wide for its ability and capabilities in the development of web applications [7]. JavaScript code can be programmed to executed based on different events which occur during use of the website. This could be when the website is loaded, based on an elements ID or on an event. For example, when the user clicks on a button. The code below displays what the HTML code would look like when executing based on a user event, on key up:

```
<input id="searchbar" type="text" onkeyup="searchAONB()"
      placeholder="Search..">
```

This would then, when the user begins typing into the text input box, perform the "`searchAONB()`" JavaScript function. The JavaScript code can be written in different ways, but the conventions follow similarities to traditional Java coding. For example, the code snippet below shows the "`searchAONB()`" code:

```
function searchAONB() {
    var userinput = document.getElementById('searchbar').value;
    var userstring = userinput.toUpperCase();
    var aonbmarkers;

    for (var i = 0; i < markers.length; i++) {
        aonbmarkers = markers[i].name.toUpperCase();
        if (userstring == aonbmarkers) {
            L.marker([markers[i].lat,
      markers[i].lng]).bindPopup(markers[i].name).addTo(aonbmap);
            aonbmap.setView(new L.LatLng(markers[i].lat, markers[i].lng),
      13, { animation: true });
        } else {
            console.log("This is not a location yet");
        }
    }
}
```

As shown, the code begins with a function being declared with the same name as what was called in the on key up action. Variables are then able to be created to perform different actions, such as searching through a JSON list of locations and checking if the string inputted matches any of the strings within this list.

In modern web development, the idea of mobile first development is now paramount. Traditionally, anyone visiting a website would be using a computer. However, a vast majority of website traffic is now generated through mobile users, such as mobile phones or tablets. Due to this, HTML, CSS and JavaScript libraries have been created which aim to achieve mobile first development. An example of this is Bootstrap. Since its initial release in August 2011, Bootstrap has become one of the most popular frontend web development frameworks [8]. In order to use Bootstrap, it will need to either be installed directly into the project files or included using jsDelivr to access the code without the need to download it. The HTML head code to include Bootrstrap through jsDelivr would look like the following code snippet [9]:

```
<link rel="stylesheet"
      href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstr
      ap.min.css" integrity="sha384-
```

```
      TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
      crossorigin="anonymous">
```

By either downloading or including Bootstrap, the different elements would then be able to be used within the HTML document with custom CSS aimed at mobile first web development. For example, creating a responsive navigation bar at the top of a webpage, which can be created using a number of navigation bar classes within a "`<nav>`" tag in the HTML, such as:

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
```

Through the use of these basic web development programming languages, a basic website is able to be created. They also form the foundation of more advanced technologies which allow for the programming of both mapping applications, but also offline applications in general. For example, when programming a map application, JavaScript is able to be extended using a number of libraries to display map data onto the website.

2.3 Advanced Web Technologies

Other than the basic web development languages for HTML5 development, there are a number of advanced web-based technologies helpful in the creation of responsive, modern web applications. For example, technologies such as jQuery and HTML5 canvas can be used to expand how the web application works. This portion of the chapter will cover some of these technologies which will be helpful in the creation of the offline mapping application.

2.3.1 jQuery

jQuery is an example of a JavaScript library. Its main purpose is to simplify HTML DOM tree traversal and manipulation [8]. The jQuery library is designed in a way to make it easier for navigation of a document, selecting DOM elements, creating animations and handling events. In general, it is used to simplify JavaScript programming. In modern browsers, jQuery will work and perform the exact same for every browser that an end user may be using.

Like with many external libraries that can be used during web development, jQuery can be installed directly into the project files and accessed locally or can be included as a Content Delivery Network (CDN) [11]. When using a CDN, you would need to include the following line of code within the head tag of the HTML document:

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.mi
      n.js"></script>
```

There is some basic syntax that is used when working with jQuery, which will look like the following code snippets from W3Schools [12]:

```
// hides the current element
$(this).hide()
// hides all <p> elements
$("p").hide()
// hides all elements with class="test"
$(".test").hide()
// hides the element with id="test"
$("#test").hide()
```

An example of a basic function of using jQuery is with JavaScript events. The following code snippets is based off the W3Schools tutorial on the jQuery action for "`click()`", which makes the element which is clicked by the user disappear from the web page [13]:

```
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
```

```
  });
});
```

Overall, jQuery is a powerful library to use in web development, as it allows for more advanced features to be programmed into a website in the simplest form possible. It also extends JavaScript with more features that would not traditionally be possible with just JavaScript in the web application.

2.3.2 HTML5 Canvas

HTML5 canvas is an element introduced with HTML5. It allows the programmer to create scriptable 2D rendered shapes and elements directly onto the web page [14]. It uses a mixture of the basic web development languages (HTML, CSS and JavaScript) to achieve the rendering of 2D shapes onto the web page. You would first need to start by creating the HTML element, which would look like the following:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px    solid
      #000000;"></canvas>
```

This would create a canvas which has a border of one pixel which is solid black and has a height of 100 and width of 200 pixels. If you wanted a smaller or bigger canvas to be displayed, the height and width variables would need to be changed accordingly. The styling can also be changed, to have a background colour or background image also. The “id” is used to find this canvas element when starting to actually do the drawing onto the canvas using JavaScript. You would then be able to add JavaScript code to the canvas element to draw different shapes. The following code snippet shows how to draw a circle onto the canvas using JavaScript, which was created following the tutorial from W3Schools [15]:

```
<script>
      var c=document.getElementById("myCanvas");
      var ctx=c.getContext("2d");
      ctx.beginPath();
      ctx.arc(95,50,40,0,2*Math.PI);
      ctx.stroke();
</script>
```

The output for this JavaScript code is shown within Appendix B of this report, to give an idea of how this will appear on a webpage. This code starts by declaring a variable (var c) which gets the canvas element that was created in the HTML code. In this case, the canvas element was defined by an “id” with the value “myCanvas”. The “getContext()” method is used to return the drawing context on the canvas, which in this case will be a two dimensional drawing. The code then begins the path that will be draw, which in this case is an arced line. This line is then drawn using the stroke function. The function of what is being drawn can be changed to draw a wide variety of objects onto the canvas.

HTML5 canvas has become a very useful technology since its creation. In terms of this project, HTML5 canvas can prove to be a powerful tool to use, as it can increase not only the usability of the mapping application being created but can also allow for the map to be created within the canvas section of the webpage, which would then give more opportunity to draw different elements onto the map through the canvas scripting, such as highlighting an area or drawing a direction/route line. However, in the end HTML5 canvas was not used in the production of the end product as there are newer technologies that work better with LeafletJS and Mapbox which is better at displaying different aspects onto the map, such as GeoJSON for displaying multiple markers onto a map.

2.4 Developing an Offline HTML5 Application

In the creation of an offline web application, there are a number of techniques to consider. The main aim of creating an offline web application is to give the user a fluent transition between using the application online and offline. There are multiple ways of achieving this, with the first technique to be considered being caching.

2.4.1 Caching

A cache is a storage area for temporary files, such as files which continuously change or get removed [16]. Web caching refers to the storing of webpages and web elements (images, text, links, cookies, etc…) for use when the user loses connection to the Internet. In other words, these webpages and elements are stored for when a user goes offline. Consequently, having a web cache in place can also improve website accessibility speeds, as the different elements which need to be loaded can be retrieved quicker. Web caching is one of the main reasons why it is quicker when loading a website, you have already visited compared to a new website being visited for the first time.

The way in which web caching works is to store the website data onto the host device, which could be the visitor's smart phone, tablet or computer. The data which is stored is collected and downloaded upon the first visit to the website. The files that are downloaded from the users visit differs depending on the type of website and how it has been created and setup. However, in general it is recommended that images that appear on the site are always stored locally, as it will mean that they will load quicker and improve the overall performance of the website for the visitor. Once the elements are stored/downloaded, every subsequent visit from the user will load the cached files.

The data that is stored within a cache, especially for a web cache, will not always stay the same or stay in the same form. The combat these changes, a web browser will periodically check that the data that it has in the cache for the visitor and website "match up" with the live website. This means that on a new visit to the website, the browser will check if anything needs to be added or removed from cache based on the live website.

2.4.2 Service Workers

Service Workers are an example of an event-driven worker, who acts as a proxy server between a web application, browser and network (when one is available) [17]. Not all browsers have support for Service Workers, such as Safari and Microsoft Edge. However, most other common web browsers have support for Service Workers such as Google Chrome, Firefox and Opera [18]. To start with, a web application needs to be served over HTTPS for Service Workers to function as intended.

The first step in creating a Service Worker is to register it. This is usually done the first time a user visits the site. The best practice is to create some, usually JavaScript, code which on every load check for a Service Worker, where if one is present and running it is not registered or created, and where one is not present register a new one. The code snippet below, taken from a tutorial from Google Developers [18], shows how to register a Service Worker.

```
(function(){
    if ('serviceWorker' in navigator) {
        console.log('CLIENT: service worker registration in progress.');
        navigator.serviceWorker.register('service-
      worker.js').then(function() {
        console.log('CLIENT: service worker registration complete.');
        }, function() {
        console.log('CLIENT: service worker registration failure.');
        });
    } else {
```

```
        console.log('CLIENT: service worker is not supported.');
    }

})();
```

You would then need to install the service worker once it has been registered. This is performed using an event listener for 'install'. The files which need to be cached are first defined, which can either be all of the web files which is called with empty apostrophise ('') or specific files by adding the required file between the apostrophes. The code snippet below, again from the Google Developers [18] outlines how this may be performed:

```
var CACHE_NAME = 'my-site-cache-v1';
var urlsToCache = [
  '/',
  '/styles/main.css',
  '/script/main.js'
];
self.addEventListener('install', function(event) {
  // Perform install steps
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      })
  );
});
```

At the barebones of a service worker, this will be enough to (at load) cache all of the required files and resources that the web application will work whilst offline. However, the application can be extended to fetch specific data when the web application changes for the user. For example, when a visitor changes to a different webpage or zooms out of the map. In this case, this change in the "event" that is occurring will trigger the installed service worker to "fetch" more data to cache. This process is done through the use of creating a cache copy which will then automatically gather any website changes and store them within the cache based on the service worker.

A service worker can be found within the "developers' tools" section of the browser. Here, a user is able to see if they have one running, the status of the service worker, the clients that are using the service worker and can check for the network requests that have been performed. Also, dependant on how the service worker has been created and setup, within the console of the same window the different mechanisms of the service worker can be seen. For example, all of the data that is being stored in the cache after being "fetched". All requests from the user to the site are performed via a fetch request, which locates all of the required recourses so that should the user lose a connection to the Internet, the website will still function as intended. The screenshot below shows an example of the developer window on Google Chrome, which shows the service worker running and some of the different console functions that have been performed.

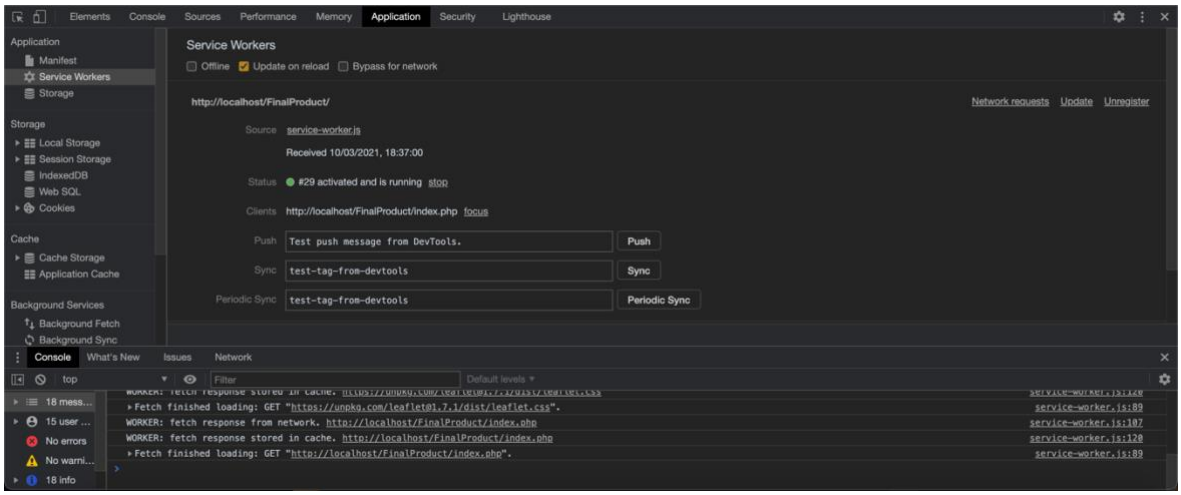

Figure 3. Screenshot showing the developer window on Google Chrome with the active and running service worker, along with some of the fetch requests that it has performed for the user when they have visited the site.

Traditionally, offline web applications simply relied on caching through the use of a manifest to be able to deliver offline web capabilities. However, with the introduction of service workers there is now a greater performance of offline web applications as the service worker is always running and checking for any updates that may have occurred anytime that there is a connection to the Internet. This means that for a web application such as a mapping website, whenever the user regains a connection to the Internet, the map will be able to be updated with more available should the user lose connection once again.

For the purposes of this project, if a service worker is unavailable (i.e., the browser being used is significantly older, then the program will not function offline as the best method which was available on the older web browsers (application caching) is now a deprecated asset, meaning that it can create errors in the service worker if attempting to use the manifest for an application cache.

2.5 Single-Page Web Applications

In modern web development, a popular application type is a single-page web application. This is one where all of the information is stored on the one webpage, accessible through the use of scrolling down to the different sections. This allows for the addition of smooth scrolling, which animates the user to the different sections and can still feel like a multi-page web application while still only being the single page. The way that these single-page web applications work is by "dynamically rewriting the current page with new data from the web server" [57]. Essentially what this means is that the page the user visits changes, rather than the user changing what page they are visiting. Examples of frameworks ideal for creating single-page web applications include Angular, Vue.JS and React. All three of which are JavaScript based frameworks. All of which have pros and cons to using them. For example, Angular has "two-way data binding which instantly replicates the changes that are made to the model instantly to the view" but can (for bigger projects) impact the performance negatively [58]. React uses virtual DOM which provides very good efficiency for the web application, but this can result in the need for more libraries which can take up more time and space [58]. Vue.JS is a very easy framework to use and implement, with great documentation. However, with it still being relatively new there is not a huge community of people that can help with an error you may encounter [58].

The reason why this project did not follow the single-page web application design and creation is because it felt like it would become too "clunky" to have all of this information being displayed on the one page. Although it could have been achieved with tabs, pills and dropdown

menus, it is with personal belief that this would not have been an appealing website to use for the end user because of the sheer number of AONB locations and the information being presented. Also, the reason why a framework such as Vue.JS, Angular or React were not used is because of personal experience creating full stack web applications which are multi-page using PHP and MySQL. However, it would be good in the future to extend into learning and developing with a single-page web application.

2.6 User Interface Design

When creating a web application, a User Interface (UI) design is important because it gives a reference of how the finished product will look, which can be used when building the prototypes and finished product. There are a number of ways in which a UI can be designed and at different stages of a project this design will change a large amount. For example, the traditional method of prototyping through UI design would see a low-fidelity prototype being created. This is traditionally a physical representation of what the finished product could look like, usually made on paper rather than electronically. This would then be able to be presented to any stakeholders and potential user groups for UI testing, which will check that all of the required heuristics for functionality are met. The photographs below show an example of a low-fidelity prototype which has been created by drawing the design onto "paper".

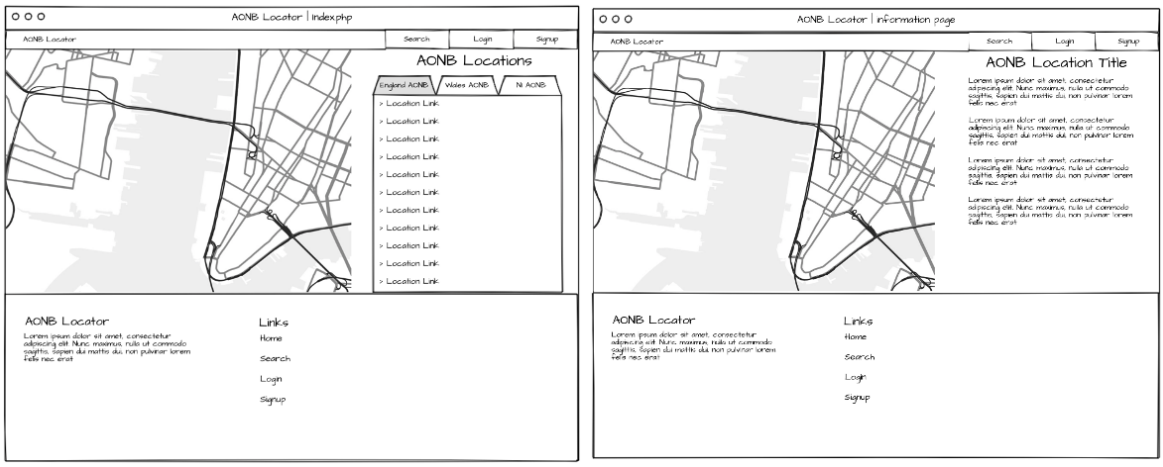

Figure 4. Photographs of an example of a low-fidelity prototype which was created when making the demo application.

Once a low-fidelity prototype has been created, a project team could then begin to work on a more advanced prototype (usually referred to as a high-fidelity prototype) which could be created using software to represent the design and functionality of the finished product. There are a number of resources available in modern designing, however some of the more reliable and feature driven are Figma [19] and MockFlow [20]. These have the tools available to create a fluid UI design which best represents what the finished product will look like. There are also tools available, especially in Figma, that allow for the design of interaction. This means that the prototype can "work" almost as if it is the finished product. Test users could click through different interactions on the site and see exactly where they would land. Prototyping and testing in this way allows for a better understanding of how best to approach building the final product and identify any issues in design very early on. Below is an example of home page, designed using MockFlow [20].

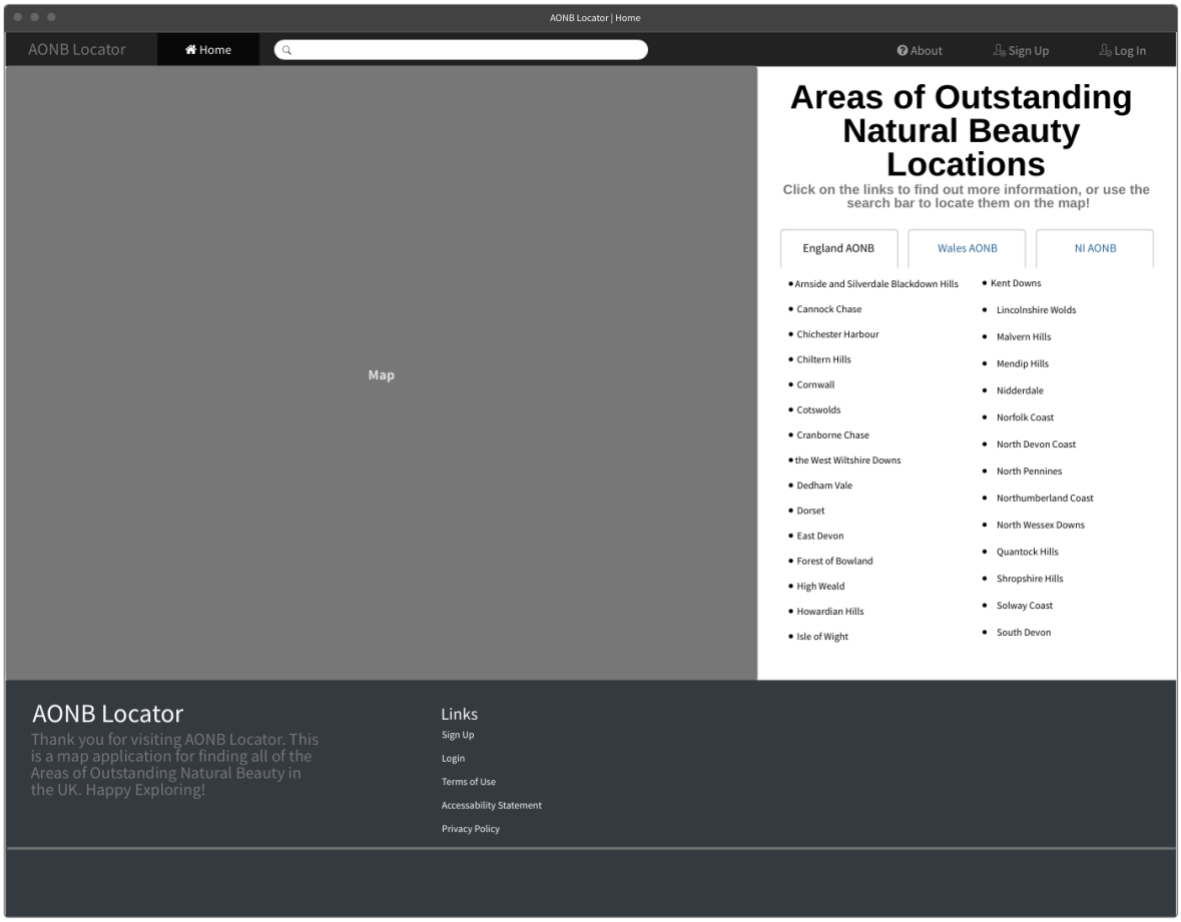

Figure 5. Screenshot showing the design of a potential homepage for the project, created using MockFlow [20]. This is what the term one demo application was based off.

Creating UI designs are a vital part of any software development project as it gives a much better idea of the functionality, look and feel of the project before any significant time has been spent attempting to create the program. This in turn will reduce the amount of time spent redesigning the UI of the finished product, which could also have an effect on the functionality. Whereas, when creating the designs to start with any little bugs in the functionality can be highlighted early on and changed accordingly.

Chapter 3: **Frontend Web Development**

In web development, you have both the frontend (which is what the end user will see on the website) and the backend (which is what makes the website function, with features such as connecting to another service). Within this chapter, the frontend web development will be covered.

In terms of frontend programming languages, there are the main three languages in use. These being, HTML, CSS and JavaScript (which is both a frontend and a backend programming language depending on use). With frontend development referring to what the end user will physically see, it is easy to assume that what is added to through these different languages will affect what is seen on the website. These technologies have been covered within chapter one of this report; however, it is good to also note that when creating a full stack web application, frontend languages can also be merged with the backend programming languages to make for a better transition. This can include doing things such as coding a HTML file using a '.php' extension so that the backend can interact with the frontend HTML better. The same code will still be displayed to the user, i.e., the HTML code will still be displayed on the webpage but will also be able to interact and make adjustments based on the backend of the application. Below is an example of some HTML code which is incorporating the PHP programming language to change how data is displayed.

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav mr-auto">
    <li class="nav-item active">
      <a   class="nav-link"   href="index.php">Home   <span   class="sr-
      only">(current)</span></a>
    </li>
    <?php
      if(isset($_SESSION["useruid"])) {
        echo       "<li       class='nav-item'><a       class='nav-link'
      href='profile.php'>Profile</a></li>";
        echo       "<li       class='nav-item'><a       class='nav-link'
      href='logout.php'>Logout</a></li>";
      }else {
        echo       "<li       class='nav-item'><a       class='nav-link'
      href='login.php'>Login</a></li>";
        echo       "<li       class='nav-item'><a       class='nav-link'
      href='signup.php'>Sign Up</a></li>";
      }
    ?>
  </ul>
</div>
```

As you can see, the mixture of both HTML tags and PHP code can be achieved fluently. This is achieved through the use of the opening PHP tags (<?php), which then allows for PHP code to be added into the HTML code. By doing so, this could change what is being displayed in the HTML tags, which in turn would change what is being displayed to the user.

In the process of building the proof-of-concept programs, prototype and final product, forms were needed for multiple reasons. For example, on the user registration and login web pages there are two different forms that the user fills out to either sign up or login. A form is just like a real-world form object, it is a group of boxes which the user of the form needs to populate with information. There are a number of different form features, such as the usual text boxes which can house names, addresses, emails or phone numbers. There are also password text boxes, which hide the information the user is inputting and there is checkboxes and sliders, which represent the inputted data in a more graphical form. Combining a number of these form features together, a developer is able to create a place for a user to fill out details for a number of different tasks.

When creating a form, whether using standard HTML and CSS or a CSS framework such as Bootstrap, there are predefined tags for it. The example below outlines a basic form, with all of the different form features in standard HTML and CSS. This includes the input type of a checkbox, input field and dropdown menu based off a tutorial from W3Schools [21].

```
<form action="/action_page.php">
<label for="fname">First Name</label>
      <input type="text" id="fname" name="firstname" placeholder="Your
      name..">
      <label for="lname">Last Name</label>
      <input type="text" id="lname" name="lastname" placeholder="Your
      last name..">
      <label for="country">Country</label>
      <select id="country" name="country">
        <option value="australia">Australia</option>
        <option value="canada">Canada</option>
        <option value="usa">USA</option>
      </select>
      <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
      <label for="vehicle1"> I have a bike</label><br>
      <label for="subject">Subject</label>
      <textarea id="subject" name="subject" placeholder="Write
      something.." style="height:200px"></textarea>
    <input type="submit" value="Submit">
  </form>
```

Within the opening tag of the form, the action command is what gives the form something to do. This means that, for the example above, that the file ‘action_page.php’ will be run when the form is submitted via the submit button at the end of the form. Within this PHP file, it will perform the backend of submitting the form such as adding all of the details inputted to a database or checking if it is possible to log a user in. In this case, it could store all of the information being inputted into a database. Below is an example of what a form would look like when displayed to the user. This example is a login form which can be completed by the user to login to a website.

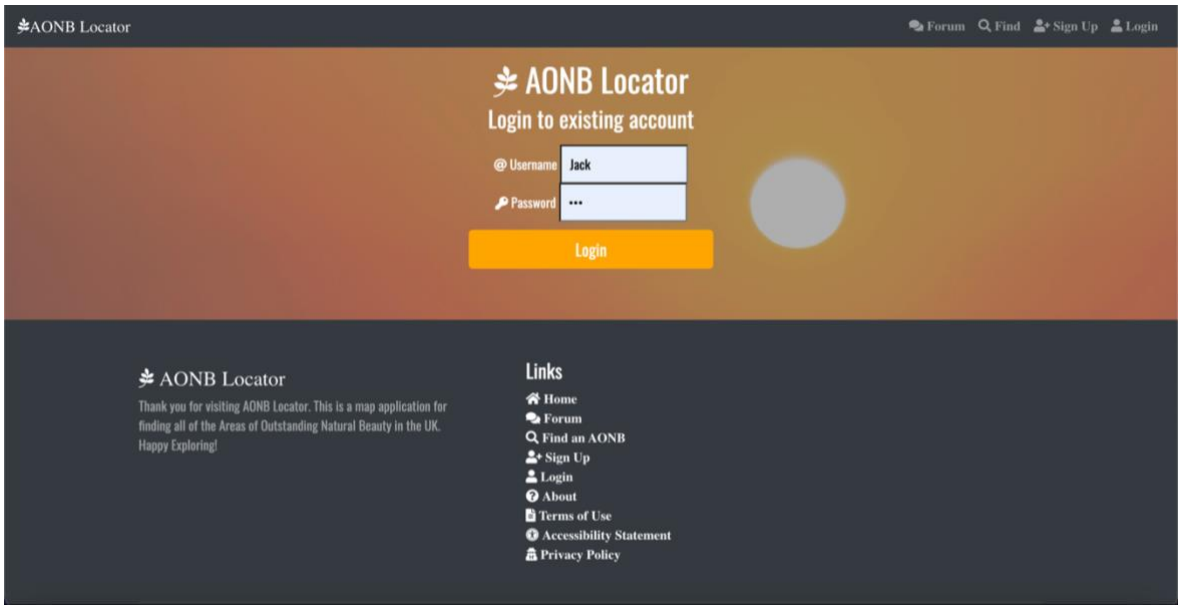

Figure 6. Screenshot showing a login form which contains a username and password field. This would be used to log a user into the website using their account.

Chapter 4: **Backend Web Development**

There are a number of backend features which can be implemented to improve a web applications usability and give more functionality to the overall website. This can also be completed in a number of different programming languages, such as JavaScript, node.js, Ruby-on-Rails and PHP. For example, these languages can be used to make animations or connect to other services, such as connecting to a MySQL database.

There is a vast amount of potential programming languages which can be used for backend development. Depending on what is to be achieved from the backend, there could be one or more programming languages that could be used to achieve the required functionality. Some examples of backend programming languages include JavaScript, PHP, node.js, Python and Ruby-on-Rails [22]. JavaScript, however, is an example of both frontend and backend as it can be used in all aspects of a web development stack [23]. Each of the different programming languages can achieve a number of different functions for a web application, for example JavaScript is an example of a good programming languages to add animations to a webpage, as it is based on the Java programming language and has a vast library of tools which can be used to achieve this. However, PHP is an example of a better programming language to connect to other backend services, such as databases.

When developing a full stack website, there is a need to connect (together) the backend and the frontend which has been created. There are a number of different ways that this can be achieved and can depend on the backend programming language being used and how the web application has been setup. The frontend and the backend interact in two major ways, which is how they would be connected together.

The first is a HTTP request. When searching for a specific page within the webpage, the users web browser is making a HTTP request to the backend development environment. It is the backend programming which returns the HTTP response. This response will contain the frontend code (HTML, CSS, etc…).

The second form is for single page web applications. Again, the only interaction is through HTTP. The frontend will send HTTP requests to the backend to create, read, update and delete data. The backend will then return a HTTP response which will contain all of the necessary data for the user interface to make changes to its state, such as displaying user data. [24]

For the creation of this project, PHP has been used and so this section will cover how it is used in the creation of the backend. In its simplest form, to connect the PHP backend code to the frontend would be to use the '.php' extension for all of the files (including the frontend HTML files). This will then allow for PHP tags to be used within the HTML base code, which can then be used to run other PHP files which contain specific functions. For example, a form on the login HTML page would take the data and store it within the PHP tag, which is then used in the user login PHP file which connects, checks and returns the user login confirmation.

4.1 Database Creation

For certain aspects of a web application, a database would be required. This could be to store user information (such as login/registration details), blog posts, photos and videos for example. Web databases are very versatile in that they are able to store a wide array of data and information which can be used by a web application to both change the state of a webpage, store users' information and display specific information for the user.

When creating a database for use by a web application, you should first decide on which type of database you would like to use. There is a large amount of different database managers, all

with different creation techniques and are used in a different way, both by the web application and the creator. For example, you may choose to use MySQL, MongoDB, PostgreSQL or Oracle Databases. For the purposes of this report/project, MySQL will be used and discussed. Also, the example of setup below is performed on the 'Localhost', however there may be some differences dependant on which host/server provider is being used.

MySQL is an example of a Relational Database Management System (RDMS). A relational database is a type of database which stores the data contained in a structured format, i.e., it uses rows and columns. By using the rows and columns, it makes it easier to locate and access specific values stored in the database [25].

A developer should first have MySQL installed. Once this is done, you can then get into the MySQL editing screen through the terminal/command line. On Mac, the following statement will allow you access to edit MySQL.

```
/usr/local/mysql/bin/mysql -u root -p;
```

You should then be presented with a screen which looks like the following, dependant on what machine/operating system is being used.

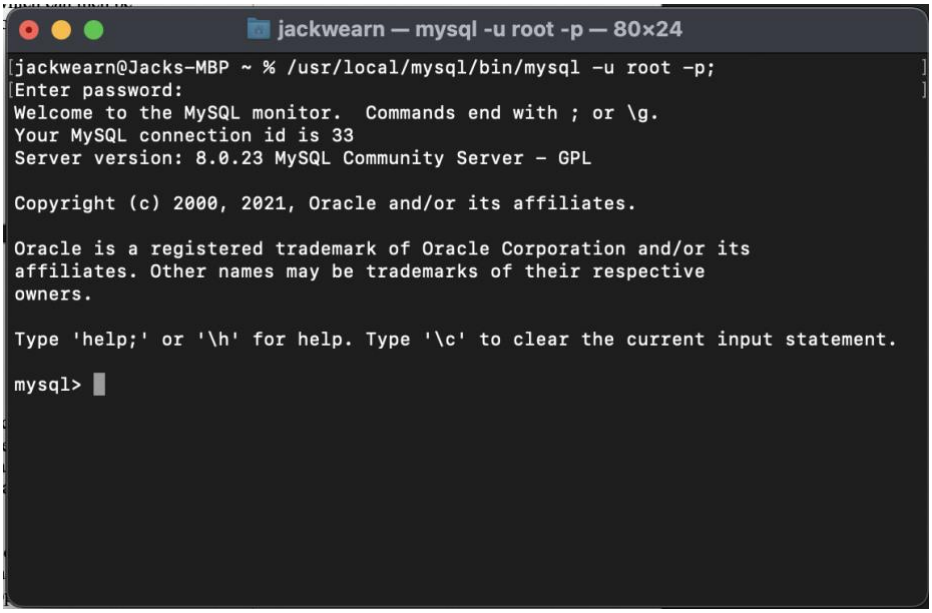

Figure 7. Screenshot showing the initial screen when loading into MySQL in the Terminal.

There is then a number of functions which can be performed. The first of which would be to display all the databases which have already been created. Depending on when running this statement, it may only show the default databases, however if some databases have been created then it will also display these. The command which would be run to do this is:

```
mysql> show databases;
```

The screenshot below shows an example of the result of this command.

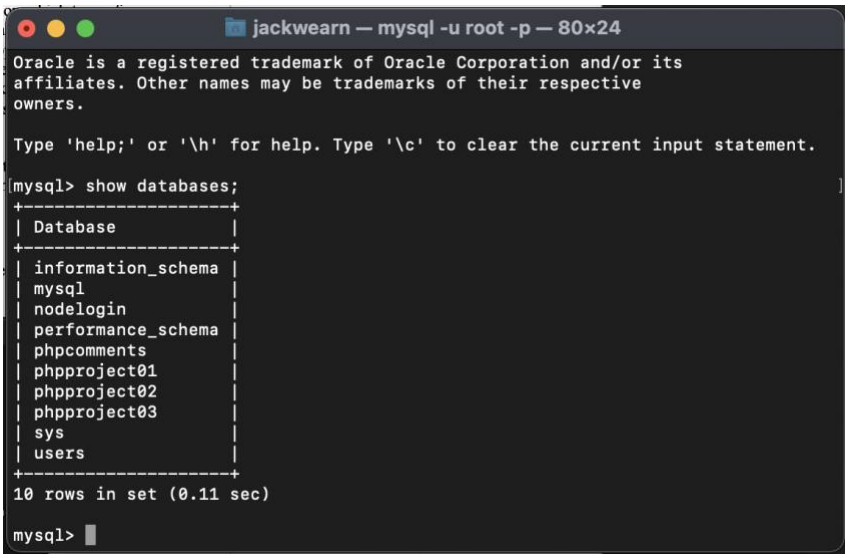

Figure 8. Screenshot showing the result of running the show databases command.

From here, you have a few options. If you would wish to just use one of the already created databases, you can run the following command. This will select the database that you wish to use, edit or add to.

```
mysql> use EXAMPLEDATABASE;
```

A new database can also be created. This is again done with a statement within the terminal window. SQL code is used to create databases, tables and to query these databases/tables. The statement below is an example of how the 'phpcomments' database was created for use in a proof-of-concept progam which will be discussed in a later chapter.

```
mysql> CREATE DATABASE IF NOT EXISTS `phpcomments` DEFAULT CHARACTER SET
utf8 COLLATE utf8_general_ci;
```

Now that the database has been created and setup, you are then able to create the table within this database which will be used to store, for this example, a user's comments which have been submitted on a webpage. This again is done through SQL commands within the terminal, show below.

```
mysql> CREATE TABLE IF NOT EXISTS `comments` (
           `id` int(11) NOT NULL AUTO_INCREMENT,
           `page_id` int(11) NOT NULL,
           `parent_id` int(11) NOT NULL DEFAULT '-1',
           `name` varchar(255) NOT NULL,
           `content` text NOT NULL,
           `submit_date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
            PRIMARY KEY (`id`)
         ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
```

This will then create the table, called 'comments', if it does not already exist within the 'phpcomments' database. If it does exist, it will not be able to execute the creation of the table. However, if it does not exist, the table will be created. This can now be used as a database on a web application. You would first need to connect the frontend and backend to this database and this table.

In order to test that the creation of the table was successful, you can also add test data to the database. This is done by adding data to the table, such as the following:

```
mysql> INSERT INTO `comments` (`id`, `page_id`, `parent_id`, `name`,
      `content`, `submit_date`) VALUES (
```

```
3, 1, -1, 'Jack', Hello', '2020-07-22 14:37:43');
```

From here, you can use the SELECT SQL command to show this data in the database, such as the following.

```
mysql> SELECT * FROM comments;
```

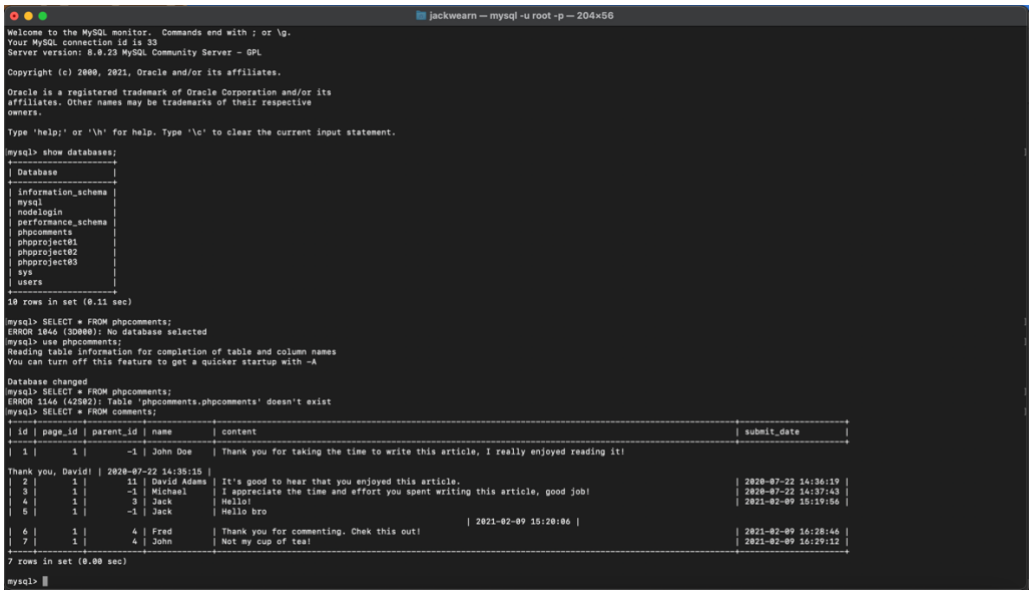

Figure 9. Screenshot showing the output of the SELECT function on the 'comments' table in the 'phpcomments' database.

Another way in which a MySQL database for use with PHP can be created is by using PHPMyAdmin. This is software which allows a developer to create, edit and delete databases and tables directly from their web browser of choice. It is accessed through, in this case, the localhost. The web address for it in this case is:

```
localhost/phpcomments/phpMyAdmin
```

This will display a screen, which will allow you to access all of the created databases through MySQL and within them all tables which have been created within them. The screenshot below shows the home page of PHPMyAdmin. As you can see, on the left hand side is all of the developer created databases, which can be expanded to display all available tables within them too.

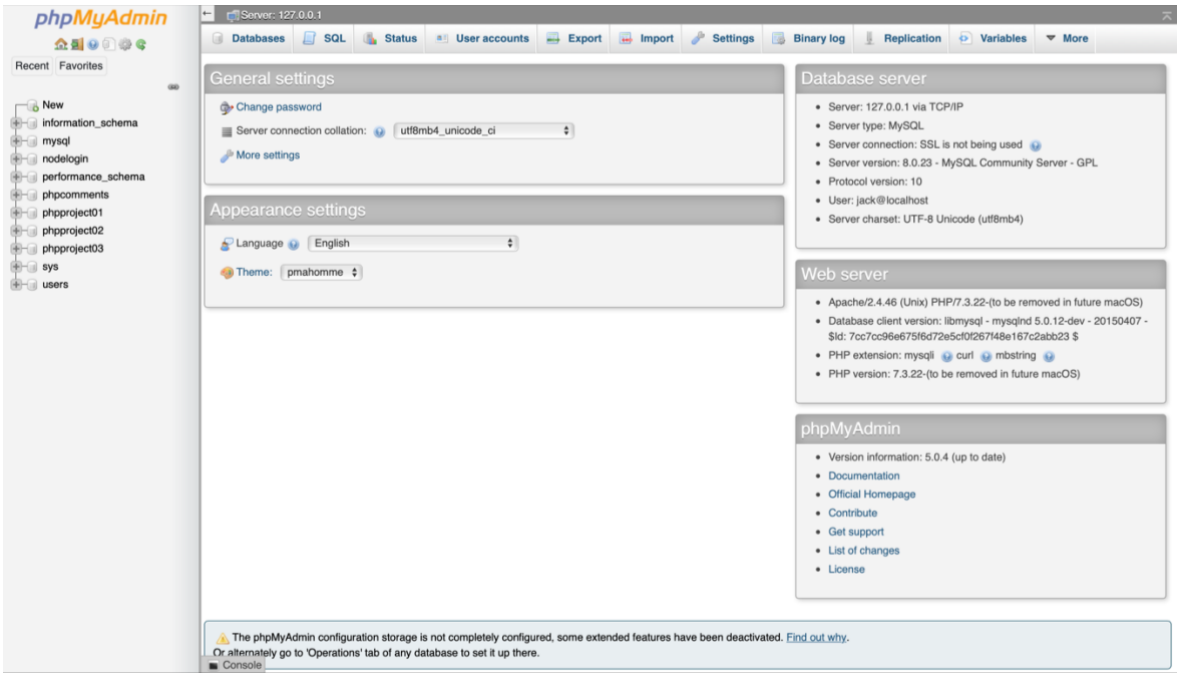

Figure 10. Screenshot showing the PHPMyAdmin home page, which is displayed after logging in.

If you then select one of the databases, it will then load in all the corresponding data for that database. In this case, the phpcomments database has been selected. On the left hand side you can now see all of the tables and the data which is used to generate them. On the main portion of the screen, it will show the different tables. This is shown in the screenshot below.

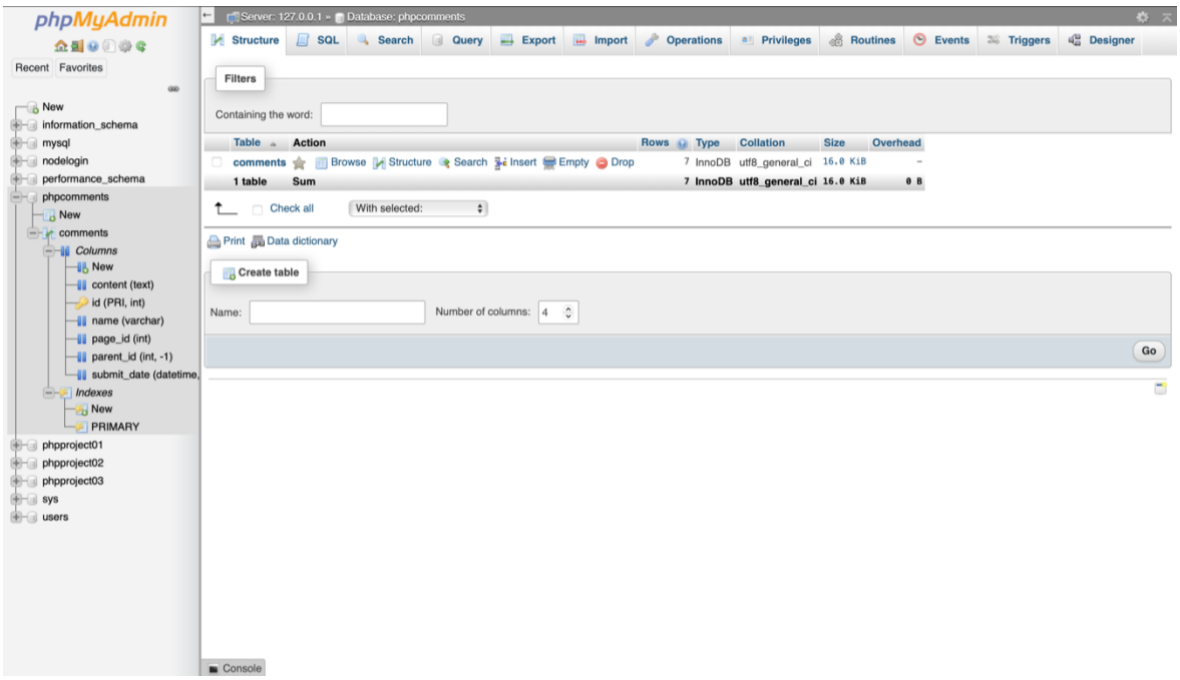

Figure 11. Screenshot showing an open database (phpcomments database) and its corresponding table (comments).

From here, there are a few options available, you can create a new table either through the UI 'create table' option, where you will need to give it a name and the number of columns it will have. Or, if you have some experience with SQL code, you can select the SQL option at the top to perform SQL queries, which could be to create a new table or to query an existing table. You are

also able to select one of the tables which has already been created. This will then show all of the data stored in the table. This is essentially the same as using the command line option of "use database" and the SQL query of "SELECT * FROM comments". The screenshot below shows the comments table open within PHPMyAdmin.

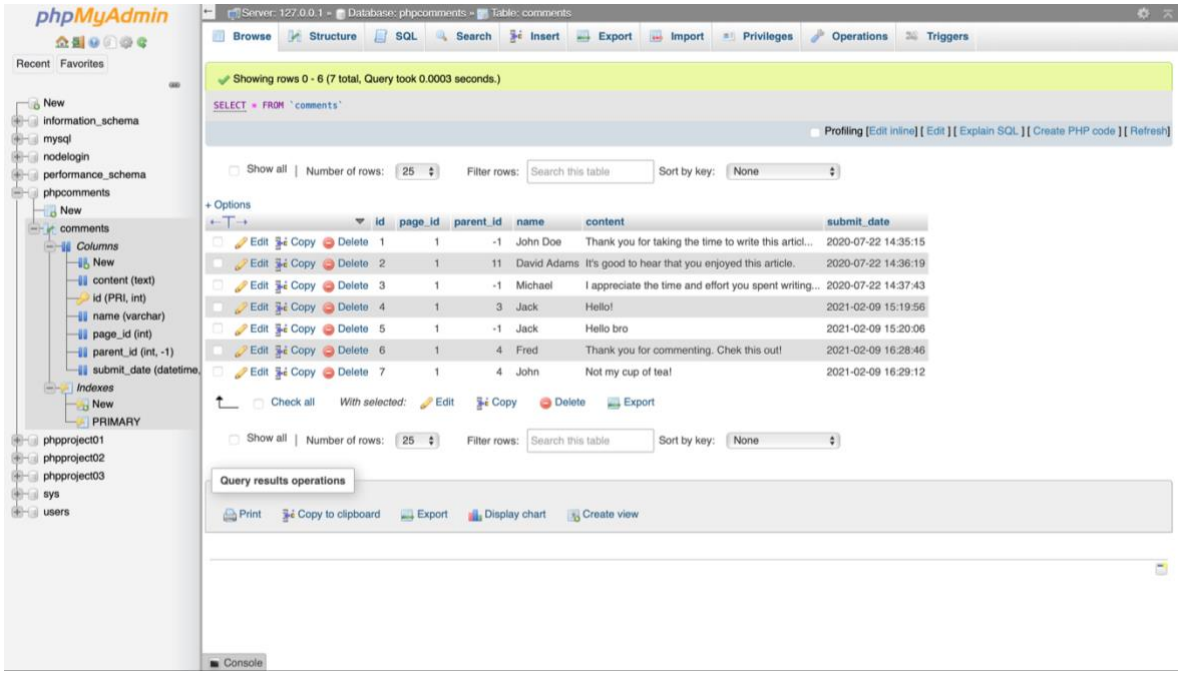

Figure 12. Screenshot showing the comments table open within the phpcomments database using PHPMyAdmin.

The next screenshot shows this same table being used within the same database, however it shows how it would look when selecting the SQL option at the top of the webpage. By pressing go at the right hand side, it will display a similar screen to figure 6, however this is dependent on what the SQL query is.

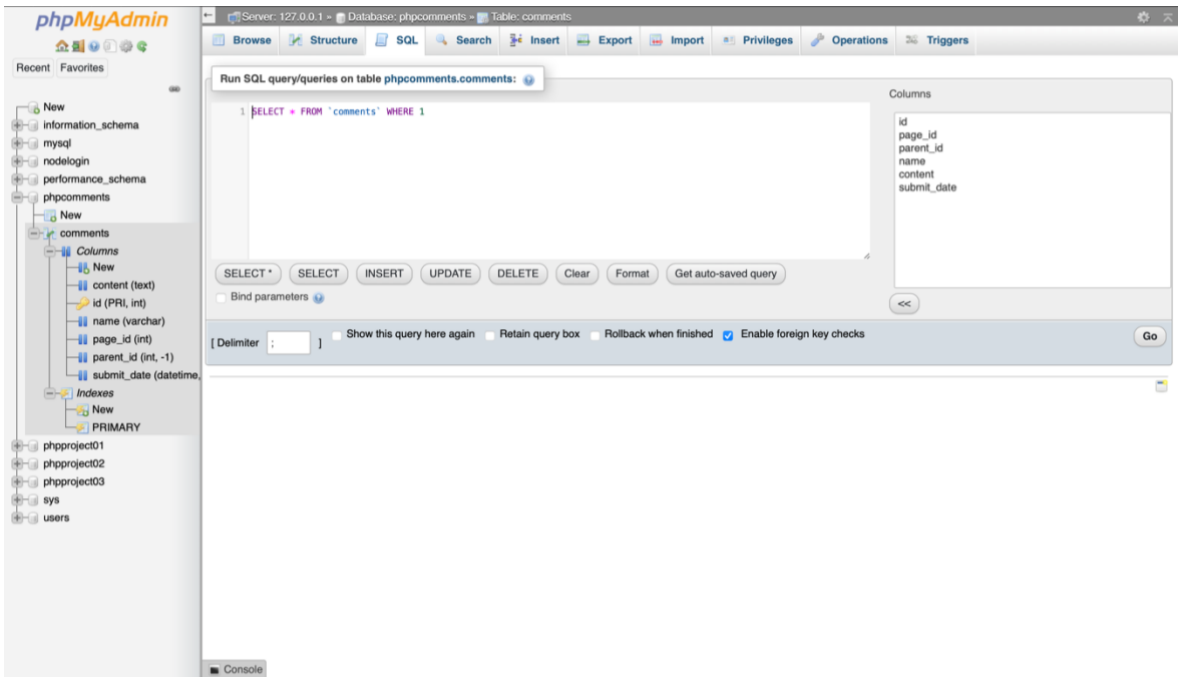

Figure 13. Screenshot showing phpMyAdmin when selecting the SQL option.

4.2 Connecting to the Database

Once a database has been created, the next step would be to connect the web application to that database so that specific functions can be carried out. Depending on what programming language is being used for the backend of the web application, this process may look slightly different in terms of the layout and functions used within the coding language. However, in principle, the idea of connecting to the database is the same for all programming languages.

You should first create a set of variables which will contain the details of the database. This will include the location of the database, the name of the database, the admin user (a user who has full privileges to make changes to the database, such as root) and the database password, should there be one. In the context of this project, PHP will be used and so this portion of the document will cover how the connection to a database is setup in the PHP programming language.

The code below is an example of how the different variables to be able to connect to a database can be set out. Here, four variables have been defined which will house the credentials for the database host (which in this case will be the localhost, or 127.0.0.1), the database name (which was named in the previous section as 'phpcomments'), the database user and the password for that admin user.

```
define("DATABASE_HOST ", "127.0.0.1");
define("DATABASE_NAME ", "phpcomments");
define("DATABASE_USER ", "jack");
define("DATABASE_PASS ", "Password");
```

Below is another example of how these variables could be defined.

```
$DATABASE_HOST = '127.0.0.1';
$DATABASE_USER = 'jack';
$DATABASE_PASS = 'Password';
$DATABASE_NAME = 'phpcomments';
```

Once these variables are in place, they can then be used to establish a connection. The variables are usually contained within a separate file, to make editing more easy should any credentials need to be changed. In the connection file, the following code could be used to establish the connection.

```
try {
    $pdo = new PDO('mysql:host=' . $DATABASE_HOST . ';dbname=' .
$DATABASE_NAME . ';charset=utf8', $DATABASE_USER, $DATABASE_PASS);
} catch (PDOException $exception) {
    exit('Failed to connect to database!');
}
```

Here, a 'try-catch' block has been implemented in order to catch a 'failed to connect to database' exception. The way this code block works is by creating a new PHP Data Object, which will be used to create the connection to the database. It then takes the different database specific variables and assigns them to the corresponding MySQL requirement. There are a number of ways to create a connection however. Below is another example of a simpler connection using the 'mysqli_connect' function.

```
$conn = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS,
$DATABASE_NAME);
if(!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

Here a variable '$conn' has been created which is the function 'mysqli_connect'. This takes the database host, user, password and name to check for a connection to the database. If a connection cannot be made, an error message will be displayed within the console and the function

will be terminated. However, if a connection can be made, more functionality revolving around the database can be done, such as selecting all of the data from a table or inserting into a table in the database.

4.3 Changing the User Interface Based on Backend Functions

When developing an application, the frontend which is being displayed to the end user can be changed dependent on any changes that have occurred through the use of backend development. For example, a navigation bar can be altered if a user is logged into an account or not. There are also other changes that can be made dependant on options that have been set. An example of a web application that does this would a social media website. Dependant on who a user follows or is connected with will change what is being displayed to them on a feed for example.

For the use of this project, the main changes that would be made based on the backend functions would be the navigation bar. This would change what navigation links are shown, for example showing a registration and login link when the user is not logged in and showing the users profile link and logout link when the user is logged in. These changes can be done from the frontend (HTML) source code with the use of backend features. The backend aspect will be discussed in more detail in the following section, however in the case of PHP code being used to change what is being displayed the change would just consist of including this PHP code within the HTML tags. Using the likes of if-then-else statements to display specific information based on the backend details. Below is a screenshot showing the change that a navigation bar may make when being changed based on the backend, i.e., showing a username rather than the login button.

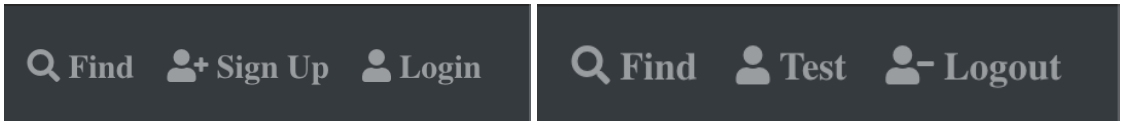

Figure 14. Screenshots showing the difference between the navigation links when a user logs in or is not logged in.

Chapter 5: **Open Street Map (OSM) Data**

5.1 Introduction

Open Street Map (OSM) is open-source map data, used in the production of web applications, mobile applications and hardware devices [26]. There are a number of ways in which this data can be presented on a web application, along with different map tile options to be displayed. Within this chapter of the report, the OSM data and how it can be displayed and used will be covered. It will also cover different ways of displaying OSM data to the end user.

5.2 Raster and Vector Tile Maps

When creating a map within a web application, there are two main options of displaying this data. These two options are vector-based tile maps and raster/image-based tile maps. Both of these options have their own advantages and disadvantages for being used. Raster tiles are based off images [27]. A map built using raster image tiles are made up of a large number of images (usually of the form ".jpg" or ".png"), which are organised to be placed next to each other. The main disadvantage to using raster image tiles is performance and speed. Rendering all of the different images in real time can result in a loss of performance for the user when interacting with the map. However, it does allow for some more customisations that other methods cannot achieve, such as displaying custom images within the map itself.

On the other hand, vector tile maps are used to deliver small chunks of geographical data to the web browser when the user is accessing the mapping application [28]. A vector tile map is similar to that of the raster image tile maps, however instead of delivering raster images, vector representations of the tile features is returned to the browser for the user to interact with. The most common way of this being delivered is through the use of GeoJSON, which can store data for representing different elements of a maps tile. For example, it can include data for bodies of water which will be made up of polygons and roads, which are made up of Line Strings. The main advantage of this approach to generating a map for web development is the speed/performance gains. The tiles usually have a very small tile size [29], which means that the map will load generally quicker and also the small size makes them ideal for offline application use. However, a major disadvantage to using this approach is that the rendering occurs on the client side, which means that not all devices will get the increase in performance as slower devices will see a performance loss/lower loading speeds [29].

Both raster and vector tile mapping methods are a good choice in their own right of displaying map data. However, both have a variety of advantages and disadvantages, which could make it more difficult to come to a conclusion of which would be the best approach to take when building a web application which incorporates a mapping element.

The major pros for vector tile maps include the following: smaller data size, lower bandwidth consumption, faster generation time, more fluid user experience, zoom level changes are not visible to the user, standard for mobile first development and easier customisation [30]. The reason why these points are major pros of vector maps is because of the technology used to generate these tiles. For example, by having smaller data size, it also means that there is a lower disk space requirement. A vector tile map is ideal to appeal to a wider customer base, as more users will be able to easily access and use the map application. However, vector tile maps do also have some disadvantages, such as: map rendering being done on the client side which will require more powerful hardware and the data is generalised and not suitable for direct edits.

The advantages of raster tile maps include the following: more suitable for data such as satellite imaging and ariel images, lower hardware requirements for the user and has better support in web-based software such as JavaScript [31]. However, there are disadvantages to raster tile maps, which include the following: larger size for each of the different tiles and data stored on the server, longer generation times and slower loading which can lead to a disruption to the overall user experience of using the web application, such as moving around the map.

Taking all of the pros and cons into account for both raster and vector tile maps, it is easy to see why there is such debate as to which is the better of the option in building a mapping web application. However, one of the better options which is available is to combine the two depending on use. Both have their speed and usability advantages which are dependent on how they are being used, which means a developer could take advantage of each of their pros and aim to avoid the cons of each as much as possible. For the use of this project, vector tile-based maps have predominately been used.

5.3 Displaying Open Street Map Data

In terms of displaying the OSM data, there are a number of approaches that can be taken. For example, Leaflet JS which is a JavaScript library used for the creation of maps for web/mobile development or Mapbox.

Leaflet JS is a powerful open-source JavaScript library [32], which allows the programmer to display the OSM data as a tiled map layer on a web application. Through the use of different plugins, JavaScript functions and built-in functionality, a full featured, interactive and responsive map application can be created. Because the library is built around JavaScript, it allows for a wide variety of custom features and implementations for the map application.

Another way in which OSM data can be displayed is through the use of Mapbox which is a Software Development Kit (SDK) created to make the process of making mapping applications easier [33]. Mapbox offer products which include map creation, but also navigation application creation. Like with Leaflet, Mapbox uses JavaScript for processing the code and displaying the map data onto the HTML document. Both Mapbox and Leaflet have their own advantages and disadvantages.

However, there are also other methods for displaying OSM data. For example, Google Maps is also able to display OSM data, as long as it is able to meet the OSM License [34]. The difference with using Google Maps to display the OSM data, is that in order to use the Google Maps API for displaying map data, a license is required which usually costs money to acquire an API key. However, if the developer wanted the map application to keep a familiar look and feel of a map that is predominantly used, Google Maps would be a good option as a large proportion of people use Google Maps as their usual map application (on mobile phones for example). An example of a website which has implemented a Google Maps, whilst also displaying OSM data, is found at Capital [35], which is a German property value finder.

For ease of creation, Mapbox may be the better choice to use when creating a mapping application. The reason for this is because it has many features pre-built into the initial call to use the Mapbox SDK. However, this does cause some limitations in implementing custom features and customising the look of the map/map elements. Which is why, for the purpose of this project, Leaflet would be the better technology to use for creating the main map. However, by using both within the application, the look and feel of the overall website could increase dramatically. For example, implementing Mapboxes static maps API for information pages on different AONB's. Also, due to the nature of the project and the need for an offline map application, the most suitable map tile type for the main map would be to use vector map tiles. However, like with working in unison with Leaflet and Mapbox, raster image map tiles could be used for the static maps on the information web pages. For the purposes of this project, the Google Maps API method of creating

and displaying a map was rules out as the cost of acquiring an API key from Google [36] seemed unnecessary. Although it would not breach the threshold to begin paying just yet, if the project was to be released it may not be the most cost-effective method.

Chapter 6: **Proof-of-Concept Development**

6.1 Introduction

Within this chapter of the report, all the proof-of-concept programs which were created in both the first and second term will be covered. It will show examples of the running programs and any errors that may have been encountered. It will also highlight what was learnt from building the different proof-of-concept programs

6.2 A "Hello World" Offline HTML5 Application

Over the course of the first term, a number of proof-of-concept programs were developed to better understand the different programming languages, tools and techniques. The first of which was a "hello world" offline HTML5 application.

This first proof-of-concept program was built using the two earlier discussed techniques, application caching and service workers. By building an application for each of these, it allows for an easy comparison of the two technologies. The application caching technique was easiest to implement into the proof-of-concept program, only requiring the manifest declaration/creation and a small JavaScript function which was an event listener on the loading of the webpage. However, as discussed earlier when attempting to make the webpage work correctly offline the error message stating that application caching should no longer be used appeared in the console of the web browser. The webpage was able to be viewed while offline and not connected to the Internet.

The screenshot below shows the error message within the console on the web browser, which points towards using the service worker method:

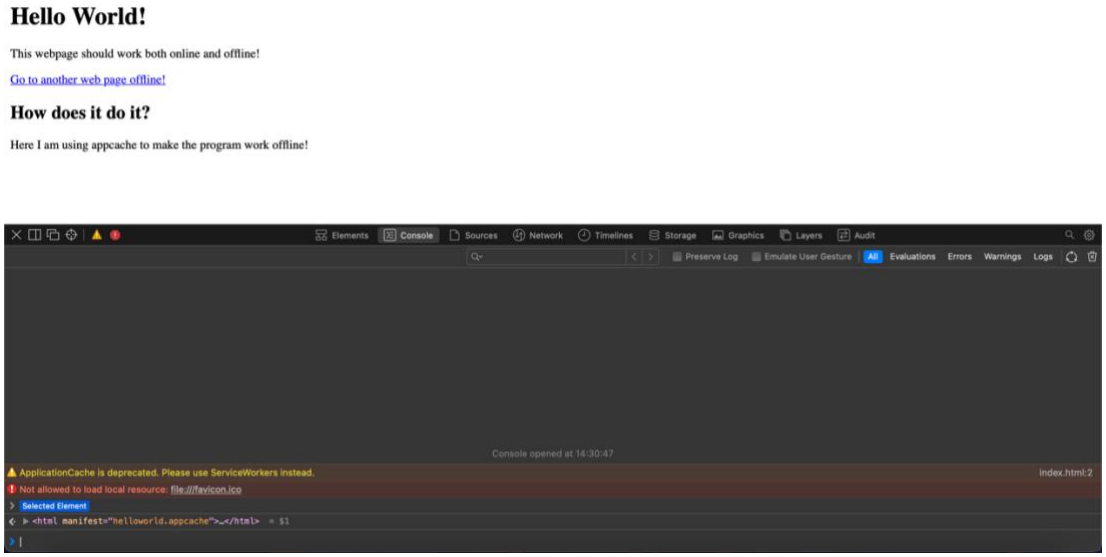

Figure 15. Screenshot displaying the error message displayed informing that application cache is deprecated and to use service workers.

The next iteration using service workers proved much more effective. Not only did the webpage continue working offline as if there was an internet connection, but the resources which were specified to be stored for offline use were clearly shown as having been cached using this technique. Compared to the application cache method, a lot more programming was required to get

the service worker to be created, installed and activated. However, once this has been setup once, the JavaScript functions can be called from any webpage on the website with minimal changes needed to be made to the code to function across the whole website.

The screenshot below shows the registration and activation of the service worker on Google Chrome:

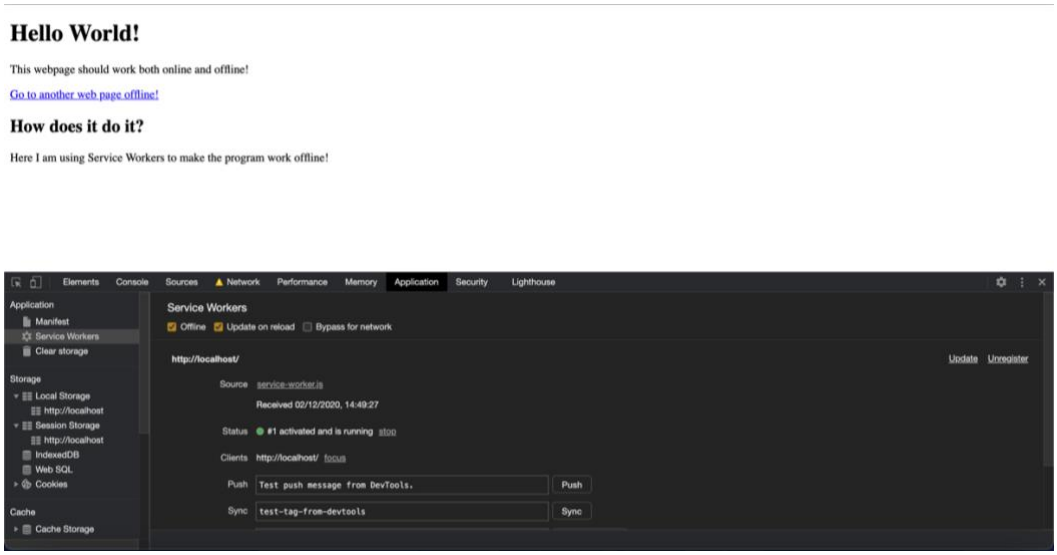

Figure 16. Screenshot displaying the working service worker for offline web application use.

What was learned from creating both of these proof-of-concept programs was that the capabilities of the manifest/application caching method were quite limited and when working with map data and other elements of the website, the better option may be to use a service worker/number of service workers to get the final product to work offline. The expansion capabilities would also be more beneficial of working with service workers, as it could be implemented to capture a specific number of map data tiles when storing for offline use with some modifications to the JavaScript code. This means that the map in the final product could be more functional while working offline.

6.3 A "To-Do List" Application Using IndexedDB

The second proof-of-concept program which was created was the "to-do list" application which used an IndexedDB. To create this proof-of-concept program, the code was based off of a tutorial from Matt West from tree house blogs [37]. The way that the application works is by using JavaScript to connect to the IndexedDB database and using that database to store what was inputted by the user. This is then outputted to the HTML document within a predefined list element within the HTML.

The screenshots below show how the proof-of-concept looked and functioned after creation, along with the storage facility within the inspect element feature of the browser:

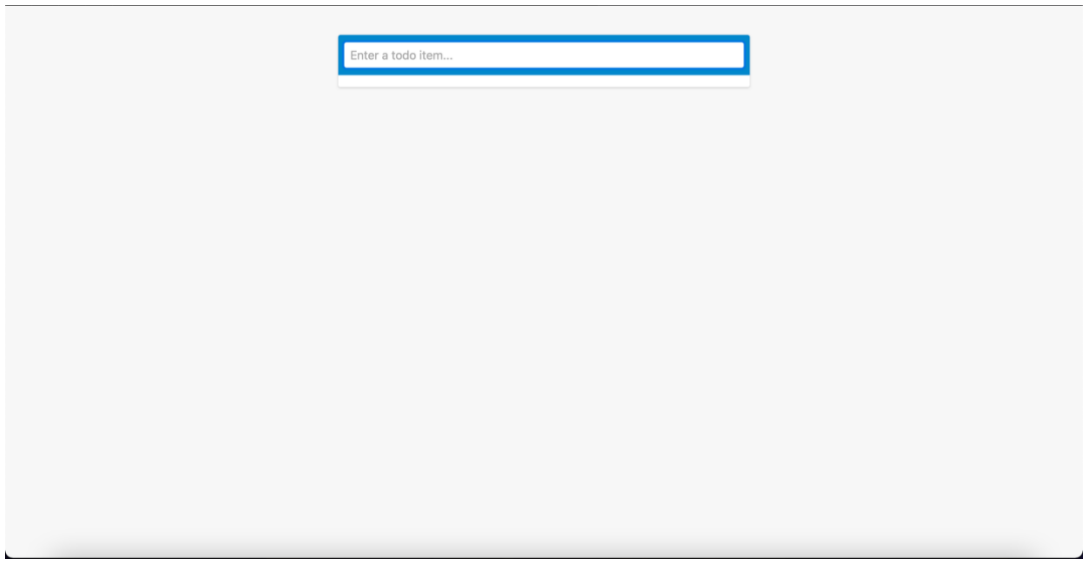

Figure 17. Screenshot showing the blank "to-list" application created.

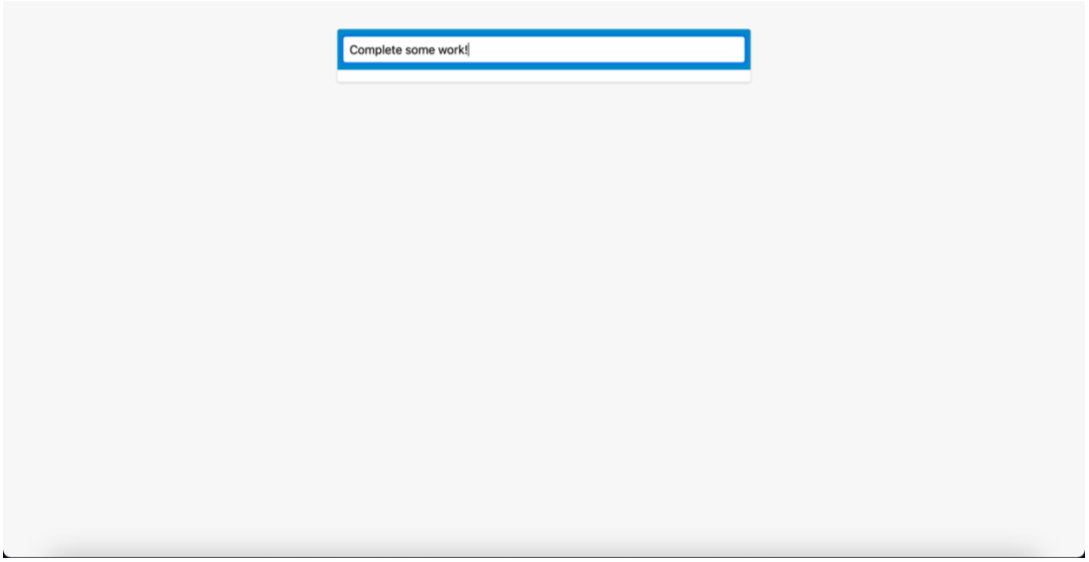

Figure 18. Screenshot showing how users would input data into the application.

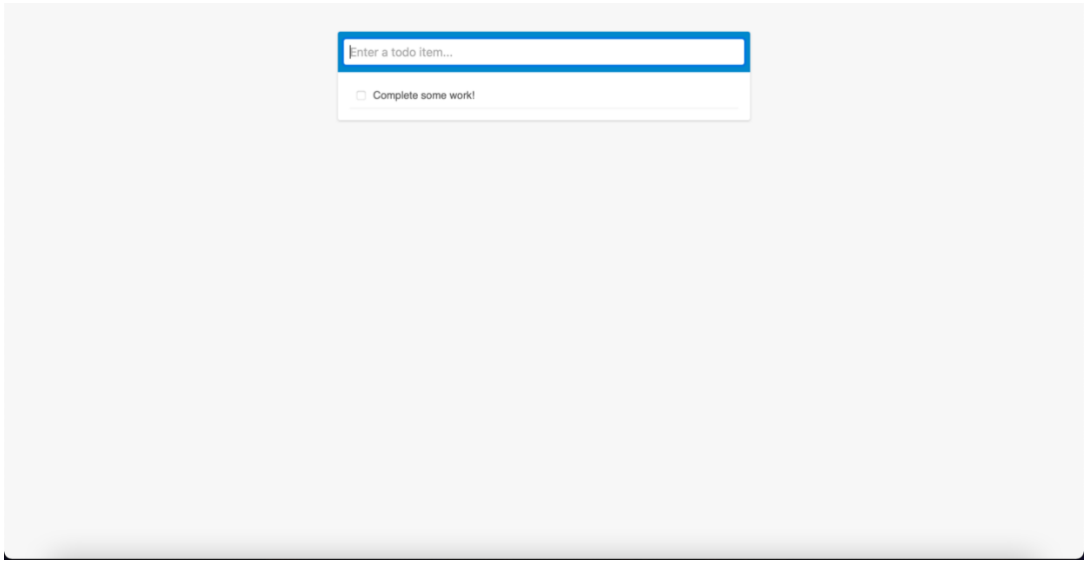

Figure 19. Screenshot showing the users input stored in a formal "to-do" list beneath the user input section.

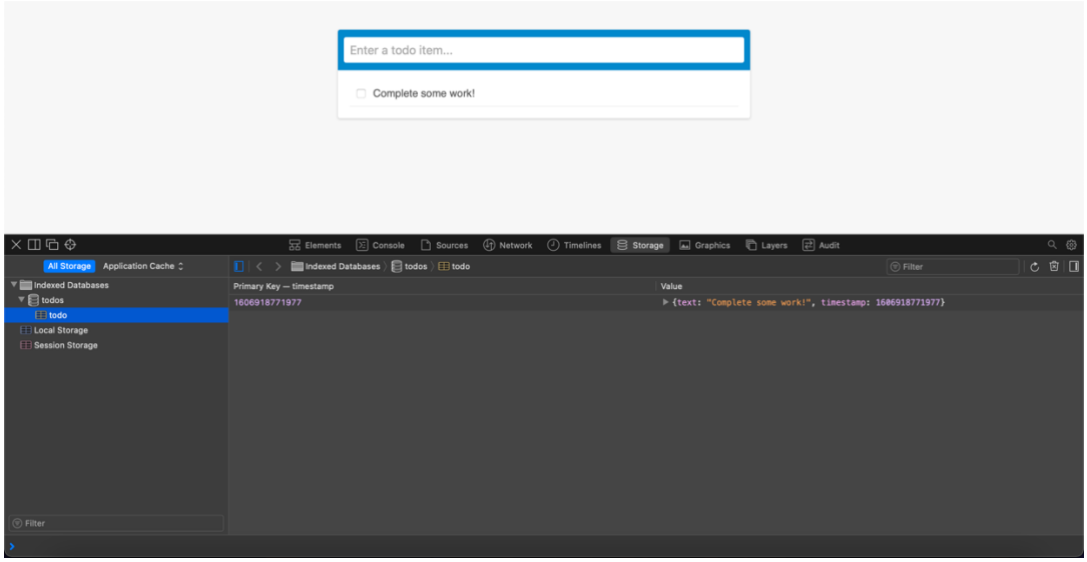

Figure 20. Screenshot showing what has been stored in the IndexedDB database, which is the user's input.

What was learnt from creating this application was that, by using IndexedDB, even when the user leaves the webpage and reloads it, the data that was inputted previously will remain in place. This is a good starting place for storing user input and other data from the end user. This could be useful in the mapping application creation, as it could prove useful in storing favourite AONB locations for example. The data, for this example, gets stored in the IndexedDB database as an object, which has two objects the text/string object and the timestamp which is an Enum. However, in creating the final product the saving of locations was not implemented which meant that IndexedDB was not used. However, it would prove useful if implementing future enhancements into the project.

6.4 Drawing Shapes Using HTML5 Canvas

The next proof-of-concept program created was the application showcasing how to draw shapes using the HTML5 canvas feature. The purpose of creating this application was to test the capabilities of the canvas feature of HTML5. Based off online tutorials from a number of sources, the application tests all of the capabilities of canvas which could be helpful in creating a mapping application. This included drawing shapes on the canvas and lines/gradients which could be used for different aspects of the map.

The first tutorial followed from W3Schools [15] showed how to draw a number of different shapes, such as circles, squares and triangles. They also had a tutorial on drawing coordinates, which from testing could be useful in a route mapping aspect for a static image map tile, showing a walking route through an AONB for example.

The screenshot below shows an example of the output that can be seen when creating a number of canvases onto the webpage, each drawing a different canvas element onto the screen:

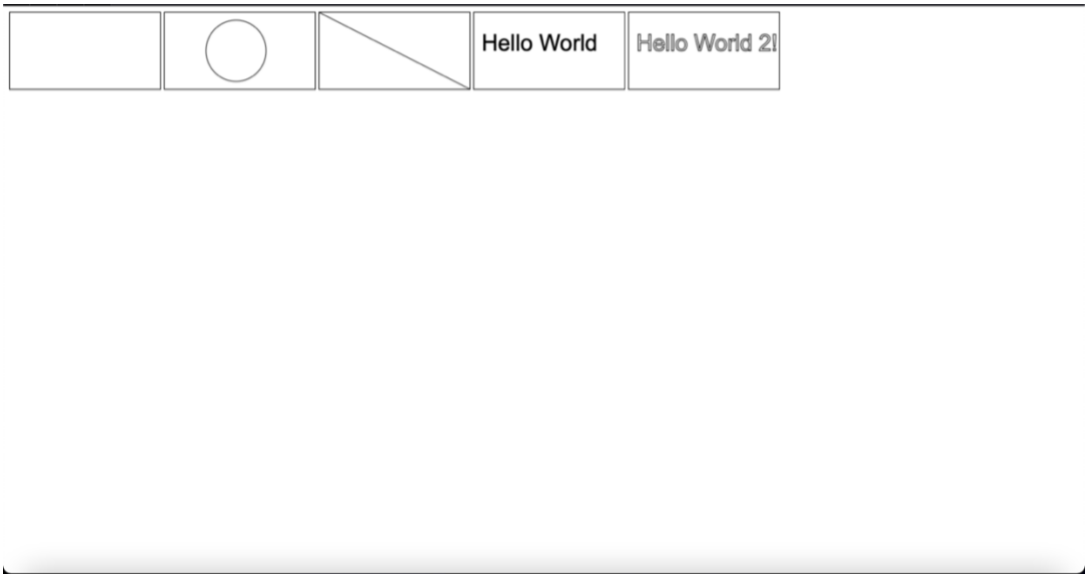

Figure 21. Screenshot displaying the output of the code for using HTML5 Canvas. More detail of the code used for this output can be seen in appendix B.

In terms of the capabilities with maps, from researching how this could be used for a map application. It was found that some shopping malls use the HTML5 canvas feature to create interactive maps of the shopping malls. This can be seen from the article from Alexander Ivanov at Arcadia [38].

Overall, after putting some time into both researching HTML5 canvas and creating the proof-of-concept program it was found that this could be a powerful tool in creating the final mapping application. HTML5 canvas expands what is capable within the map itself and will allow for more features to be highlighted and extendable features which would improve the look and functionality of the mapping application. However, when building the final product, it was found that there were technologies more closely related to both LeafletJS and Mapbox which worked better in displaying different objects onto the maps, including multiple markers and outlining the regions of the AONBs which was achieved through the use of GeoJSON objects.

6.5 Loading and Displaying Open Street Map Data

A proof-of-concept program was then created which was a webpage which loads and lists OSM data, along with some other features which could be useful to practice implementing before the creation of the final project deliverable. This proof-of-concept program was broken up into two different iterations. One which uses Leaflet to display the map data and one which uses Mapbox.

The first proof-of-concept program which was created was the OSM data being displayed and manipulated using Leaflet. This iteration was fairly straightforward to implement, with a lot of good tutorials supplied directly from Leaflet JS themselves. Also, with it being open source, there was a number of available plugins which can be either downloaded and referenced in the project files or can be designed from as inspiration for other features of the map. All features that need to be included in the map would be able to be generated using Leaflet, however some of them may become time consuming is what was found from the creation of this proof-of-concept program. However, overall Leaflet was a good technology for displaying the OSM data onto a web page.

The screenshot below shows how the proof-of-concept program was outputted, i.e., showing the OSM data through the use of Leaflet JS:

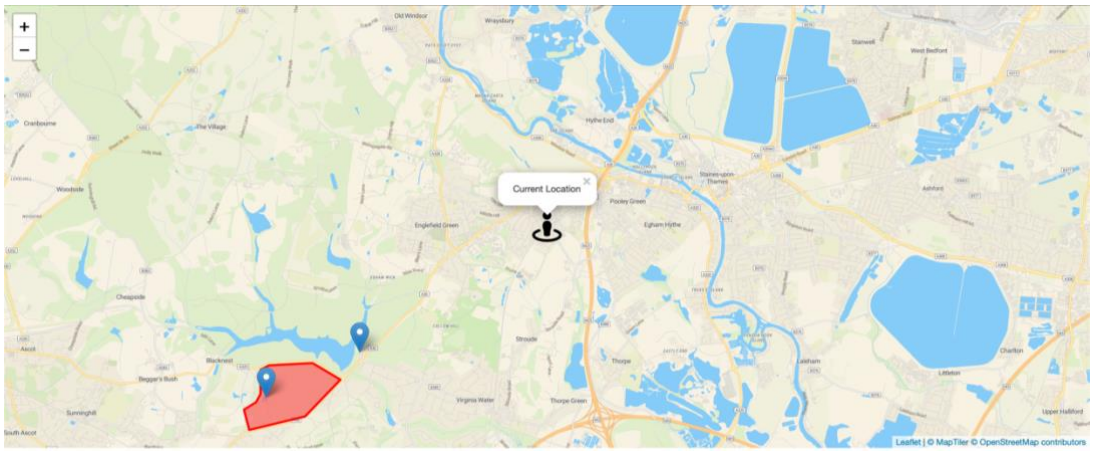

Figure 22. Screenshot showing the output of the map data using Leaflet JS to display the data.

As shown in the screenshot above, the proof-of-concept program tested not only display the base map from OSM using Leaflet, but also delved deeper into displaying some of the other mapping elements that may be useful in the creation of the final mapping application, such as adding custom markers (current location marker) and standard location markers/boundaries in the form of coloured polygons.

The second iteration of displaying the OSM data was by using the Mapbox API. This was a very simple method of getting the OSM data to be displayed, along with any other features that a designer/creator may wish a map to have. For example, when implementing a search feature or a route mapping feature, it is all built into the API itself. This means that it just needs to be called through the use of JavaScript to be fully functioning on the map on the web application. This is a major timesaver and great for convenience, however when attempting to customisation these features is when issues could be had. There is limited allowance for customisation of the pre-built features, unlike Leaflet. Overall, Mapbox is a great way of easily implementing a map onto a website/web application and by creating this proof-of-concept program, a greater understanding of the technology was gained.

The screenshot below shows how the proof-of-concept program was outputted, i.e., showing the OSM data through the use of Mapbox API:

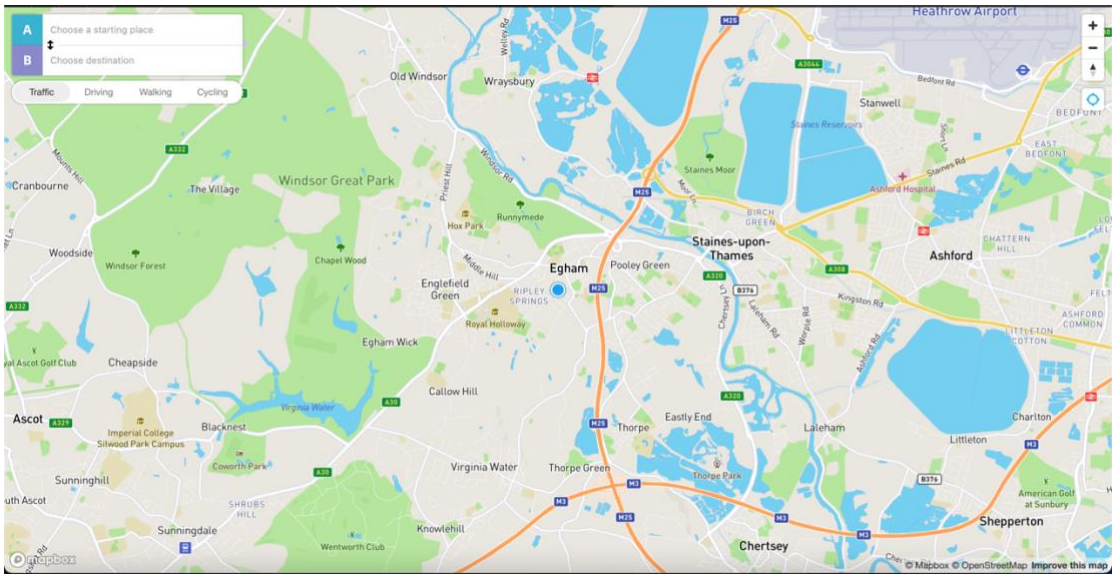

Figure 23. Screenshot showing the output of using Mapbox API to display the map data onto the web application.

Just like with Leaflet, Mapboxes other tools and extensions were testing within the proof-of-concept program. This included current location finding and a route mapping element, which allows for user input in the form of two locations and generates the fastest route between the two.

Within these proof-of-concept programs, extra features were also created/tested. Such as with the Leaflet example, a custom search box was implemented using JavaScript, Leaflet JS and GeoJSON. When the user types into the custom displayed search bar, the algorithm checks through JSON list to see if the string inputted matches any of the stored locations. If it does, it moves the map to that location, based on its latitude and longitude, and displays a marker on the map with a custom text box which shows the name of the location.

6.6 PHP and MySQL User Registration and Login

At the start of the second term, more proof-of-concept programs were created to better understand the backend of the program better. This included programs which focused on users being able to create accounts and login to the accounts which they had created. The subsections below detail these proof-of-concept programs.

The first of the proof-of-concept programs which were created was a user accounts feature using the PHP programming language and MySQL database. The first step in setting up the application is to create the required directories. The image below shows the directory setup:

Figure 24. Example of the file structure for a PHP user registration and login system. With "|-" being folders and "|-" being the different files within the folders.

The config file was the first to be completed. In this file, a connection to the database is created. The way that this is done is by defining four variables, which will contain the information to be able to connect into the database which was created. The code below shows how this was achieved with a MySQL database accessible through the machines localhost server.

```
define("DB_HOST", "127.0.0.1");
define("DB_NAME", "phpproject03");
define("DB_USER", "jack");
define("DB_PASS", "Westham4123");
```

Once the variables were created for connecting to the database, these could then be used within the different PHP classes to connect to the database and perform the registration or login based on the actions within the different classes. In general, the way that the classes work is that they firstly check to see if what the user has entered is valid and that there is no information missing. If there is missing information, an error will be thrown, and the user will be informed with an error message on the website's frontend. However, if no error is thrown and therefore all of the information is present, then it will establish a connection to the database, check if the user is already registered (i.e., the data entered is already within the database), and for registration if it is

not, it will then use the SQL command shown below to insert that user's information into the users table within the database.

```
INSERT INTO users (user_name, user_password_hash, user_email) VALUES('" .
      $user_name . "', '" . $user_password_hash . "', '" . $user_email .
      "');"; $query_new_user_insert = $this->db_connection->query($sql)
```

For the login class, it will check that the users' data is present in the database and if it is will allow the user to login to their account. The following SQL statement was used to check if the details were within the table.

```
SELECT user_name, user_email, user_password_hash FROM users WHERE
      user_name = '" . $user_name . "' OR user_email = '" . $user_name .
      "';";
```

By using the two different classes, the visitor is then able to register for an account and login to that account that they have created. When a user does log into their account, they will be taken to a view which is for being logged in, which uses data from the database to display their name on the website. The screenshots below show the process of registering for an account and logging into it.

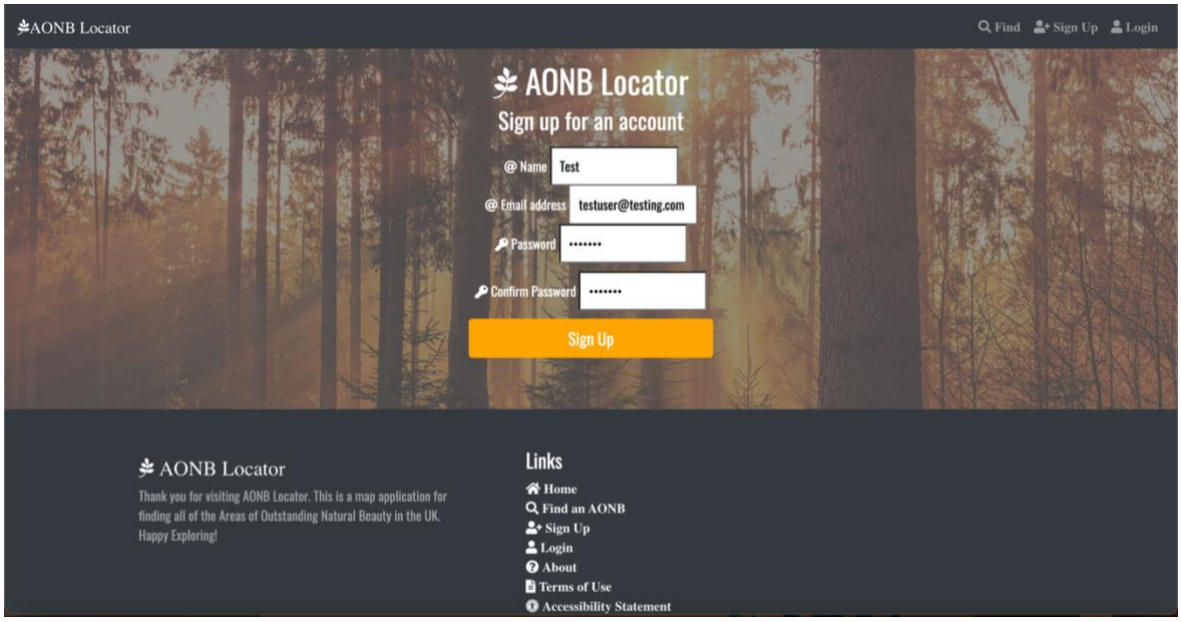

Figure 25. Screenshot showing the user entering their details into the signup/registration form on the signup webpage.

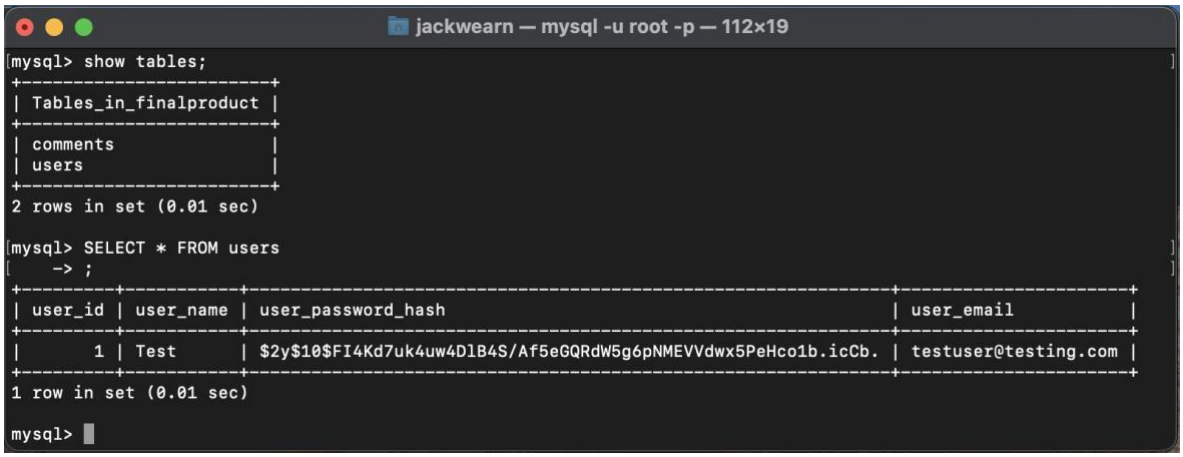

Figure 26. Screenshot showing the users details now stored within the MySQL database. Here they have the user_id of 1, name of Test, a hashed password and the user_email.

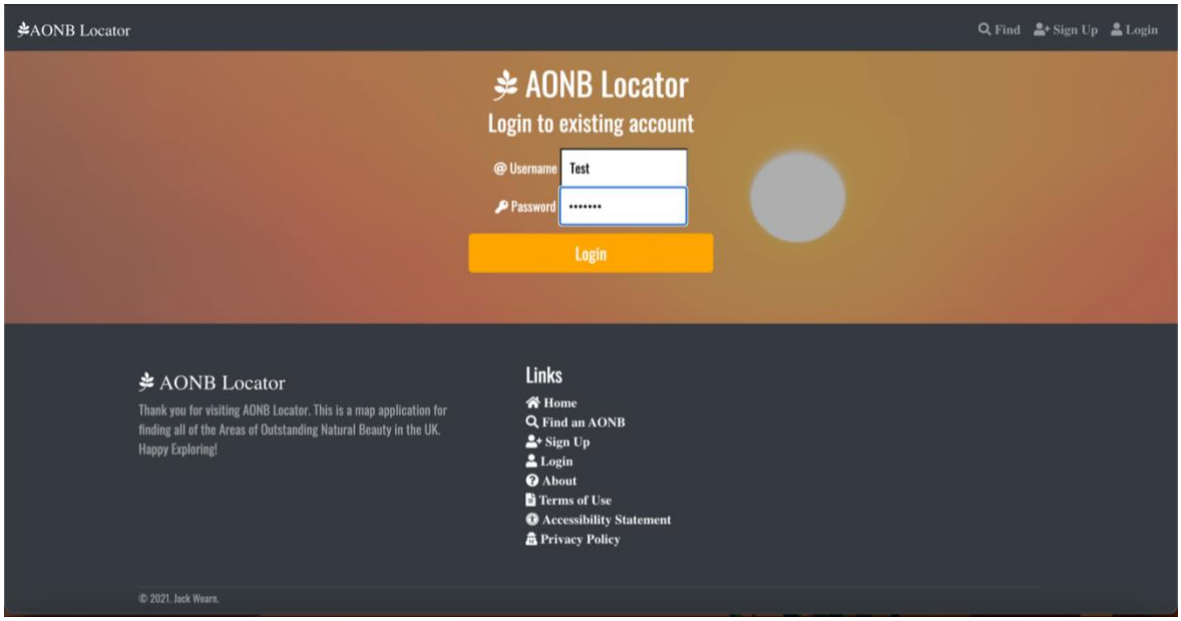

Figure 27. Screenshot showing the "Test" user entering their login details on the login webpage.

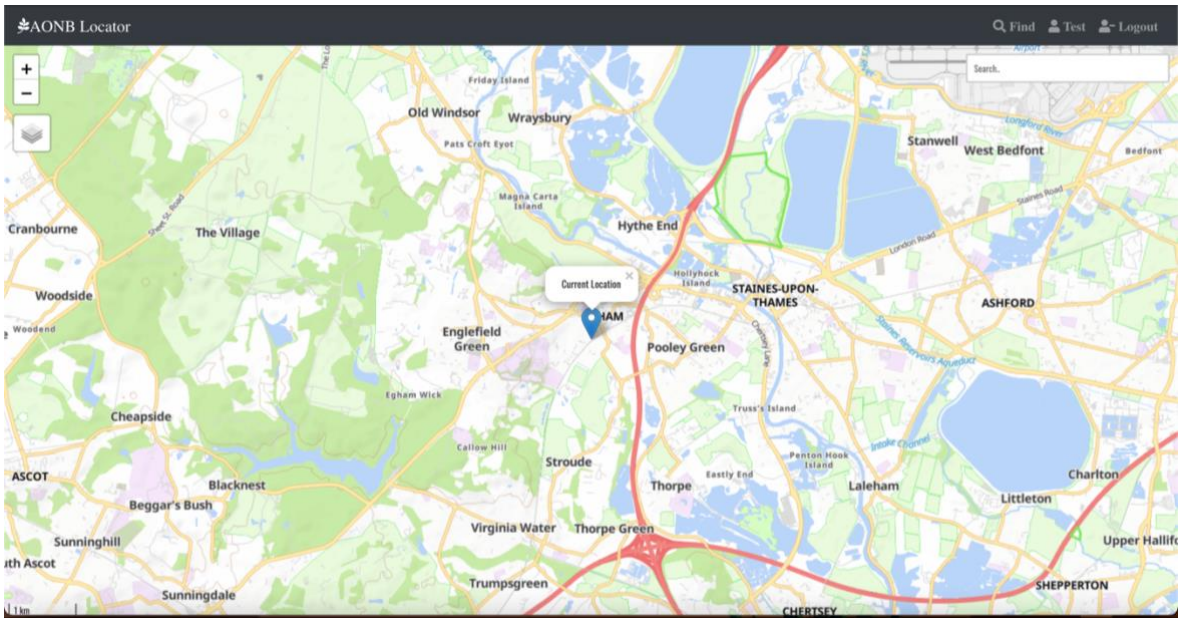

Figure 28. Screenshot showing the user now logged into their account and having been returned to the home page.

In terms of the password hashing, this was performed by the PHP 5.5's simplified password hashing API created by Anthony Ferrara [39], which is a free open-source API which efficiently hashes the user's password. The way that this API works is that the password which is entered by the user is hashed using an algorithm, which changes the characters that have been entered into a number of different characters. This is then used both when registering (i.e., hashing the user's password so no one can see the plaintext of the password unless they know the specific algorithm which has been used) and when logging in, where the password entered is checked against an un-hashed version of the stored password. If they are a match, the user will be able to login

successfully. If the passwords do not match, the user will be prompted that the entered password is incorrect.

Overall, this proof-of-concept program was one of the more useful tutorials to follow as the idea of how this worked was used in the creation of the final product. The reason for this is because the PHP language was used in order to build the application and this registration and login system worked the most efficiently.

6.7 User Comments System

The final proof-of-concept program which was created during the course of the second term was a user comments system. This was a program which allowed any visitor to add a comment, which would then be displayed on the screen. The user could either add a new comment or reply to a previously added comment. This proof-of-concept program was created whilst following a free tutorial from David Adams on CodeShak [40].

The first step in creating this proof-of-concept program was to create the database, which will store all of the comments that have been submitted and will also be used to fetch the data from in order to display the comments onto the webpage. The below is the SQL code which was used to create the required database.

```
CREATE TABLE IF NOT EXISTS `comments` (
       `id` int(11) NOT NULL AUTO_INCREMENT,
       `page_id` int(11) NOT NULL,
       `parent_id` int(11) NOT NULL DEFAULT '-1',
       `name` varchar(255) NOT NULL,
       `content` text NOT NULL,
       `submit_date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
       PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
```

This would then create the table, with the credentials for an ID, page ID, parent ID, name, content and comment submission date. The screenshot below shows the database ‘phpcomments’ which was created for this proof-of-concept program and the table ‘comments’ created from the code above.

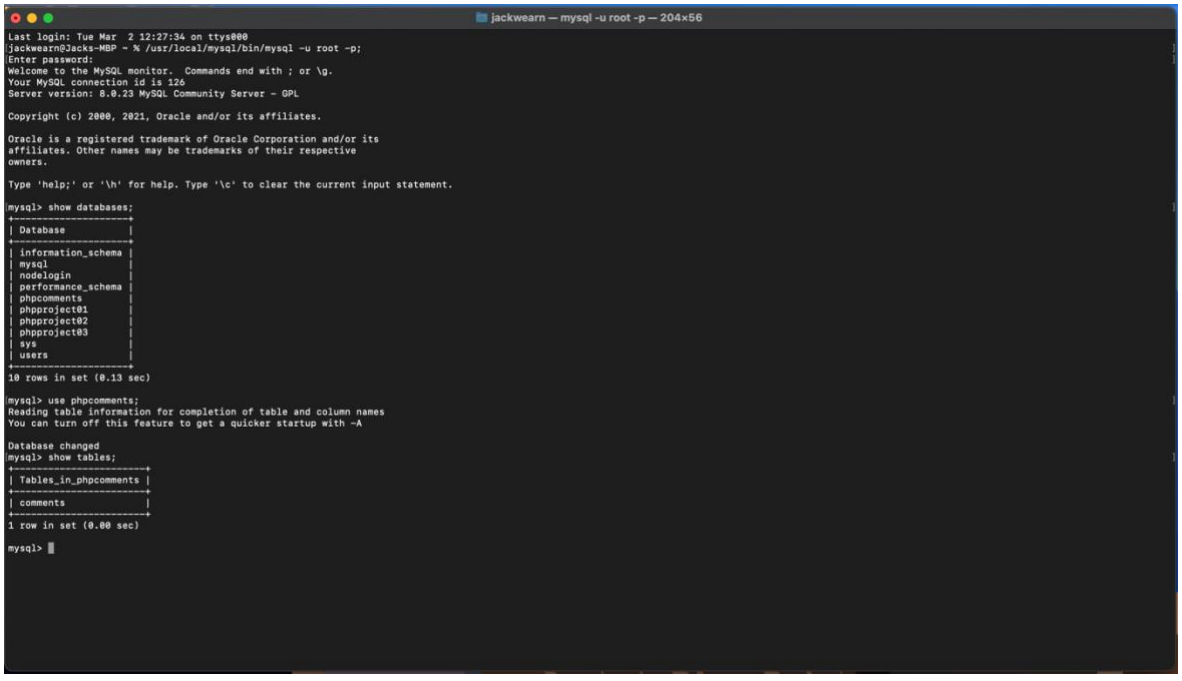

Figure 29. Screenshot showing the database and table created for this proof-of-concept program.

If you then wanted to see what the table is populated with, you can use the command "SELECT * FROM comments;". Which will display all of the table's contents as shown below.

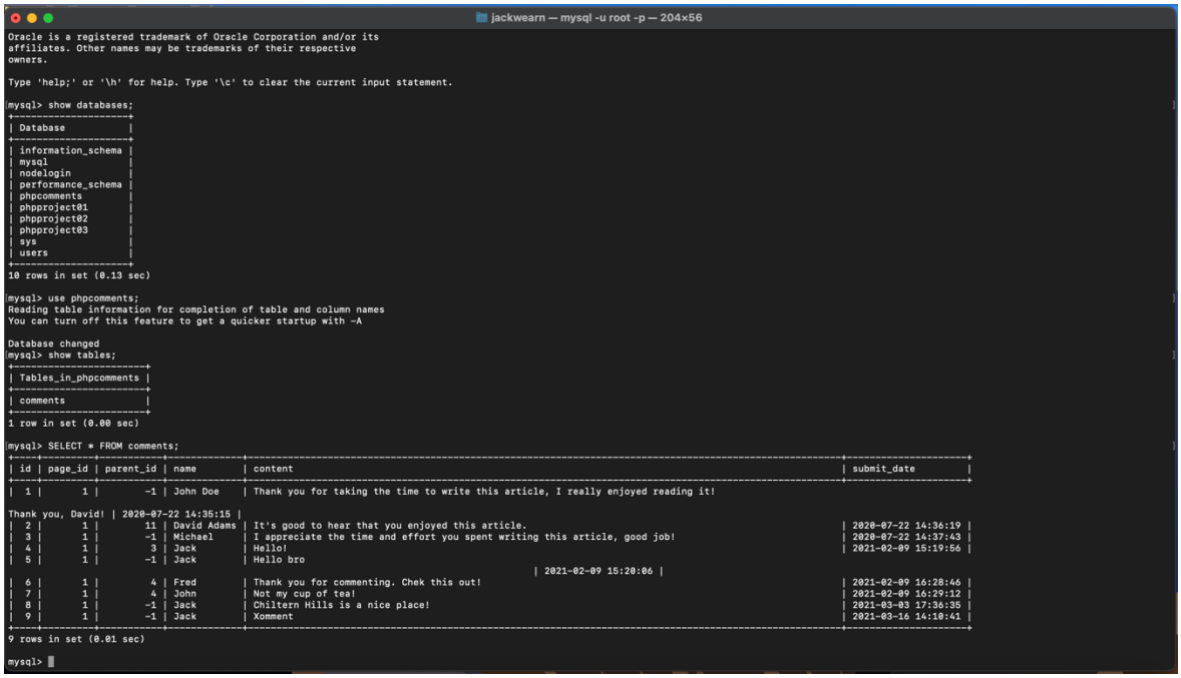

*Figure 30. Screenshot showing the populated table with different users' comments when using the 'SELECT * FROM comments;" SQL command.*

The next step was to create the styling for the comments system. This is how the comments will be displayed on the screen to the user. There are a vast number of ways in which this can be done, and so will not go into full details of how this was done. However, as an example the replies class, which is used for displaying the comments which are replies to another comment, have a padding to the left to show them as being indented. By doing so, the reply is easier to spot for the users and it can be identified as a reply much quicker.

The next step was to create the PHP code for the comments system. This involved connecting the database and table which had previously been setup using MySQL and creating functions which loop to add new comments onto the screen so the user can see them. This code could then be used to display the comments section onto a webpage. This is shown in the screenshot below.

Figure 31. Screenshot showing the working comments section for the proof-of-concept program.

This proof-of-concept program gave a better understanding of how a comments section can be used and implemented for different purposes. It also gave a better understanding of PHP, such as connecting to the database and table and displaying HTML data through PHP code.

Chapter 7: **Term One Prototype Development**

7.1 User Interface Design for Prototype

The creation of the demo application was based off the low-fidelity prototype which was developed using physical designs. The UI designs that were produced can be seen from the photographs below. Before implementing them into a piece of software, the designs were tested for their functionality. This involved attempting to "use" the low-fidelity prototype and ensuring that everything made sense on paper. The actual production of the demo application did slightly differ from the drawn prototypes as some elements which were drawn up proved difficult to translate into actual web elements. The photograph below shows a basic mock-up of what the UI should look like when building the demo application.

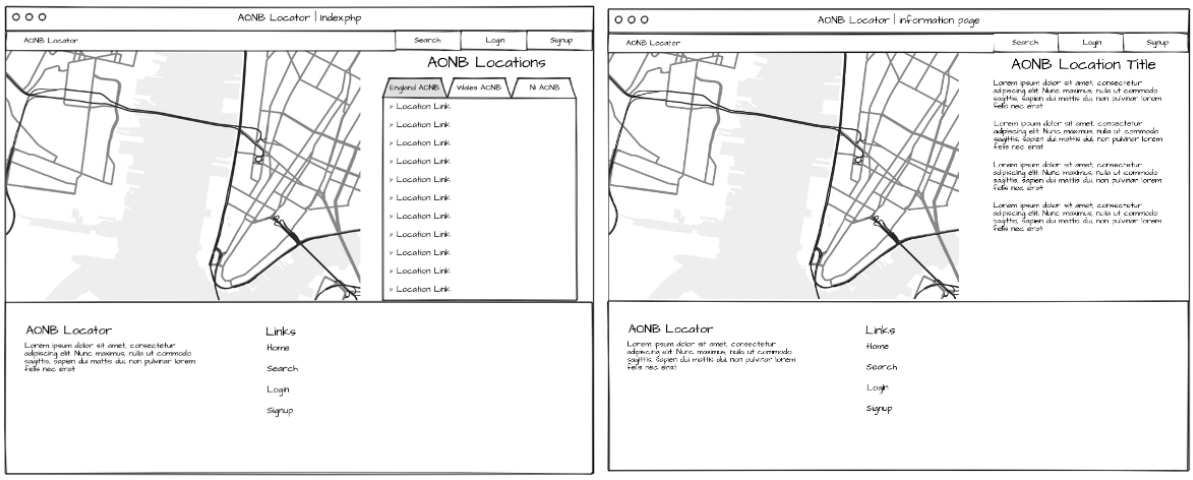

Figure 32. Photographs showing the hand drawn designs for the prototype of the demo application, which is of low detail i.e., low fidelity. Created using MockFlow [20].

7.2 Demo Application

As an opportunity to put all the knowledge gained from these proof-of-concept programs, a demo application was also created to give an idea of what the final product could look like. This was created based on the UI design which was also created during the term. This had some of the functionality of the what the final product will have and gave an opportunity to showcase what had been learned already during the first term phase of the project. It was also showcased during a project meeting, to give an idea to the supervisor of the project on exactly how it will function and what had been achieved so far during the term. This was also written about in a project diary entry [4] [Appendix B]. The screenshot below shows how the demo was created:

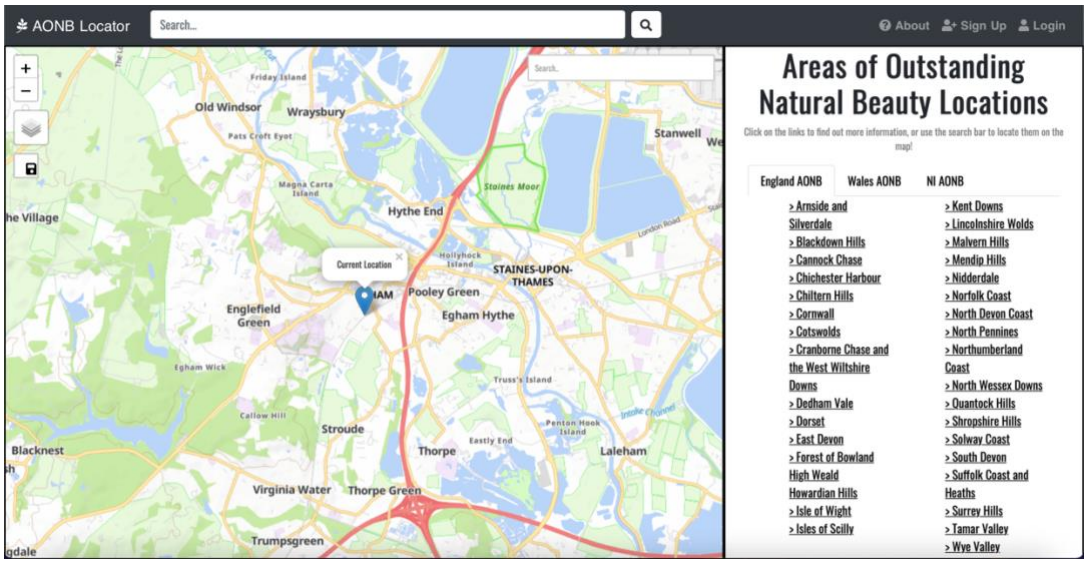

Figure 33. Screenshot displaying the final product of the demo which was created to test the knowledge gained through the creation of the proof-of-concept programs.

A working search bar was also created on the map. When a user searches for an AONB, which is stored within a JSON variable, the map moves the user to the correlating location and creates a marker in the searched location. This can be seen from the screenshot below:

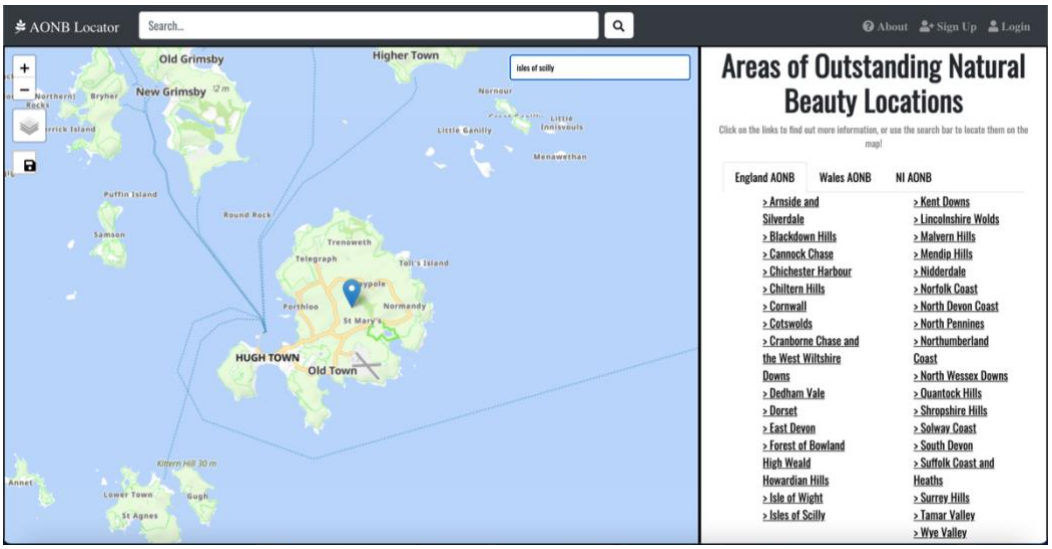

Figure 34. Screenshot showing what the map does when the user inputs a search for an AONB location.

The next screenshot shows the template for an information page, which will highlight the location with a static map from Mapbox and information on the right-hand side:

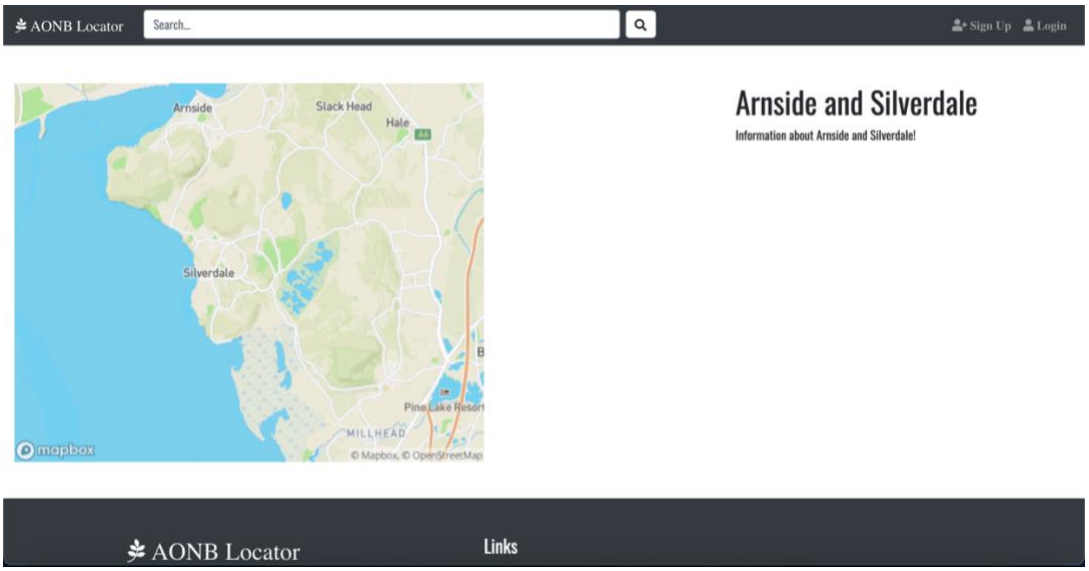

Figure 35. Screenshot displaying the template to be used for the information pages when clicked from the index web page.

Chapter 8: **End Product Development**

8.1 Introduction

Within this chapter, the end product will be discussed. It will start off with the preparation which went into the development of the end product. This included things such as creating the file structure, database and tables, and making any changes to the UI design. The architecture of the product will then be discussed before looking at each of the different features of the end product. This will include screenshots of the finished web application and snippets of code which went into making these features work. Any issues which were encountered when developing the end product will also be discussed before going over how to run the application and a look at the work log for this development. Finally, any potential future enhancements will be discussed. These are features, tools or technologies which were included in this product that could have been to further enhance the project.

8.2 Development Preparation

The preparation stage of the end product involved setting up a number of elements which will define how the application will be created. Preparation is key in understanding the capabilities of the application and better understand what will be needed. This included setting up the file structure of the product. For example, creating the specific folders/directories for the styling files (CSS files) which are usually kept separate to the main development files. Below is a file structure of the final product.

```
\ - AONBLocator
      \ - backgroundImages
            | - background-login.jpg
            | - background-signup.jpg
      \ - classes
            | - Login.php
            | - Registration.php
      \ - config
            | - db.php
      \ - informationwebpages
            | - {Each different webpage for all AONB locations}.php
            | - header.php
            | - headerloggedin.php
            | - footer.php
      \ - libraries
            | - password_compatibility_library.php
      \ - scripts
            | - mapScript.js
      \ - styles
            | - comments.css
            | - loginStyles.css
            | - styles.css
      \ - views
            | - not_logged_in.php
      | - accessibility.php
      | - footer.php
      | - header.php
      | - headerloggedin.php
      | - index.php
      | - log.php
      | - login.php
      | - privacy.php
      | - register.php
      | - script.js
      | - service-worker.js
      | - signup.php
      | - termsofuse.php
```

Figure 36. Screenshot displaying the final file structure of the finished product. "\ -" denotes a folder and "| -" denotes a file. In place of showing all of the AONB information pages, "{}" represents each file name based on the locations name.

Another step taken in the preparation of the final application was to setup the database. In the previous iteration, the demo application, a database was not in use. For the purposes of this project, this was initially done via the localhost (host machine). When setting up the database, it was decided that there would be a need for a table for the users to register into, which will also be used when a user wishes to login and another table which will be used for the user comments to be stored. Below is a screenshot of the database structure, which is using MySQL.

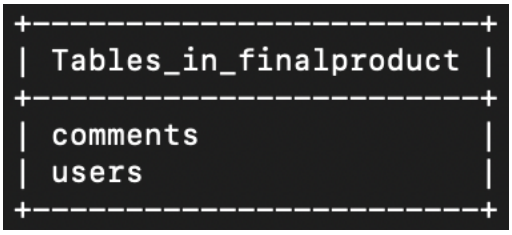

Figure 37. Screenshot showing the tables which are within the 'finalproduct' database, which is the 'comments' and 'users' tables.

Figure 38. Screenshot showing the two tables which are used in the MySQL database.

The final step in getting prepared to undertake the development of the end product was to make any final changes to the UI. Any changes would be based on the feedback received after presenting the demo application from term one. There were a number of changes that were decided upon to be made, and so new designs were created to tailor to these changes. These designs can be seen below created using Figma [19].

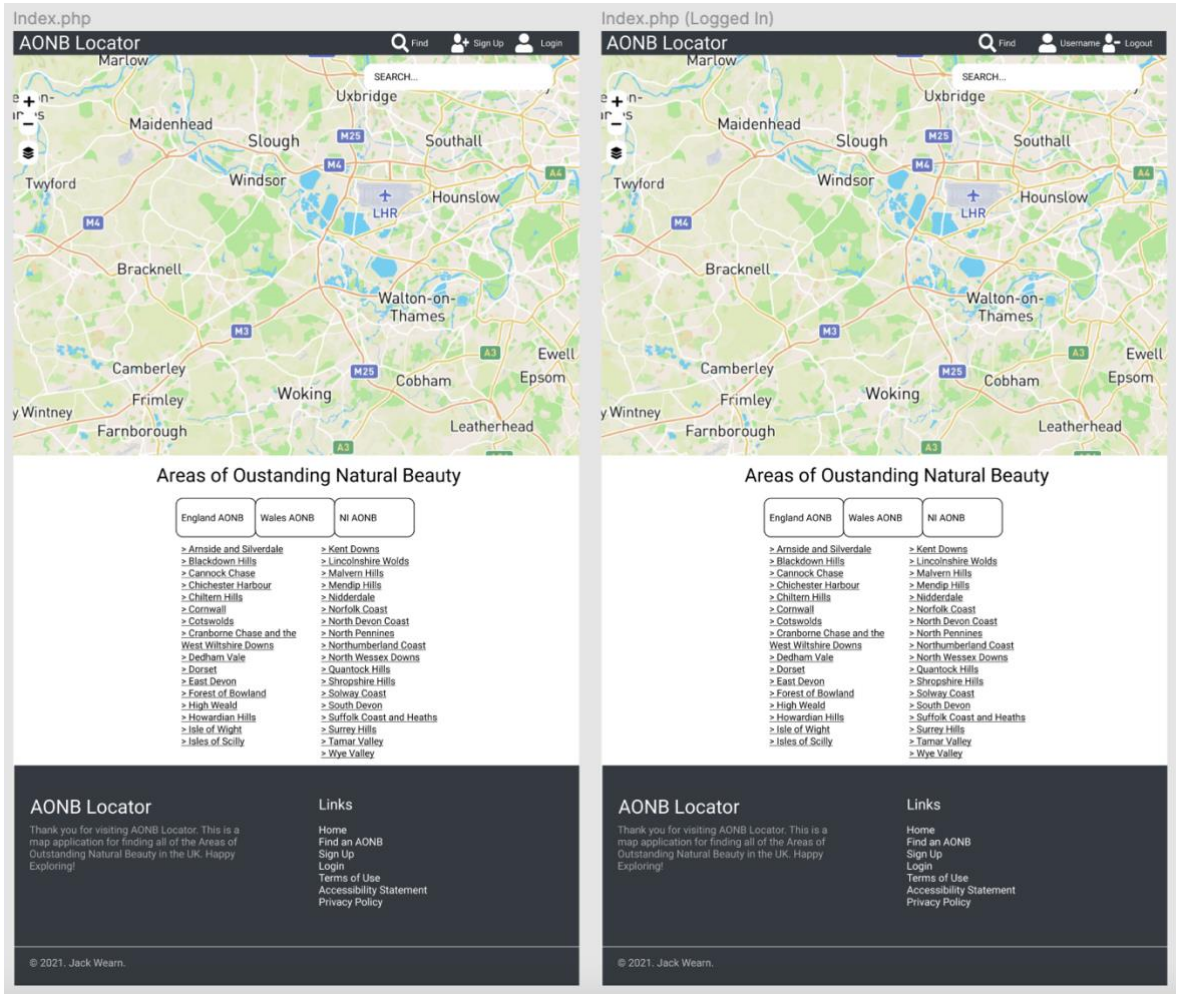

Figure 39. Screenshot showing the index pages new designs using Figma [19]. This shows the screen when the user is logged in and not logged in.

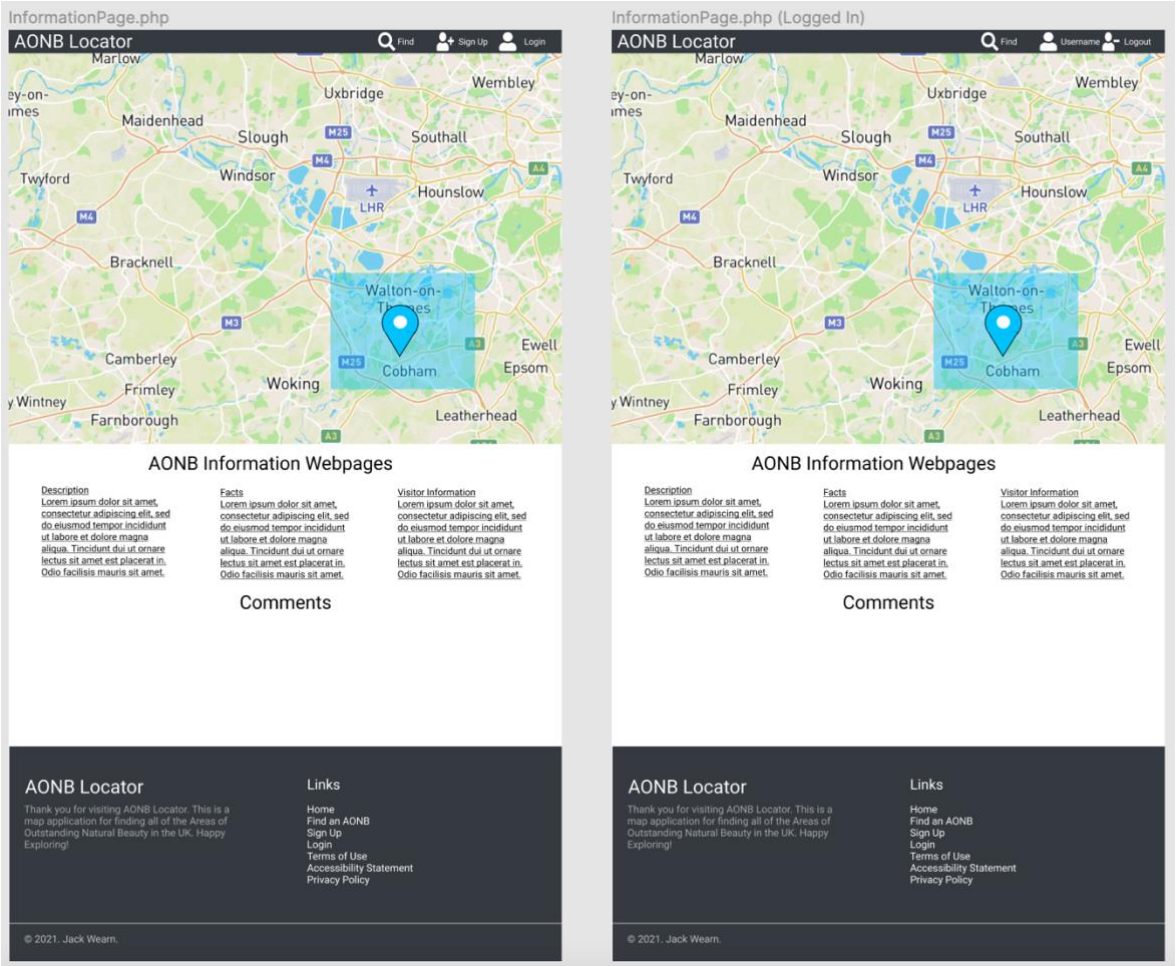

Figure 40. Screenshot showing the AONB information webpage design created using Figma [19]. This shows the what the page could look like with the user logged in and not logged in.

8.3 Architecture of the End Product

The architecture of a system is the fundamental structure of the software that is being created and the discipline of creating the structures and systems [41]. The photograph below shows the architecture diagram for the final product. It displays the different PHP web pages, how they connect to the MySQL database and what is achieved from the connection.

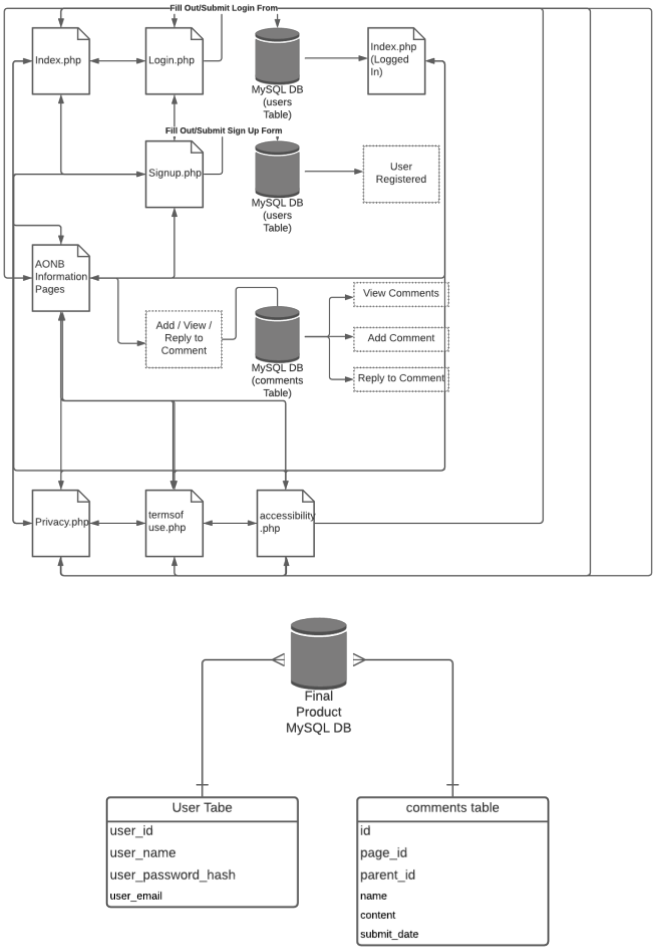

Figure 41. Photograph of the architecture diagram for the project. It shows how the difference PHP files connect to the MySQL database and tables and what can be achieved from them. It also shows how the database is structured.

8.4 Features of the End Product

In the end product there were a number of features, both basic and advanced, which were created to give the user the best possible experience when using the offline HTML5 map application. This section of the report will cover each of the different features which were included in the final product and the impact that they had on the overall look, feel and performance of the application.

8.4.1 Offline Capabilities

One of the main objectives of this project was to produce a map application which works offline. Therefore, this feature was one of the most important features to implement. The decision to use service workers were made on the basis of it being the most recent technology, in use by the most modern and used browsers, such as Google Chrome which is one of the most popular web browsers of choice. Also, one of the other main options for the offline capabilities, which is an application cache manifest is now a depreciated feature. The way in which this feature works is that when a visitor enters the site, a new service worker is created for the duration of their time on the website. This service is always running in the background, caching any of the important aspects of the application. Should a user lose connection to the internet, the service worker is in place to provide the information required locally so that the user does not experience a loss in connection to the website. The screenshot below shows the service worker starting when a user visits the site and the console which shows data being cached for the user.

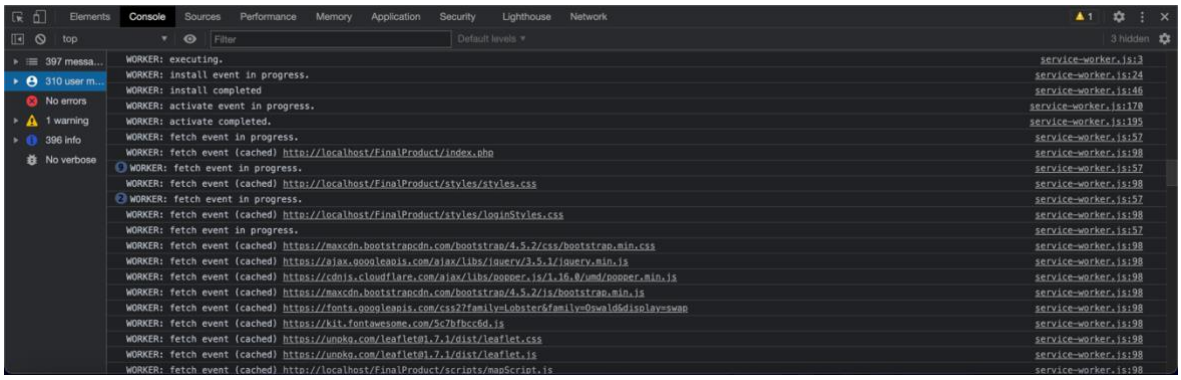

Figure 42. Screenshot showing the service worker executing and performing the fetch functions to gather all of the required cache data.

Once the service worker is running, and caching all of the important features of the site that would be needed to work offline, if the user does go offline the website still functions as if the user is still connected to the internet. Google Chrome has a helpful feature implemented into its developer tools which allows the developer to simulate going offline. This means that the browser thinks that the user has no connection to the internet. The screenshot below shows this option having been selected and that the website is still visible.

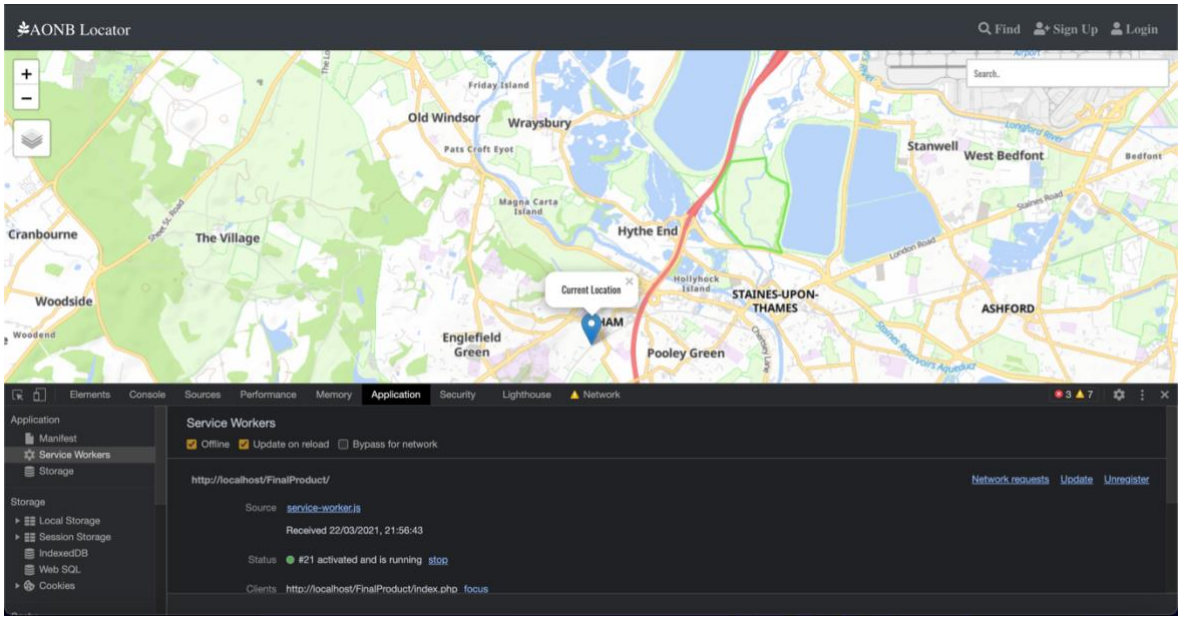

Figure 43. Screenshot showing the website still running while using the developer tool "offline".

To highlight how this would work should the service worker not be in place, the screenshot below shows the web application running in the developer tool "offline mode", which as can be seen now displays Google Chromes "no internet connection" information screen.

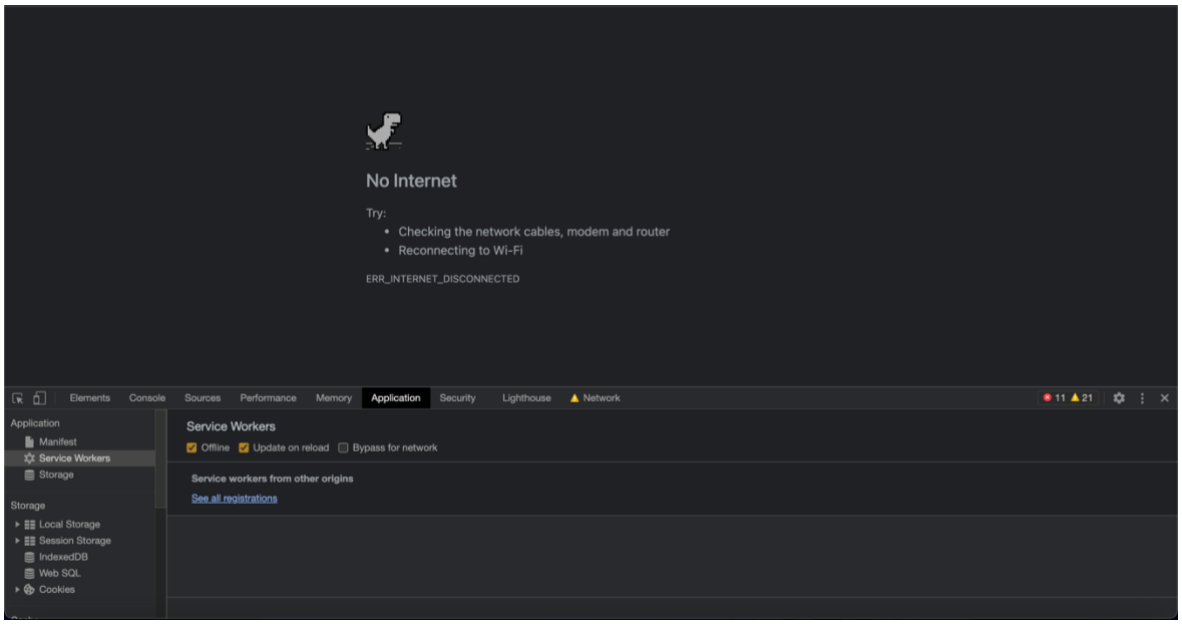

Figure 44. Screenshot showing what appears when a user goes "offline" and the service worker is not running.

In the event that a user's browser does not support service workers, which is very rare in modern web browsing as most other forms of offline application development methods have now been deprecated, the offline functions will not work for a user and they will be presented with the screen shown above. However, with most, if not all in the near future, having the use of service workers built in there was no need to implement an older service such as application caching which did cause errors in getting the service worker functioning as intended due to the application cache interfering with other parts of the HTML and JavaScript code.

8.4.2 Main Map on Home Page

The first page that any visiting the web application will see is the index webpage. On this page, they will be presented with the "main map" of the application which is simply the main map aspect of the application. Unlike the information pages, which will be seen later within this section, this map allows the user to pan, move and zoom around the map of the UK and World. There are a number of features included within the main map too. Within this portion of the chapter, these will be discussed in detail. The screenshot below shows the index webpage, which is what the user will first see.

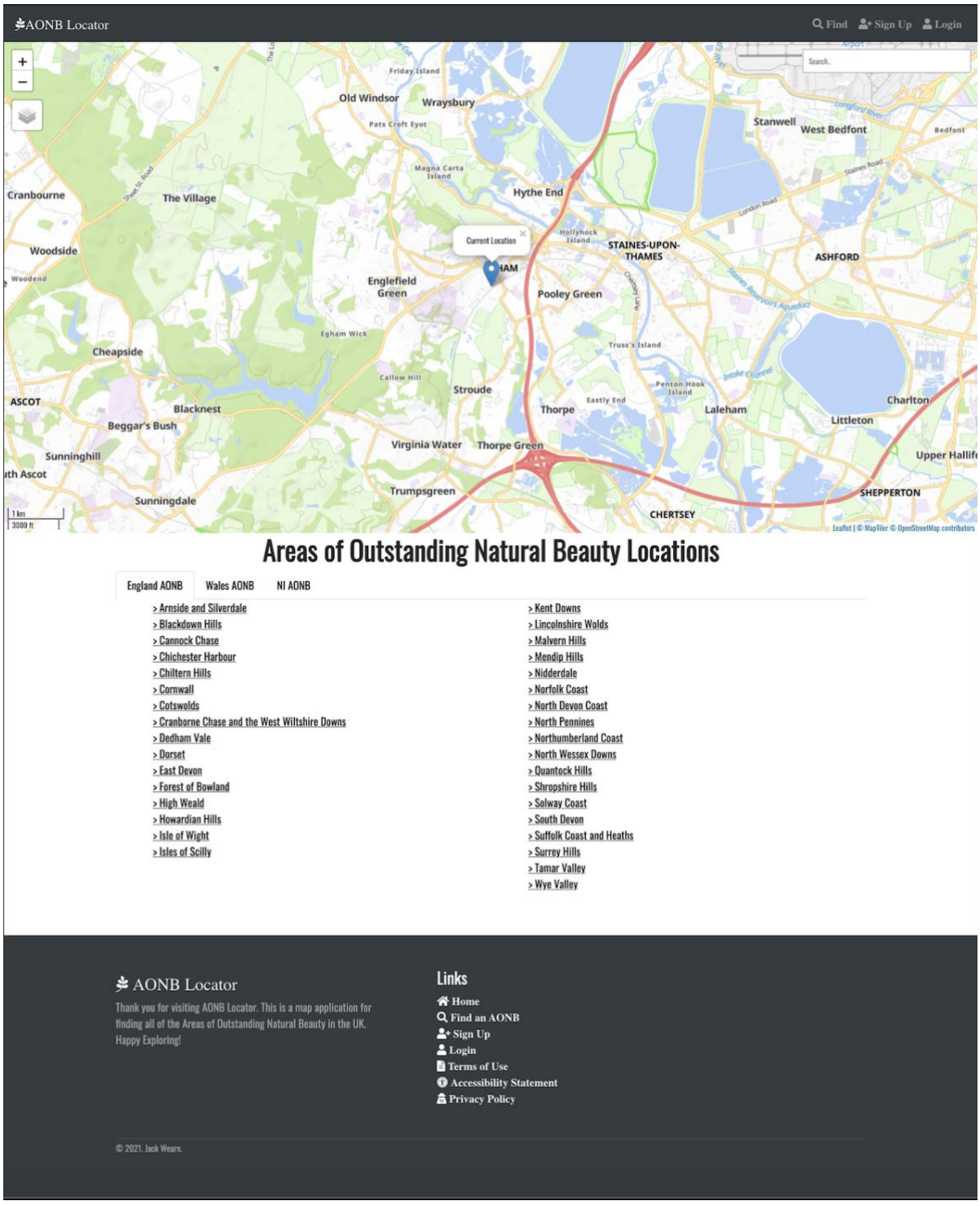

Figure 45. Screenshot showing the full screen of the home page which the user will be presented with when they first enter the web application.

Firstly, this map is implemented and displayed through the use of LeafletJS. This is all done through JavaScript code which allows the webpage to display the base map. However, this extended further to allow for the first feature which was implemented which is the ability to change the type of map tile which is being displayed. For the user to change what map tile type is being displayed they will use the option below the zoom controls on the left-hand side. When this is active, they will be presented with the two options that have been implemented, which is a natural map and a street map variant. The screenshots below show the two options once selected.

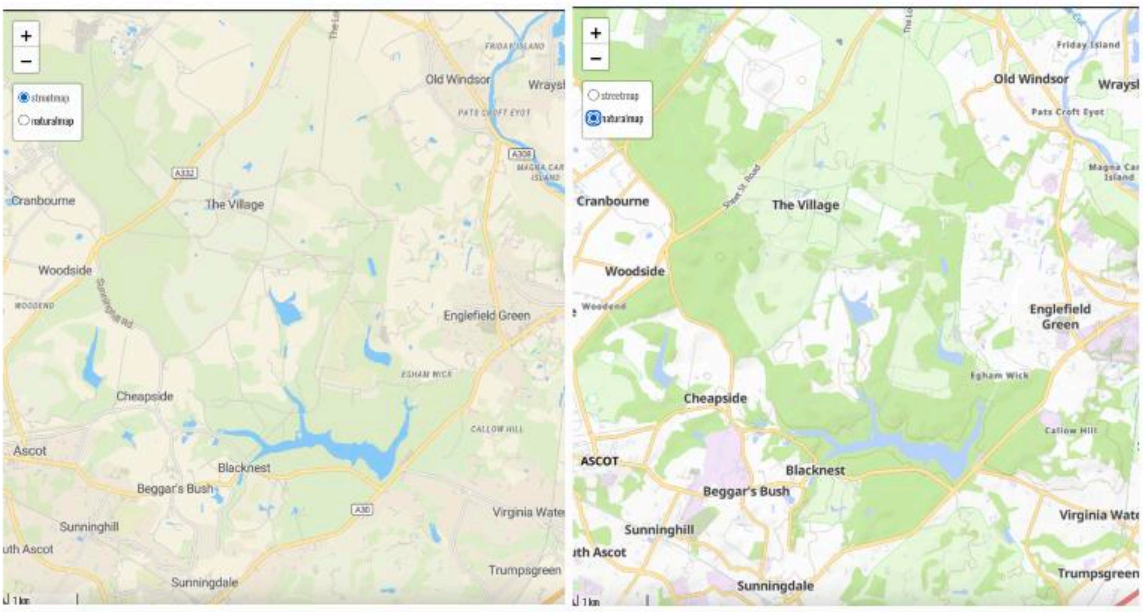

Figure 46. Screenshot showing the change when a different map tile layer is selected.

The next feature which was included in the map was the custom search bar, which was made from scratch as it is not a prebuilt LeafletJS feature that can be altered. To start with a searchbar variable was created which is of type 'control'. This is then positioned in the top right-hand side of the map. The next step was to add the HTML code elements which will display the searchbar itself. This is done through the use of "DomUtil.create" which can then be used to add some inner HTML code which will create the searchbar form. This is shown in the code extract below.

```
var searchbar = L.control({
    position: 'topright'
});
searchbar.onAdd = function (aonbmap) {
    var div = L.DomUtil.create('div', 'searchbar');
    div.innerHTML = '<input id="searchbar" type="text"
onkeyup="searchAONB()" placeholder="Search..">';
    return div;
}
searchbar.addTo(aonbmap);
```

Next a list of locations which are searchable are added, through the use of JSON. Here a list of locations is added, which have a name, latitude location and longitude location. Then whenever a user starts to search for a location, it will look through this list of places to find that location and add the marker to the main map. Below is an example of the JSON element which houses one of the locations that can be searched.

```
var markers = [
    {
        "name": "Arnside and Silverdale",
        "lat": 54.1708,
        "lng": -2.7995
    },...
```

The final step for the search bar is to create a function which is performed whenever a user is interacting with the search bar. This takes the string that the user has entered, makes it all uppercase so that it is not case sensitive when the user is searching. This means that no matter how they are written in the 'markers' variable, the user will be able to find them. The screenshot below shows the search bar working.

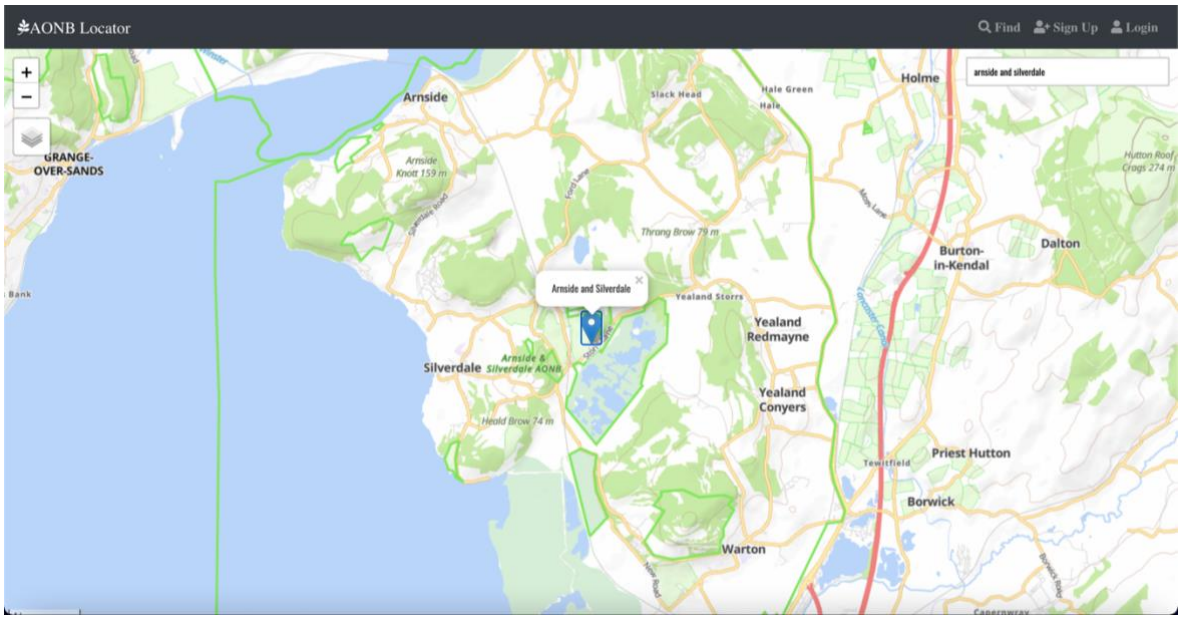

Figure 47. Screenshot showing how the screen will change when a user searched. The example being used is the user searching for 'arnside and silverdale'.

Other than the features of the main map, on the home page there is also a secondary navigation field which is in the form of a 'nav tab'. This is a selectable navigation link which alters the current screen what is being displayed. This is used to show the links to each of the different information webpages to the three nations: England, Wales and Northern Ireland.

8.4.3 Mobile Friendly UI

The whole web application has also been designed with mobile first development in mind. This means that should the site be visited on any device smaller than a 'traditional' computer monitor, such as a mobile phone or a tablet, then the web application will function the same on the smaller devices. This is achieved through the use of Bootstrap 4 [9] which allows for easier design and creation for mobile devices. The screenshot below shows the web application running on a mobile device using Google Chromes built in device running feature.

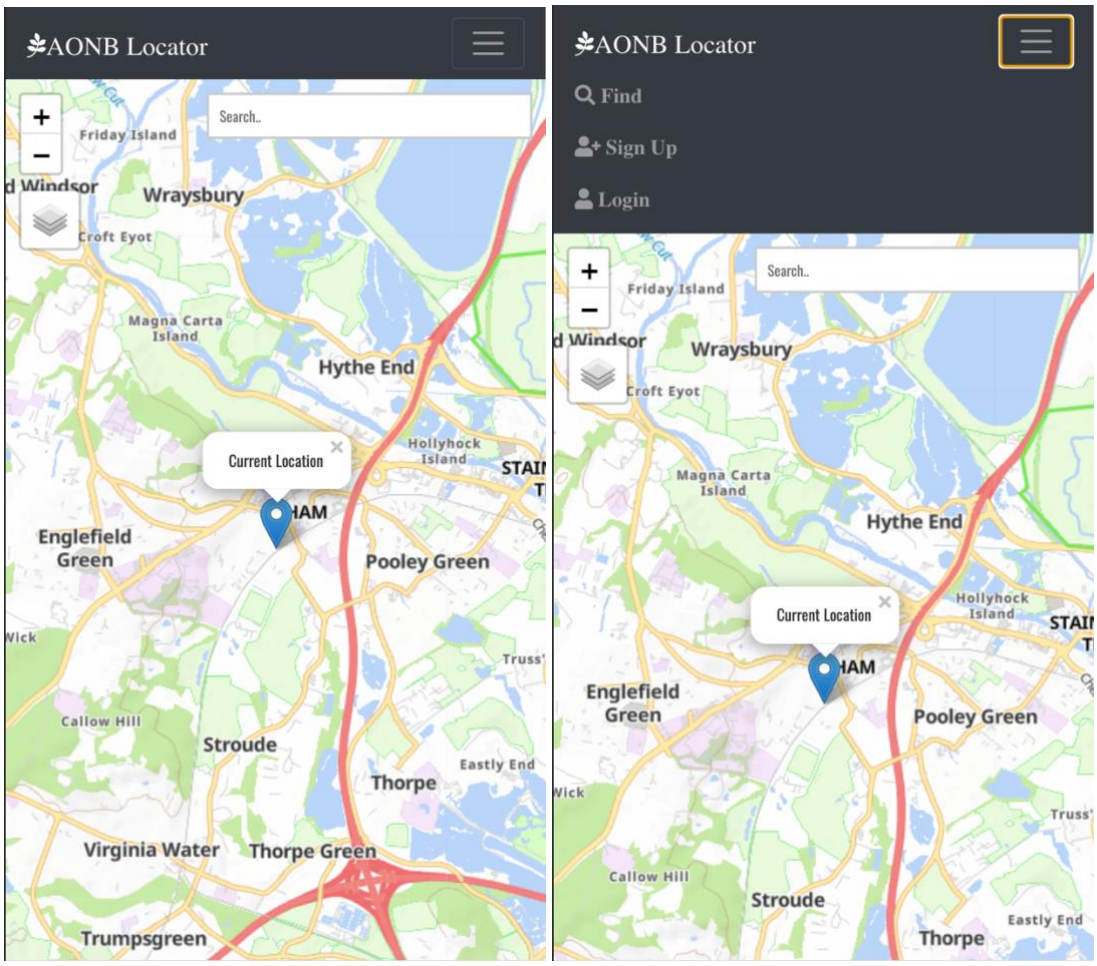

Figure 48. Screenshot showing the web application running on a mobile device using Google Chromes web developer tools.

By developing with mobile devices in mind, it means that the web application being created will be more accessible to a larger range of users. Also with the rise in smartphone usage by most individuals, which has seen an increase of around 48% increase in the number of households which have a mobile phone between the year 2000 and 2019 [42]. Another element to consider is the number of users which will be using these mobile devices to access the Internet and therefore the web application. A study conducted in 2016 found that 51.3% of Internet usage worldwide came from mobile devices compared to 48.7% for desktop devices [43]. Taking the trend found in their research, it would be best to consider the possibility that, on average, most people by now will be using a mobile device to view and use the web application. The study also found that in the last seven years (from 2013 to 2019) mobile traffic is up 222% [43].

8.4.4 User Accounts

The user accounts feature which was added would need some further enhancements to tie it into the project and application more. However, it was included in this iteration of the project as a foundation for future enhancements. As it stands, the user is able to register for an account where their details are securely stored in a MySQL database and with those details, they can then log into their account on the website. In the future, which will be discussed more in a following section, this would then change the functionality of the site more.

As it stands, a visitor has two options when it comes to the user accounts feature. They can either register for an account or login to an existing account. Both of which have their own links found within the navigation bar at the top of the screen or within the footer at the bottom of any webpage. On the user registration webpage, there is a required form that looks to capture a user's

name, email address and password (which they have to repeat to ensure they have entered the correct password and minimise errors in the future when attempting to login). This webpage can be seen below.

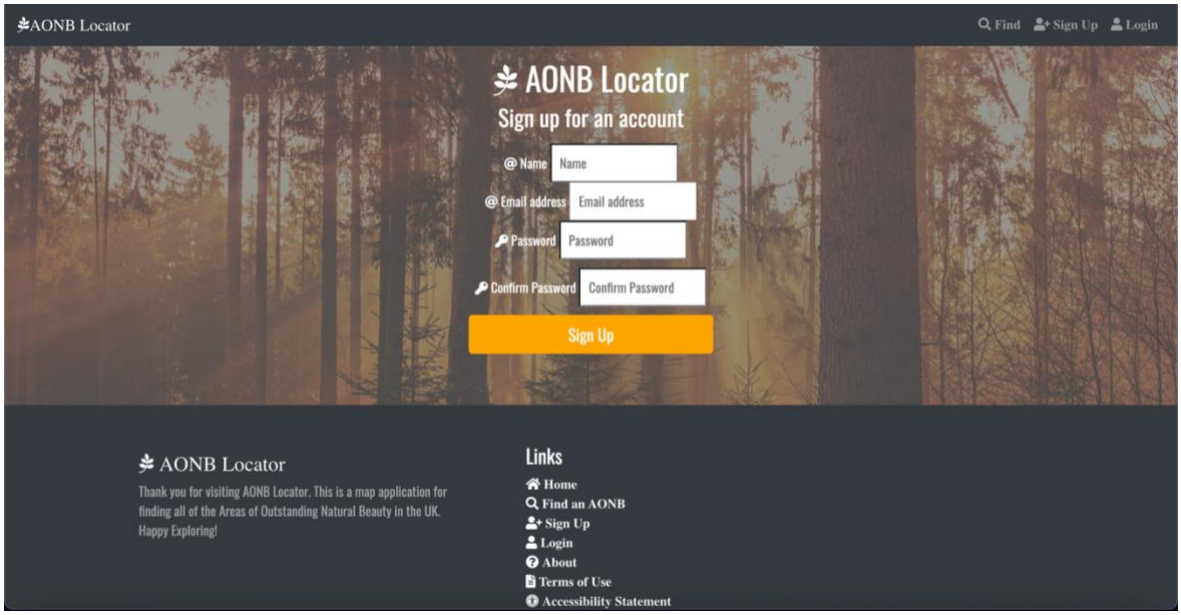

Figure 49. Screenshot showing the user registration page with the registration form for the user to fill out.

Once the user has inputted all of the information and hits the "Sign Up" button, the backend code then performs a number of checks on the information that has been entered. It does this first before establishing a connection with the database to make it quicker when 'rejecting' a user's sign-up request because of an error, such as the username or email address already being in the database or the passwords not matching. These checks are performed through a series of if-elseif-else statements which check for all possibilities that could occur. This includes the base cases of the field (such as email address) being left empty or the length of what has been entered being too short or too long. From this point, if everything has succeeded the connection to the database is then created. This is done by using the credentials within the database configuration file and creating a connection variable. Whilst creating this connection, another check is performed for any errors in connecting to the database. This is to ensure that there are no uncaught errors where the user may not realise that an error has occurred. Below is a screenshot example of an error message appearing to the user to give a visual cue that an error has occurred.

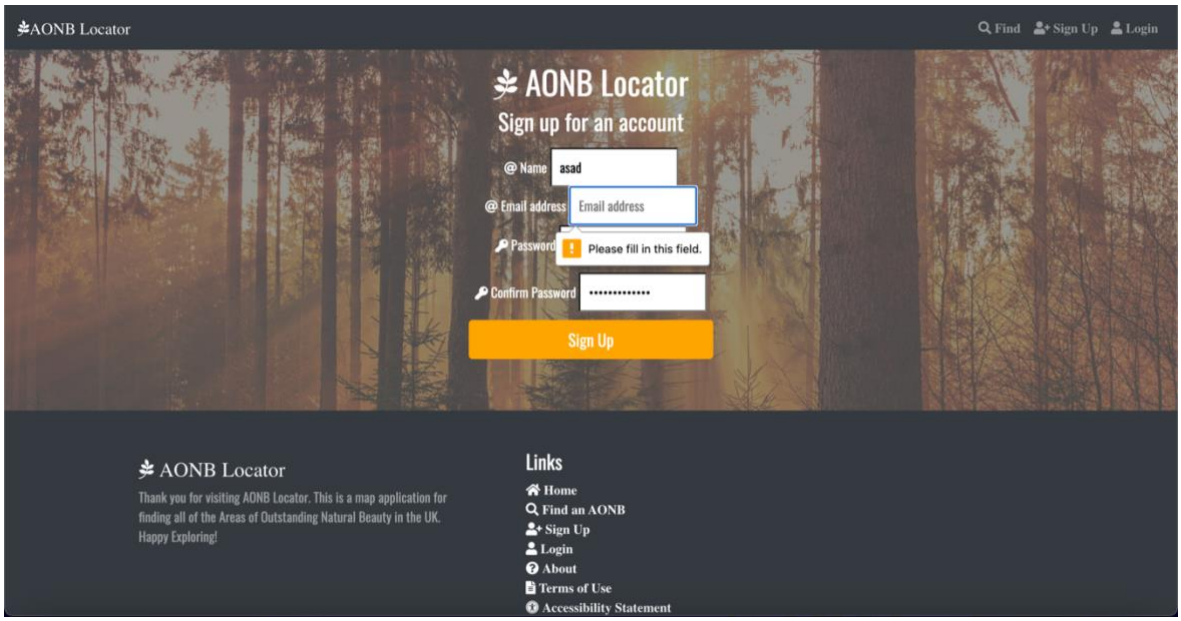

Figure 50. Screenshot showing an error message appearing when a field has been left empty by the user.

The way in which the check on if the user already an account has is done, along with the adding of the user to the MySQL database is through the use of SQL commands within the PHP code. For example, the following SQL command is used to check if the user is already a part of the user accounts table.

```
"SELECT * FROM users WHERE user_name = '" . $user_name . "' OR user_email
= '" . $user_email . "';";
```

In order to add the user to the database table, another SQL command can be used to insert all of the information that has been entered into the various fields of the form into the correlating column of the table. The command shown below will insert all of the information into the table correctly, pulling all of the required information from the form the user has filled out.

```
"INSERT INTO users (user_name, user_password_hash, user_email) VALUES('"
. $user_name . "', '" . $user_password_hash . "', '" . $user_email .
"');";
```

This performs the “INSERT INTO” SQL command, using the variables that store the users inputs (user_name, user_password_hash and user_email). Once a user has successfully entered all of the required information, they are then presented with a success message which shows that the account has been successfully created. This will then allow the user to use these details to login to their account. The screenshot below shows the success message displayed once the user has successfully signed up.

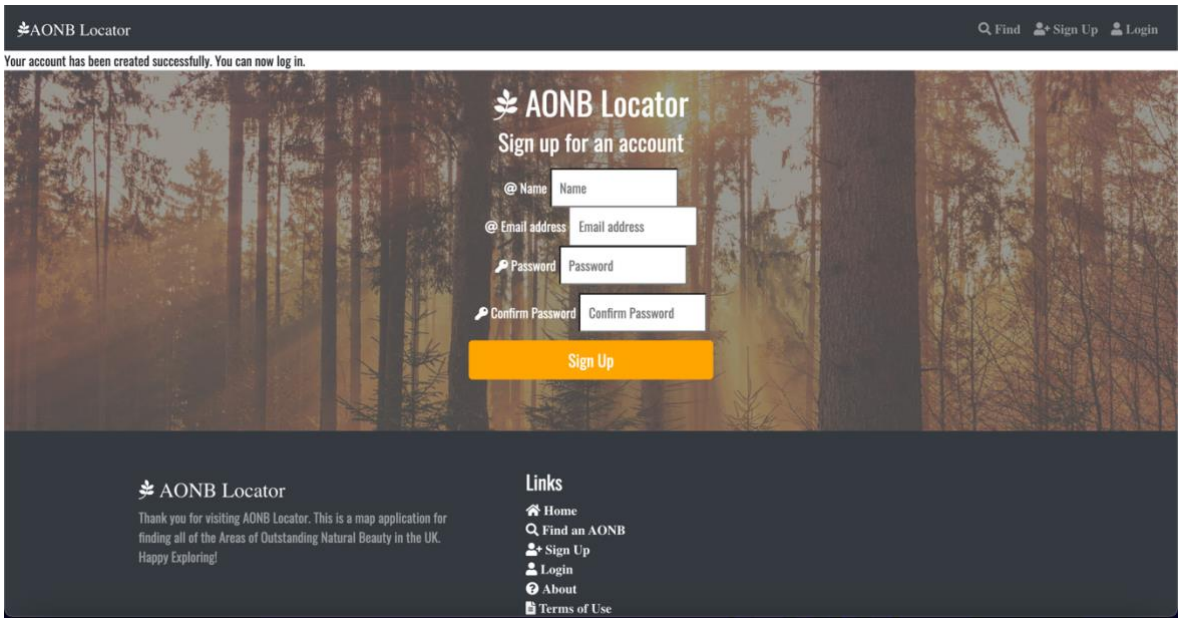

Figure 51. Screenshot showing the successfully signed up message to the user once they have signed up to the web application. Background image from unsplash [55].

Now that a visitor has registered for an account, they will then be able to log into that account. This again has its own webpage which houses the form they will need to fill out. This can be seen below.

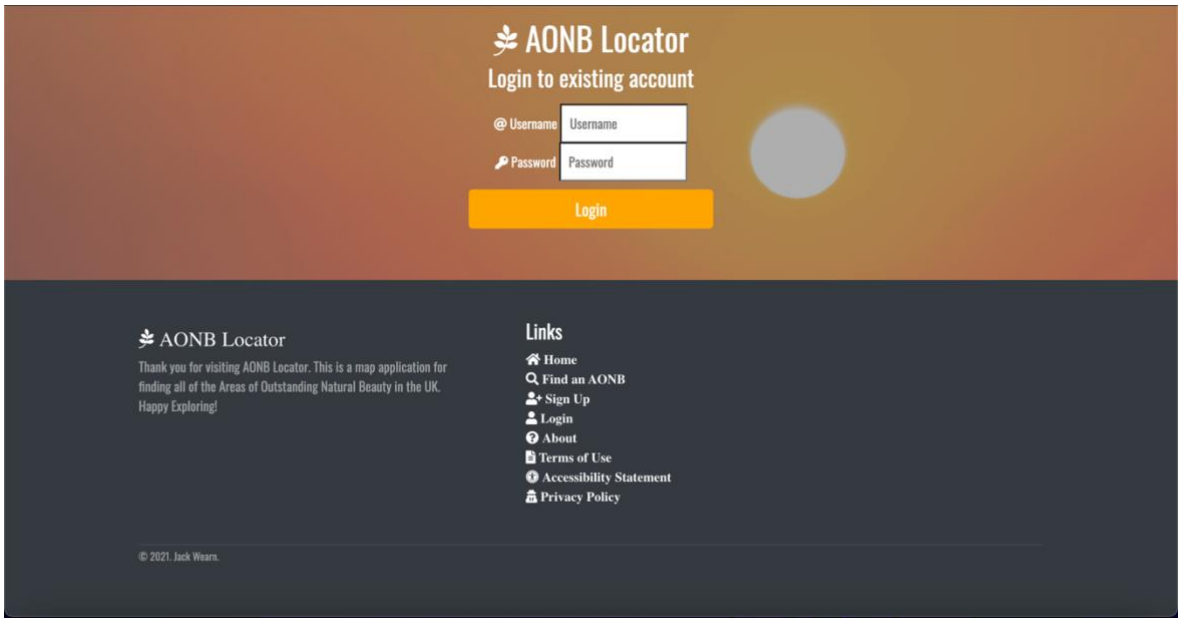

Figure 52. Screenshot showing the login webpage, which houses the form for the user to complete to login to their account. Background image supplied from unsplash [56].

Just like with the registration system, the login system also performs checks that all of the required fields have been completed and that they match what is stored on the database. For example, if the user does not complete the password field an error message will appear to prompt them to complete the that required field. This can be seen in the screenshot below.

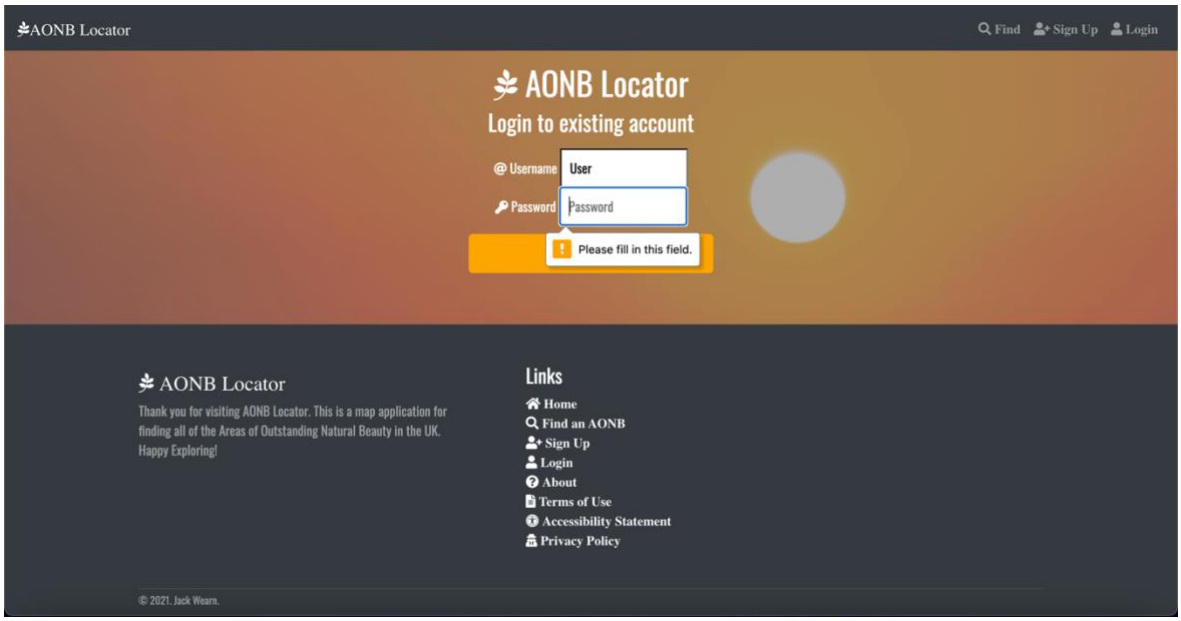

Figure 53. Screenshot showing the error message that a field has been left uncomplete by the user.

However, once the user has successfully completed the form with the correct details for their account, they will be logged in. The process of logging the user in follows some simple steps. It first performs a check that all of the fields have been completed. It then connects to the database, just like with the registration system. This is done with the below code.

```
$this->db_connection = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
```

It will then need to check if the details enter match what has been stored on the database for that user. This includes checking that the username or email address matches with the password that has been entered. This is done in the reverse of the registration in that it needs to check that the hashed passwords match, which involves un-hashing the stored password and checking it against the entered password. It firstly does this by using the SQL "SELECT" command based on the username, user email and user hashed password variables, with the following command.

```
"SELECT user_name, user_email, user_password_hash FROM users WHERE
user_name = '" . $user_name . "' OR user_email = '" . $user_name . "';";
```

If the username, email and password match what is stored for that user, then they are able to successfully be logged in. When this happens they are returned to the home page which now has a changed navigation bar in that it will now display their username in place of the sign up button and a logout link in place of the login link. This can be seen in the screenshot below.

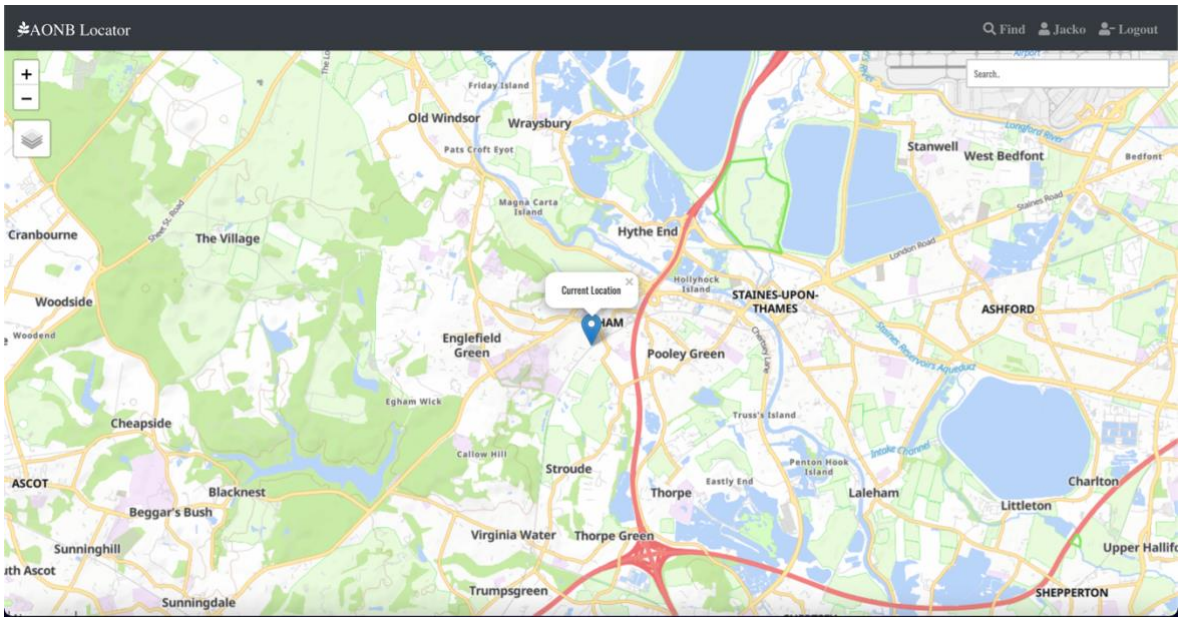

Figure 54. Screenshot showing the change in navigation bar once a user has been signed in.

The user will remain logged in regardless of what webpage they subsequently visit. For example, if they should visit an information page for an AONB, they will stay logged in until they choose to logout of their account. This is done through the use of sessions. A sessions is created when a user logs in and is only "destroyed" when they close the web application or logout explicitly using the logout link in the navigation bar. The screenshot below show the username still present on a different page from the home screen.

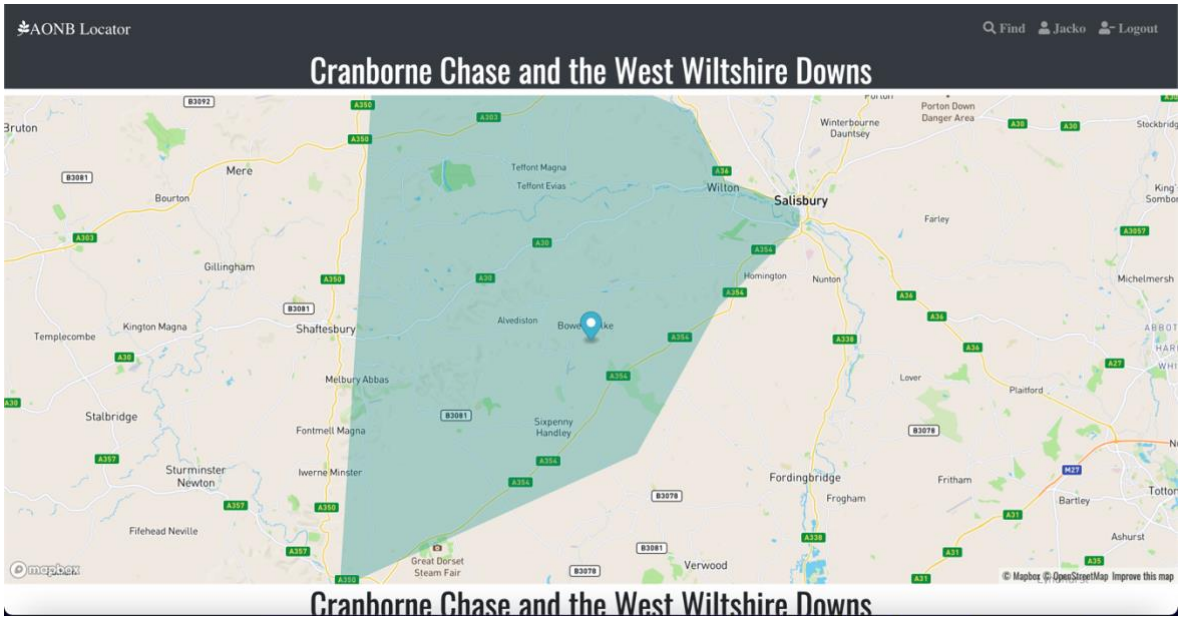

Figure 55. Screenshot showing that the user remains logged in when navigating the web application.

8.4.5 User Comments

An extended feature which was included in the application was the use of user comments on each of the different AONB information web pages. This was a space for the visitors to leave their own thoughts and feelings about the area and give hints of nice places to go and attractions to see whilst visiting the AONB location. In order to achieve this, there was a requirement for both frontend and

backend development. In terms of the backend, this involved creating a database which will store all of the comments that have been left by the user, stored based on a page ID. The screenshot below shows an example of some of the allocated page IDs, however the full table can be found in Appendix A.

Information Webpage Name	Corresponding Page ID for Database of Comments
Arnside & Silverdale	1
Blackdown Hills	2
Cannock Chase	3
Chichester Harbour	4
Chiltern Hills	5
Cornwall	6

Figure 56. Screenshot showing an example of the table which allocates a page ID to each of the different AONB locations [Appendix A]. The full table can be seen in Appendix A.

In terms of the creation of the comments section, each information webpage was created in the same way. There was a ' ' tag which had the class of comments, which will be used to display all of the comments and the add comment form. To make the comments system work, a connection to the database is first required. Once this connection has been secured, there are a number of important functions which are performed to ensure that the data being presented is in the best form. The first of which changes the current timestamp of the users comment into a time elapsed string. This means that should a user submit a comment, it will be displayed as 30 minutes ago, 1 day ago or 2 weeks ago. This is more of a cosmetic change but will make the user experience much better.

The next function is the show comment's function. This is used to display each of the individual comments and replies into the ' ' which was created on the HTML webpage. The way in which this works is through the use of creating JavaScript HTML tags, which will be used to display the collected data from the database. This data is collected via a SQL command:

```
'SELECT * FROM comments WHERE page_id = ? ORDER BY submit_date DESC'
```

Which gathers all of the data from the comments table within the database that has a matching page ID as the AONB information webpage. The screenshot below shows an example of one of these information pages displaying a comment which has been left by a visitor of the site.

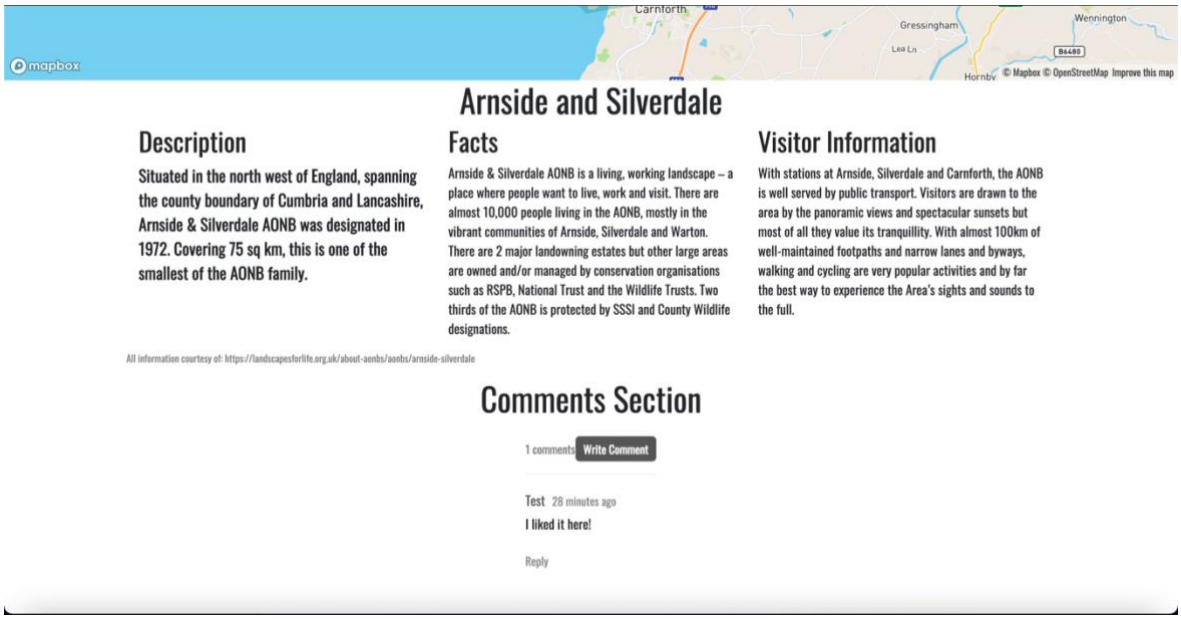

Figure 57. Screenshot showing a comment which has been left by a visitor of the Arnside and Silverdale information webpage.

8.4.6 Information Webpages for Each AONB Location

For each of the different AONB locations, an information webpage has been created which is used to show an outline on the map of where that AONB resides and some basic information about it. All information for this was generally gathered from the AONB site [44] itself. The importance of each of these is to give a better understanding of where the area is located and what a visitor could do there.

Each webpage is setup in the same exact way, it has the navigation bar at the top of the webpage, a static map from Mapbox [33] which uses GeoJSON to display a polygon shape to outline the area and a marker to show the location. Below this is the information itself and finally has the comments sections. This can all be seen in the screenshot below.

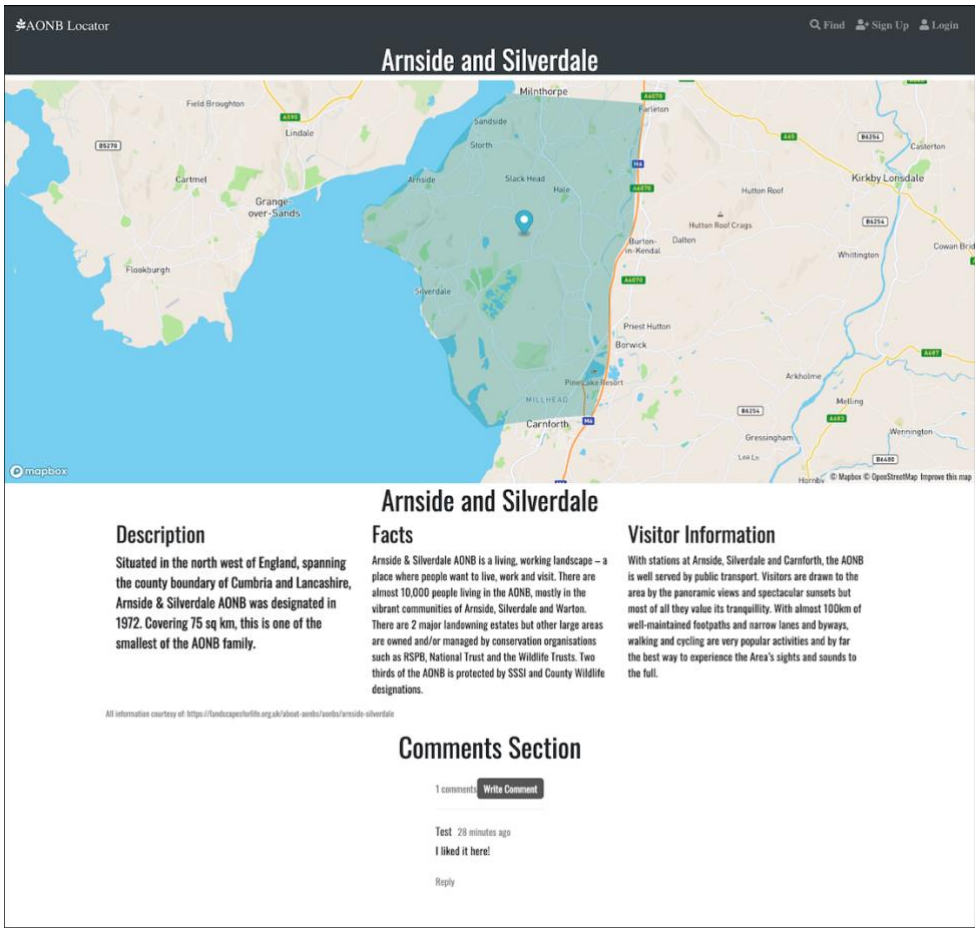

Figure 58. Screenshot showing the full Arnside and Silverdale webpage. This is uniform across each of the different information webpages.

The way in which the GeoJSON works is by creating a new map source, which has some properties. In this case, it has the properties of the feature polygon. The coordinates can then be listed which will make up this polygon. The site geojson.io [45] was very helpful in creating this much quicker as it allows you to draw the polygon on a map and retrieve all of the required coordinates at once. The code snippet below gives an example of this feature being implemented.

```
map.on('load', function() {
        map.addSource('arnside', {
            'type': 'geojson',
            'data': {
                'type': 'Feature',
                'geometry': {
                    'type': 'Polygon',
                    'coordinates': [
                        [
                            [
                                -2.8639984130859375,
                                54.183534825556876
                            ],
                            [
                                -2.8543853759765625,
                                54.18132476428218
                            ]
    ]
                    ]
                }
            }
```

```
        });
```

You can also change the colour and opacity as with any other object in HTML, CSS and JavaScript. For example, in the case of this web application each of the webpages have a fill colour of #088 (which is almost like a teal colour) and the opacity of 0.3. This can be seen in the code snippet below.

```
map.addLayer({
      'id': 'arnside',
      'type': 'fill',
      'source': 'arnside',
      'layout': {},
      'paint': {
            'fill-color': '#088',
            'fill-opacity': 0.3
      }
});
```

The output of this can all be seen on each of the different information webpages, such as in the example of the screenshot below.

Figure 59. Screenshot showing the output of the GeoJSON polygon and marker on the Arnside and Silverdale information webpage.

8.5 Issues During the Creation of the End Product

A number of issues were encountered during the course of creating the project, including problems in the proof-of-concept development, term one demo application and the final product. This section of the chapter will discuss these issues that did arise during the creation of the end product.

The first issue that was encountered was creating the database for the comments system. At first, it looked like each webpage would not be able to have its own comments section due to it not recognising that it was a different page. The way in which this issue was overcome, at first, was to move the comments section to its own webpage which would work more as a forum. However, after discussions with the project supervisor a new plan was put into action which saw an alteration to both the database and each of the different webpages. This was to create a new column which would store a page ID, which was to correspond with the different information webpage. This did in fact fix the issue that was being had, however in the future should a large number of people leave

comments, may encounter a new issue that all comments are stored in the one table and therefore could decrease performance in displaying the comments.

Another issue that was encountered in the creation of this final product is that the MySQL database on the localhost which was being used for testing the site would not work and could not be edited. It was found through further research into the issue that it was a change in command to open the MySQL command on Mac, which was to use the following command: "/usr/local/mysql/bin/mysql -u root -p".

Overall, through the use of creating the proof-of-concept programs a lot of errors were caught earlier or plans were changed accordingly to avoid these issues arising in the final product. From the knowledge gained from the creation of these programs, the development of the end product was much smoother than if they had not been created.

8.6 Running the Web Application

In order to run a web application, it must be opened within a browser such as Google Chrome or Firefox. However, as this project has been created with the use of PHP code for the main functionality of the web application, a server must be present in order for the code to be able to output what it is trying display. This can be achieved by running the application through the Localhost server. This is where the application is stored on the developer's machine and a local server is setup, allowing for the PHP code to be able to run. You are then able to open any browser and navigate to the localhost directory within the browser and the application should run. However, this project also has an extra constraint in that not all browsers support service workers. Although most are looking to adapt to the newer technology, some do not currently facilitate service workers. The best option for running the project is Google Chrome.

The application can also be hosted through a server, such as Heroku. This would then create a public URL which anyone could visit and use the web application. This would also facilitate a location for the MySQL database to be stored/hosted. This would remove the need for anyone that wished to run the application having to create their own database on their local machine, as is the case when running through the Localhost.

In the case of this project, the code has been hosted via Heroku. The link for this will work from on the submission date which is the 12th April 2021 and can be accessed via the link below []54:

https://aonblocator-final-year-project.herokuapp.com/index.php

8.7 Work Log

In order to ensure that all tasks were completed and that the project run as smooth as possible, a project diary was used. Within this diary, a weekly log of tasks was spoken about and any issues that may have arisen throughout the course of the project were noted. This allowed for a better understanding between everyone involved as to how the project was progressing. It also gave an opportunity to reflect on the speed in which the project was progressing. The table below will outline the initial plan for tasks, when they were planned to be completed, the actual time taken to complete and a breakdown of reasons why this may have been the case.

Figure 60. Table outlining the work log and project progression from start to finish.

Task set out during project planning phase	Estimated Completion Dates	Actual Completion Dates	Breakdown of the task.
Create basic web development proof of concept program and corresponding report.	28/09/2020 - 30/09/2020	02/10/2020	The creation of the proof-of-concept program took less time than what was expected, but the report took slightly longer to create.
Create a proof-of-concept program for OpenStreetMap and Leaflet JS, along with the corresponding report	29/09/2020 - 02/10/2020	09/10/2020	This task took slightly longer than first anticipated as learning about OSM data and LeafletJS took longer. The report also took longer to write, but from what was learnt it was easier.
Setup GitHub ready for use for the project files	01/10/2020	09/10/2020	This task was started later than first anticipated, but still took just the one day to complete.
Begin the creation of the map application	05/10/2020 - 06/10/2020	12/10/2020	This was actually started in the second term. However, the demo application file was started.
Begin to learn what is required for a webpage to be able to be viewed offline and how to make this work with OSM/Leaflet JS	07/10/2020 - 14/10/2020	16/10/2020	The research of this did take roughly as long as planned, however was started slightly later.
Create a report on offline webpages	09/10/2020 - 16/10/2020	02/11/2020	This took longer than had planned, as the offline aspect of the project was harder to grasp personally. However, this was completed in time for the interim review.
Begin to implement the required OSM/Leaflet JS data into the project files	19/10/2020 - 21/10/2020	20/10/2020	This was completed before the report for the offline webpages, as it gave a needed break from trying to learn what was needed to implement the offline application.
Continue to work on the project and learning more advanced Leaflet JS to style the map, markers, etc...	22/10/2020 - 30/10/2020	30/10/2020	Completed a number of tutorials which gave a better understanding of LeafletJS and implemented these features.
Gather map Latitude & Longitude points for the different points of interest and set them up in a file for use (potentially GeoJSON) +	28/10/2020 - 09/11/2020	19/03/2021	This ended up being performed in the second term. This is because the few points of interest were hard coded for the demo application. The report was also not needed and just spoken about

(Create a report of GeoJSON if used)			within this final report.
Research and implement (into project/proof of concept program) how to create layered maps (user can change look of map for road networks, outdoor trails, etc....)	05/11/2020 - 13/11/2020	30/10/2020	This was learnt about while learning new LeafletJS technologies and tools.
Convert file of points of interest locations into markers, boundaries, etc… into project	16/11/2020 – 18/11/2020	19/03/2021	Again, this was implemented in the second term, and so was not completed in the original timeline set out.
Create a report on Dijkstra's algorithm, explaining how it works and how it could be used for navigation	19/11/2020 - 25/11/2020	N/A	This was not implemented and therefore a report was not created for this project. This is because a navigation system was not created for the project. However, some research was conducted as it could have been a potential inclusion should there have been enough time.
Create proof of concept programs and report for route planning	24/11/2020 - 30/11/2020		
Begin to implement the route planning aspects to the main project	01/12/2020 - 07/12/2020		
Create a proof-of-concept program for offline web pages, implementing different ways	01/12/2020 - 08/12/2020	11/12/2020	This took longer to complete as getting to grips with the different offline technologies took longer than expected.
Work on the project to get it to work offline (basic starting of it)	02/12/2020 - 11/12/2020	11/12/2020	This was completed alongside the proof-of-concept program so was also able to be implemented into the demo application.
Continue creating actual projects map application – initial features (displaying the map, adding some markers and locations)	11/01/2021 - 18/01/2021	18/01/20201	Once coming back from a Christmas break, the end product was began working on. This included taking what was created for the demo application and making changes.
Begin implementing more locations and other ways of displaying the data on the map application	15/01/2021 - 25/01/2021	12/02/2021	This took longer than first anticipated as the way in which the application was going to work was changed so that it all ran through PHP code rather than just HTML to accommodate for the user accounts system and

			comments system.
Implement some extended features (including user login/sign up)	25/01/2021 - 08/02/2021	12/03/2021	This took much longer than first anticipated, as the two systems did not like to both be working together and connecting to the database was causing a number of issues, which can be seen from the project diary.
Continue working on the map application, fixing bugs and starting to create the final report based on the proof-of-concept reports	08/02/2021 - 26/03/2021	26/03/2021	Work was continuously worked on up until the deadline for the project which was the 26/03/2021.

8.8 Potential Future Enhancements

There were a variety of features which were included in the end product of this web application. However, there are also a number of potential enhancements that could be made in the future or if the project was to be 'restarted' that could be included to better enhance the web application. The first of which would be to change the user comments systems to link up better with the login system so that only a registered user could leave a comment. This would be a worthwhile addition to this project as it would minimise a lot of risk surrounding who is able to comment and who they can communicate with, as each user commenting would have had to supply more details than just their name and their comment.

Another feature that could be included as a future enhancement to the project would be to create a user profile page. This would take the web application further into a more 'social media' category of website, however it would be useful in allowing the registered user to save locations that they like, have visited or wish to visit in the future. It could also allow for people to connect with others who live locally or would wish to find someone to explore these areas together. Again, this would have a number of professional risks that would need to be considered but could be part of a consideration for future enhancements to the program.

In the future, another consideration that would be made is the fact that each individual location information page is separate of themselves. This would mean that should the design change again in the future each of these pages would need to be separately changed, which would take a long time to do. So. a future enhancement would be to make a template page, just like with the header and footer PHP files which would then change dependant on the location or area. This would mean that any design changes would only need to be done once rather than 46 times for each AONB location.

A final enhancement that could be considered would be a navigation system. This could be just a walking navigation system which would show the user the best walking trails or routes to get the best views. This could again be extended further to allow for registered users to add their own routes or points of interest that others may find interesting. Although a complete program was developed, there are a number of additions that could be made should the project be extended further or if it were to be restarted completely from scratch.

Chapter 9: **Software Engineering**

9.1 Methodology

The software engineering methodology which was adopted within the creation of this project was the agile development methodology. The reason for this is because the deadlines that were set out matched up with the agile process model. In the agile approach, a plan is first created, then the product is developed and worked on and finally delivered (either to end users or stakeholders). Feedback is then received, and any changes are made in the collaboration stage (stage where the project is worked on). The diagram below from the Synopsys Editorial Team [46] illustrates this process model.

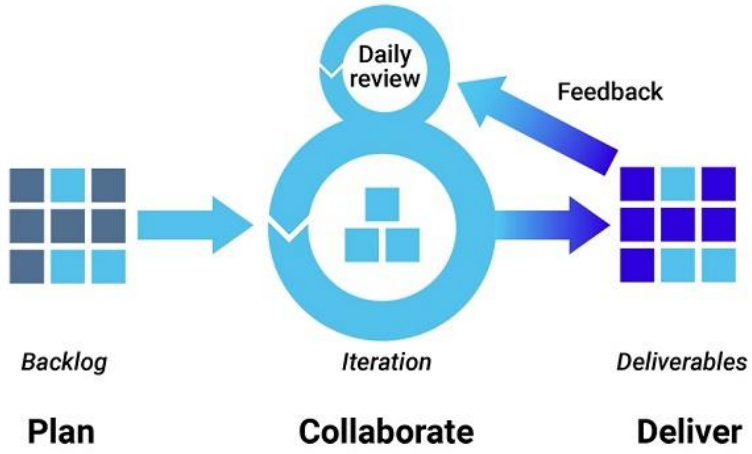

Figure 61. Example diagram of an agile development approach. Image courtesy of Synopsys Editorial Team [46].

For this project, the project plan was the first stage (i.e., the planning phase). From this, the proof-of-concept programs, prototype/demo program and final product were created. This is the collaborate phase. These were then delivered to the project supervisor for feedback, either during the course of the first term or at the end via the end of term review. Feedback was then given, and the required changes were made. This flow worked very well during the course of completing this project, as it allowed for good feedback on what had been created and changes adopted early on. It also allowed some freedom on getting feedback and implementing the required changes.

9.2 Version Control

During the course of this project, a GitHub repository [47] has been used. This has been used to store all files related to the project which includes all of the reports, proof-of-concept programs, the demo application and the end product. The reason why version control is so important to use is because it gives a history of what has been completed. This is not only helpful in recognising times when not much work was completed, but also allows for the reversal of files. For example, should some aspect of the application stop functioning as intended, you are able to 'roll back' the code to a working version and start again to implement the required changes. It also allowed the project to be worked on at any location on any device. All that was required was the GitHub repository be 'pulled' onto the device being used and then any changes 'pushed' back onto the repository once finished. Version control proved very helpful when, in the first term of the project, the main computer system that was being used to create the project was accidentally destroyed beyond

repair. Thankfully, once the new device was obtained no code was lost as it was all stored within the repository which meant that the pull request was performed on the new machine and the project could swiftly continue.

9.3 Testing

In order to ensure that all the tests have taken place, it is best to create a test suite which will ensure that all elements of the project are thoroughly tested. When testing software, there are two main types of testing: functional and non-functional testing [48]. Functional testing encapsulates a number of different tests, which each serve their own purpose on ensuring the final product works as it is intended. This includes unit, integration, end-to-end, acceptance and interface testing (to name a few). Non-functional testing encapsulates all of the tests which focus on the performance, reliability, usability and security of the software [48]. This includes performance, security and stress testing. There is also testing which falls more into the categories of UI design testing, which includes tests such as A/B testing.

For the purposes of this project, and due to time constraints, the testing which was performed for this project included security testing (for the user login and registration), UI testing, performance testing and black box testing.

The security testing involved checking that each of the different forms, which were connected to the user registration and login database were correctly collecting and storing the data that had been inputted. It also needed to check that SQL code could not be submitted into any of the fields and therefore reduce the likelihood of an SQL injection into the database. The testing also was used to check that a user that already has an account could not sign up again with the exact same credentials. The tests involved attempting to "break" the user registration and login system. The results of these test concluded that, for the most part, the system was very secure. However, the testing could have been more extensive to ensure that all kinds of database style attacks were accounted for, which would lower the risk of any potentially important information being made available to anyone.

The UI testing was done through the use of the prototype/demo application. This was created based off the first iteration of prototype designs. The demo was then delivered to a group of people who, through the use of questions and details of potential changes, helped to identify potential UI faults. This included the fact that the map did not seem to be the prominent feature of the application. The feedback received was then used in the creation of the final product and therefore the UI was adapted to accommodate for the suggested changes based on the feedback. The result of the testing can be seen in the differences in the design of the user interface from the prototype/demo application and the final delivered product.

Performance testing involved checking how well the web application worked both online and offline. The Google Chrome web browser made this much easier to complete, as within the developer's tools window, they have the option to make the browser see the user as 'offline'. This meant that during the testing of the functionality and therefore performance, any clear differences could be easily identified and adjusted for. For example, the data that was being cached from the map initially meant that the user would only see the active tiles when they go offline. However, with some changes to the way in which the service worker cached the data, more of the map was then made available offline.

Black box testing involves testing the application as a whole. This means that every feature of the application should be tested as if it is the end user going through the flow of using the web application. To do this, the testing started with opening the application, browsing the map and creating an account. The user would then sign into that account, leave a comment within the 'forum' and return to the main map. They would then 'lose connection' and continue to use the

application, checking that they could sign out and back in again and go to any of the other web pages. The test concluded that the application functioned as intended.

Overall, there was more testing that could have been conducted to ensure that the application was fully functioning and that all aspects were covered. However, with the testing that has been performed, it can be said with confidence that the offline map application is able to meet all of the criteria of the project specification, aims and objectives.

9.4 Documentation

Within this portion of the report, the documentation will be produced to give a better understanding of the project and how it is run. This will give a brief but better understanding of the project. Firstly, should the link to the Heroku site [54] **`https://aonblocator-final-year-project.herokuapp.com/index.php`** which is running this application not work or not yet be available the project can be run through the localhost of the machine being used.

In order for the application to run, because of the use of PHP, it will need to be run via the Localhost. There are a number of ways to do this dependant on the Operating Systems (OS) being used. Personally, this was all done using MacOS which has an easy way of setting up the Localhost. However, this portion will briefly explain how to do it on a number of different OS with links that may give more details on this. For MacOS, you would firstly need to create a new folder (if you haven't already) named `Sites` and should be within the user section of finder (the part which says the name of the user). You would then need to turn on the Apache server. This is done in the terminal using the following command: `sudo apachectl start`. For this project, PHP will also need to be turned on. This is done through the terminal once again using the following command: `sudo nano /etc/apache2/httpd.conf`. Once this has opened, you will need to search for `php` using `CTRL+W`. Then all that is required is to remove the hashtag from the following: `#LoadModule php7_module libexec/apache2/libphp7.so`. All of the code from this project can then be added to the `Sites` folder and navigated to on Google Chrome with the path: `Localhost/AONBLocator/index.php`. A good site that outlines this in more details is by Website Beaver [49].

To do this on Windows there are four steps that can be followed, which come from a tutorial by Jim Cambell at Techwalla [50]. The steps show that the localhost can be setup by first opening control panel and navigating to the programs link. You should then be able to "Turn Windows Features On or Off" and check the box which is labelled as "Internet Information Services". Once the computer is reboot you should then have the localhost running. Another tool would be needed which is either XAMPP or WAMP, which allows for the configuration of the localhost server. Once this has been followed, you should be able to navigate to the site using the following URL: `127.0.0.1 /NameOfSite/Index.php`. The steps for setting up the MySQL database would then need to be followed. Another great tutorial for this is from Pankaj Sood at Code Briefly [51]. Just like with Windows, a great tutorial on how to setup the required stacks for running the web application on Linux can be found Mark Drake at Digital Ocean [52], which go through the steps of setting up apache, PHP and MySQL for web development.

This project also requires a MySQL database be created. Again Website Beaver [49] outlines how to do this for MacOS, in which case you will need MySQL downloaded onto the machine and you can use the following code to get into the terminal MySQL window: `sudo /usr/local/mysql/bin/mysql -u root -p`. You will then need to create the Database and the tables required for this project. These are outline within the above sections, but the following code will create the database and the two required tables:

```
mysql> CREATE DATABASE finalproduct;
mysql> CREATE TABLE IF NOT EXISTS `comments` (
```

```
    -> `id` int(11) NOT NULL AUTO_INCREMENT,
    -> `page_id` int(11) NOT NULL,
    -> `parent_id` int(11) NOT NULL DEFAULT '-1',
    -> `name` varchar(255) NOT NULL,
    -> `content` text NOT NULL,
    -> `submit_date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
    -> PRIMARY KEY (`id`)
    -> ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;
mysql> CREATE TABLE IF NOT EXISTS `users` (
    -> `user_id` int(11) NOT NULL AUTO_INCREMENT,
    -> `user_name` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
    ->  `user_password_hash`  varchar(255)  COLLATE  utf8_unicode_ci  NOT
NULL,
    -> `user_email` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
    -> PRIMARY KEY (`user_id`),
    -> UNIQUE KEY `user_name` (`user_name`),
    -> UNIQUE KEY `user_email` (`user_email`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

Now all that is required is to navigate on Google Chrome to the Localhost where the files are stored and you should have a fully working version of the web application.

Chapter 10: **Assessment**

10.1 Self-Evaluation

The work produced during the first term of this project overall went well. It was a productive period of the project with a lot of new information and knowledge gained which proved useful in the creation of the final product during term two. One thing to take from completing the first terms iteration of this project is the timings of different tasks. With some tasks taking longer than initially planned, it was best to use this when planning for what was planned to be completed in the second term. This meant having some fallback time for some more advanced tasks and allowing for some elements of the project to overrun without putting the project behind schedule for the delivery date. This was then taken into account when deciding on what needs to be completed and what other features may be able to be included or excluded from the final deliverable. It was planned that at the start of the second term, some of the other features which were not explored during the first term was to be further developed and learnt about in much more detail. This was the best move, as the first term was spent ensuring that all of the basic knowledge needed to make a fully-fledged map application.

The second term of the project was a difficult one, to balance the learning of some more advanced technologies with building the final deliverable. However, in general the term went well and the work that was completed was done so in as efficient of a way as possible. The project diary could have been updated more regularly, however with managing time around the different tasks of this project the diary did fall out of time for some weeks during the busier periods of work. The work that was completed, including the proof-of-concept programs, the final product and this final project report, was all worked on throughout the duration of the project.

Overall, the project has gone very well, and the work completed was extended further than what was initially thought of as being completable. If this project was to be completed again some changes to timeframes and project requirements would have been changed, but overall, the project went very well.

Chapter 11: **Professional Issues**

No matter the type or magnitude of the project being undertaken, there are a number of professional considerations that should be made to minimise risks. This is the case whether it is a software related project or not. Within this chapter of the report, these professional issues will be explored in the context of this project and how they may have effect the "wider-world".

This project was a web application, which means that it is a website that is deployed onto the Internet for anyone to visit. This in itself creates an issue of who will be viewing the application. In this sense, for this project it is aimed at anyone of any age range. An element that would need to be considered in this case would be that all of the information that is displayed on the site would need to be appropriate for people of any age and background, i.e., not using any inappropriate language or phrases. It would also need to be accessible by anyone, so should work on any form of device. Most websites have some form of age rating, such as a website selling products only for use by people over 18 for example. In this case, the site should have some form of check that authorises the user to enter the site. Although not needed on this application, it should be considered for certain web applications. One form of check that is deployed by a number of web applications is having the user enter their date of birth before entering which minimises the risk of a user being too young to enter the site. However, another issue does arise from this in that it is very easy to "fake", which bypasses the age filter. Because of this, a number of sites now employ new ways of confirming a user's age, such as at the purchase stage where they are required to supply proof of age.

There were also a number of risks which were identified within the project planning phase of this project. The first of which is damage to any of the hardware or accidental removal of any required software. In the wider world, there are usually large project teams that are all working on the project at one time. Natural disaster could be a cause of loss of hardware, such as a server being destroyed by a fire or flood. This risk is usually minimised by having external backups which are stored off site, meaning that if anything should happen on the main premises there would be a backup of the project in another location which could be used by the project team. In the sense of this project, the GitHub repository was used as an external backup of all work completed and did prove helpful when the main machine that was being used to create the project was damaged beyond repair. The project could still be worked on from another machine without any loss of work completed.

Another professional risk that should be considered is introduced when creating features such as the user accounts system. By allowing a visitor to create an account, the website creator then becomes responsible for ensuring that the users data they have given is kept secure. A way of minimising this risk is by ensuring that all information that is kept about a user is stored securely on the server and the database and that any passwords for example, are encrypted to minimise the internal risk of someone who has access to the database seeing their full login details and also minimise the risk of should someone gain access to the database that they will not be able to simply see the plaintext of the users password. For this project, this risk was taken into consideration by using a password hashing API [39] which ensures that no plaintext passwords are stored on the MySQL database.

Another feature which was introduced within this project was the user comments. This is a place for users to write their own comments about the AONB locations and interact with other users which may share a similar interest. However, when creating software with a social element, there are a number of professional considerations that must be taken into account. The first being the idea around who is able to leave a comment. For most sites, this is done via an accounts system, which only allows for comments to be left if the user is logged in. The risk of not doing this is that anyone would be able to leave a message and not be tracked as to who had left it much easier. This raises the concerns of cyber bullying taking place and harassing behaviour going unidentifiable.

This is the reason why many modern-day pieces of software now require the user to have an account the leave a message. However, this is not always a perfect system as when the technology evolved to attempt to deter people from leaving potentially hateful comments, so too did the attackers. They are able to create a fake account with an email and credentials which is not their own.

Another consideration that must be taken into place when creating a feature such as the comments system is who will be communicating. If the system is open to people of all ages, there should be some thought into who can have an account and therefore who can leave a message. The risk associated with this is that there is a risk to anyone who is younger communicating with any potential threats from older users. Threats such as this have been looked into from a number of software creators, such as Facebook [53] which requires a user to be at least 13 years old and there are security features that can be implemented by the user to ensure that not anyone can comment on any post or message them without being a 'friend' on the system or confirming that they are able to communicate.

In terms of this project, the risks surrounding the comments section were taken into account however without the required time were not mitigated. This means that any visitor can leave a comment onto the web application. The risk here is that anyone can speak to anyone, no matter their age or background or whether they have an account or not. This removes more of the element of risk of leaving a mean or offensive comment onto the system, which is not ideal for a web application open for anyone. In the future this would need to be examined much more closely to minimise any risks associated with the user comments section.

Conclusion

As a conclusion to this report, and to this project, this portion will discuss the project as a whole. From the start, a large number of new technologies and tools have been used which has given a great insight into the creation of offline web application using new HTML5 standards. Personally, a number of these programming languages and technologies had not been used previously which was exciting to delve deeper into them and gain a greater understanding of how these are used in the creation of offline web application. From the very beginning, professionalism was needed to ensure that the project was a success, which in personal opinion was achieved. With the upkeep on ensuring that the project was worked on regularly and communication with the project supervisor, the project run smoothly.

Bibliography

[1] "Using service workers", [Online]. Accessed: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers [Date Accessed: 27/11/2020].

[2] "Using the application cache", [Online]. Accessed: https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache [Date Accessed: 27/11/2020].

[3] "Online web tutorials", [Online]. Accessed: https://www.w3schools.com/ [Date Accessed: 27/11/2020].

[4] "Project Diary", [Online]. Accessed: https://pd.cs.rhul.ac.uk/2020-21/ [Date Accessed: 29/11/2020].

[5] "HTML, Living Standard", [Online]. Accessed: https://html.spec.whatwg.org/multipage/ [Date Accessed: 02/10/2020].

[6] Lie, H.W. and Bos, B., 2005. *Cascading style sheets: Designing for the web, Portable Documents*. Addison-Wesley Professional.

[7] Seibel, P., 2009. *Coders at work: Reflections on the craft of programming*. Apress.

[8] "Bootstrap (front-end framework)", [Online]. Accessed: https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework) [Date Accessed: 30/11/2020].

[9] "Get Bootstrap", [Online]. Accessed: https://getbootstrap.com [Date Accessed: 30/11/2020].

[10] "jQuery", [Online]. Accessed: https://en.wikipedia.org/wiki/JQuery [Date Accessed: 30/11/2020].

[11] "Content Delivery Network", [Online]. Accessed: https://en.wikipedia.org/wiki/Content_delivery_network [Date Accessed: 30/11/2020].

[12] "jQuery Syntax", [Online]. Accessed: https://www.w3schools.com/jquery/jquery_syntax.asp [Date Accessed: 30/11/2020].

[13] "jQuery Click() Event", [Online]. Accessed: https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_click [Date Accessed: 30/11/2020].

[14] "Canvas Element", [Online]. Accessed: https://en.wikipedia.org/wiki/Canvas_element [Date Accessed: 30/11/2020].

[15] "HTML Canvas, Draw a Circle", [Online]. Accessed: https://www.w3schools.com/html/html5_canvas.asp [Date Accessed: 30/11/2020].

[16] "Caching ", [Online]. Accessed: https://www.businessinsider.com/what-is-cache?r=US&IR=T [Date Accessed: 24/03/2021]

[17] "Service Workers API", [Online]. Accessed: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API [Date Accessed: 30/11/2020]

[18] "Service Workers Tutorial", [Online]. Accessed: https://developers.google.com/web/fundamentals/primers/service-workers [Date Accessed: 24/03/2021]

[19] "Figma", [Online]. Accessed: https://www.figma.com [Date Accessed: 24/03/2021]

[20] "MockFlow, UI Designer", [Online]. Accessed: https://www.mockflow.com [Date Accessed: 23/11/2020].

[21] "HTML forms tutorial", [Online]. Accessed: https://www.w3schools.com/html/html_forms.asp [Date Accessed: 24/03/2021]

[22] "Frontend Vs. Backend Development", [Online]. Accessed: https://www.coursereport.com/blog/front-end-development-vs-back-end-development-where-to-start [Date Accessed: 24/03/2021]
[23] "JavaScript Frontend or Backend", [Online]. Accessed: https://careerkarma.com/blog/javascript-front-end-or-back-end/ [Date Accessed: 24/03/2021]
[24] "Connecting Frontend and Backend", [Online]. Accessed: https://stackoverflow.com/questions/45413340/how-to-connect-frontend-and-backend-correctly [Date Accessed: 24/03/2021]
[25] "Relational Database Management System", [Online]. Accessed: https://techterms.com/definition/rdbms [Date Accessed: 24/03/2021]
[26] "Open Street Map", [Online]. Accessed: https://www.openstreetmap.org/about [Date Accessed: 30/11/2020].
[27] "What is Tile and Differentiate between Raster Tile and Vector Tile", [Online]. Accessed: https://bachasoftware.com/what-is-tile-and-differentiate-between-raster-tile-and-vector-tile/ [Date Accessed: 30/11/2020].
[28] "Vector Tiles", [Online]. Accessed: https://wiki.openstreetmap.org/wiki/Vector_tiles [Date Accessed: 30/11/2020].
[29] "Raster Vs. Vector: Pros and Cons of both Map Tile Types", [Online]. Accessed: https://www.geoapify.com/raster-vs-vector-map-tiles [Date Accessed: 30/11/2020].
[30] "What is a Vector Tile", [Online]. Accessed: https://www.maptiler.com/news/2019/02/what-are-vector-tiles-and-why-you-should-care/ [Date Accessed: 24/03/2021]
[31] "What are Vector Tiles", [Online]. Accessed: https://www.maptiler.com/news/2019/02/what-are-vector-tiles-and-why-you-should-care/ [Date Accessed: 24/03/2021]
[32] "Leaflet, An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps", [Online]. Accessed: https://leafletjs.com [Date Accessed: 30/11/2020].
[33] "Mapbox, The Building Blocks", [Online]. Accessed: https://www.mapbox.com/about/company [Date Accessed: 30/11/2020].
[34] "Google Maps for OSM Data", [Online]. Accessed: https://help.openstreetmap.org/questions/615/can-google-maps-take-osm-data-and-use-it-in-their-own-maps [Date Accessed: 24/03/2021]
[35] "German Real Estate Website Displaying OSM Data", [Online]. Accessed: https://www.capital.de/immobilien-kompass#karte [Date Accessed: 24/03/2021]
[36] "Pricing of Google Maps", [Online]. Accessed: https://cloud.google.com/maps-platform/pricing/?_ga=2.199066284.2006462946.1615828601-867933138.1615397538 [Date Accessed: 24/03/2021]
[37] "Create Your Own To-Do App with HTML5 and IndexedDB", [Online]. Accessed: https://blog.teamtreehouse.com/create-your-own-to-do-app-with-html5-and-indexeddb [Date Accessed: 01/12/2020].
[38] "Interactive map of shopping mall on HTML5 canvas", [Online]. Accessed: https://softwarecountry.com/articles/interactive-map-of-shopping-mall-on-html5-canvas/ [Date Accessed: 01/12/2020].
[39] "Simple PHP Password Hashing by Anthony Ferrara", [Online]. Accessed: ircmaxell@php.net [Date Accessed: 24/03/2021]

[40] "Free PHP Commenting System Tutorial", [Online]. Accessed: https://codeshack.io/commenting-system-php-mysql-ajax/ [Date Accessed: 24/03/2021]
[41] "Software Architecture", [Online]. Accessed: https://en.wikipedia.org/wiki/Software_architecture [Date Accessed: 24/03/2021]
[42] "Mobile phone usage in the UK", [Online]. Accessed: https://www.statista.com/statistics/289167/mobile-phone-penetration-in-the-uk/ [Date Accessed: 24/03/2021]
[43] "Statistics of Mobile Phone Internet Usage Compared to Desktop", [Online]. Accessed: https://www.broadbandsearch.net/blog/mobile-desktop-internet-usage-statistics [Date Accessed: 24/03/2021]
[44] "AONB Information", [Online]. Accessed: https://landscapesforlife.org.uk/about-aonbs/aonbs/ [Date Accessed: 24/03/2021]
[45] "GeoJSON Site Used to Map Markers and Boundaries", [Online]. Accessed: Geojson.io [Date Accessed: 24/03/2021]
[46] "Top 4 software development methodologies", [Online]. Accessed: https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/ [Date Accessed: 01/12/2020].
[47] "Personal GitHub Repository", [Online]. Accessed: https://github.com/RHUL-CS-Projects/IndividualProject_2020_Jack-Wearn [Date Accessed: 24/03/2021]
[48] "Different Types of Software Testing", [Online]. Accessed: https://hackr.io/blog/types-of-software-testing [Date Accessed: 24/03/2021]
[49] "Setting up Localhost on Mac", [Online]. Accessed: https://websitebeaver.com/set-up-localhost-on-macos-high-sierra-apache-mysql-and-php-7-with-sslhttps [Date Accessed: 24/03/2021]
[50] "Installing a Localhost server on Windows", [Online]. Accessed: https://www.techwalla.com/articles/how-to-install-a-localhost-server-on-windows [Date Accessed: 24/03/2021]
[51] "Setting up Apache and MySQL on Windows 10", [Online]. Accessed: https://codebriefly.com/how-to-setup-apache-php-mysql-on-windows-10/ [Date Accessed: 24/03/2021]
[52] "Setting up Localhost stack on Ubuntu", [Online]. Accessed: https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-ubuntu-18-04 [Date Accessed: 24/03/2021]
[53] "Facebook social media website", [Online]. Accessed: https://www.facebook.com [Date Accessed: 24/03/2021]
[54] "Heroku Link to Web Application", [Online]. Accessed: https://aonblocator-final-year-project.herokuapp.com/index.php [Date Accessible: 12/04/2021]
[55] "Signup Background from Johannes Plenio", [Online]. Accessed: https://unsplash.com/photos/RwHv7LgeC7s [Date Accessed: 10/03/2021]
[56] "Login Background From Klavs Taimins", [Online]. Accessed: https://unsplash.com/photos/w1sbytdTTKA [Date Accessed: 10/03/2021]
[57] "Single-Page Web Applications", [Online]. Accessed: what is a single-page web apps [Date Accessed: 25/03/2021]

Appendix A: Full Table of Allocation of Page IDs

Information Webpage Name	Corresponding Page ID for Database of Comments
Arnside & Silverdale	1
Blackdown Hills	2
Cannock Chase	3
Chichester Harbour	4
Chiltern Hills	5
Cornwall	6
Costwods	7
Cranborne Chase and the West Wiltshire Downs	8
Dedham Vale	9
Dorset	10
East Devon	11
Forest of Bowland	12
High Weald	13
Howardian Hills	14
Isle of Wight	15
Isles of Scilly	16
Kent Downs	17
Lincolnshire Wolds	18
Malvern Hills	19
Mendip Hills	20

Nidderdale	21
Norfolk Coast	22
North Devon Coast	23
North Pennines	24
Northumberland Coast	25
North Wessex Downs	26
Quantock Hills	27
Shropshire Hills	28
Solway Coast	29
South Devon	30
Suffolk Coast and Heaths	31
Surrey Hills	32
Tamar Valley	33
Wye Valley	34
Anglesey	35
Clwydian Range & Dee Valley	36
Gower	37
Llŷn	38
Wye Valley	34
Antrim Coast and Glens	39
Binevenagh	40
Causeway Coast	41

Lagan Valley	42
Mourne Mountains	43
Ring of Gullion	44
Sperrins	45
Strangford and Lecale	46

Appendix B: Project Diary

Within this Appendix, the entirety of the project diary can be seen below. It will show each of the diary entries that have been made throughout the entirety of this project.

Jack Wearn 10:23 am on October 9, 2020

Permalink

Initial Project meeting (What was discussed) (01/10/2020): > Discussed the changes I had made to the milestones section of the report. I had laid out my milestones in not the best way, so discussed the better way of laying them out and what to included (term 1 and term 2 milestones, reports, etc...). > Discussed MapBox as an option for creating the map application. I mentioned that it may make it all a bit too simple as everything I would need to create the map itself (route planning, markers, location services, etc...) is all already built into the API and just needs to be called. > Discussed how I want the project the run and what I will be including in my map application. Discussed in detail the user account feature I plan to implement. We spoke about maybe having routes saved that the user has done and potentially a rating system for different markers for anyone looking for somewhere to go. > Discussed the reports, to ensure I was doing them correctly (i.e. if there was any word limit, layout requirements, designs/flows to follow). > Planned for the next meeting to be on Monday 12th October 2020 and scheduled this in the diary at 11:30am. This will be a reoccurring meeting once every two weeks, with all meetings scheduled and in the diary.

Jack Wearn 10:18 am on October 9, 2020

Permalink

Week 2: Monday 05th: Tasks Completed: > Updated the Trello board with anything new I may have thought of over the weekend. This included different things that I will want to display on the map. > I began making a simple folder layout for the actual project (map) to be stored into and used for the production. This will all get submitted to the GitHub by the end of the week. > Looked into GitHub best practices, as it has been a while since I have used GitHub for production of a product and wanted to ensure that I knew all of the commands, etc... to be able to use it efficiently. Tuesday 06th: Tasks Completed: > Completed the second section (CSS section) of the web development report. Added some more code fragments to the first section (HTML section). > Worked some more on user login and sign up. Now have a proof of concept program which is able to accept user data and can create a profile for them. Just need it to sync up with the display on the HTML, CSS, etc.... > Did some research on some more map applications to find some positives and negatives. This will be added to the report on Open Street Map in a section outlining how others have done it. I can use this as inspirations of things I would like to do on my map and things to avoid as it may not be the best user friendly approach, etc... > Had a read through the W3 Consortium to check on the HTML standards that are set out in the latest version of HTML. This includes double checking all the tags are set out correctly and all of the required tags are included in my project. Wednesday 07th: Tasks Completed: > Created a program which displays MapBox Data, which includes a route mapping system, different layers of maps, etc... Issues: > Getting the MapBox data to work correctly, find my location and display the correct/fastest route > Tried resolving this by adjusting some of the code manually, but the issue with fastest route not always being correct is still a problem. Will need to correct this once I begin looking into route planning and why it may not always find the fastest route. Thursday 08th: Tasks Completed: > Went through some different map tile layer providers to find the most suitable maps tiles. I have found one which would be perfect for the areas of natural beauty as it has the different height layers for mountains and different terrain. I have decided I will also be including map layering, which means if the user wants to see a map which shows shops, train stations, etc... They can select from the menu this map. > Implemented a map switching feature onto my proof of concept program. This had quite a few issues detailed below. Issues: > Map switching not working correctly. Does not change the map. > Had to adjust the JavaScript. This now partially works but I want to get it working more efficiently, as it currently takes a long time to actually switch which map is being displayed. > New map showing over the top of old map, rather than replacing it > This was an error in the scripting code I had written along with the HTML. This was fixed by adding another argument which removes the previously viewed map for the new map tile layer. Friday 09th: ** Will update at end of day after completing tasks** One task already completed is to populate my diary with these entries

Jack Wearn 9:55 am on October 9, 2020

Permalink

Week 1: Monday 28th: Tasks Completed: > Worked on my project plan, to help set out what I will be completing at each milestone that I want to achieve. On this day, I had completed the abstract section (with changes that I discovered after doing more research into the topic) and the milestone section. > Began working on the proof of concept programs, firstly by creating a basic web development (HTML, CSS, JavaScript) Proof of Concept program to defamiliarise myself with the basics of web development. > I also started the report for this proof of concept, which will outline HTML, CSS, JavaScript and anything else that may come up for me in terms of basic web development. Issues: > There was not really any issues on this day, as most of what I was doing was getting report templates sorted and refamiliarise myself with the basics of web development practices. > The only issue I ran into was that my JavaScript file was not functioning correctly, but this was fixed by changing where in the HTML file the JavaScript file was linked. Tuesday 29th: Tasks Completed: > Began looking into Open Street Map data and the different ways I am able to display and manipulate this data. > Started creating a basic Proof of Concept program for Open Street Map technologies, which included then researching more into Leaflet JS, which is the one I believe I will be using to display my map and add markers, etc... to the map. > I also setup the report which will correspond with this proof of concept, which will be completed once I have fully tested and understood everything I need to for Open Street Map and Leaflet JS. Issues: > One issue I had was that Leaflet would not display the map correctly, it kept showing just a blank grey square where the map tiles should be. > This was resolved by changing the tile layer URL, as the one that I had originally tried was no longer available so could not be displayed. Wednesday 30th: Tasks Completed: > Continued to work with the Open Street Map proof of concept program, by trying to add specific markers, custom markers and circles and polygons onto the map. > Looking into another method of creating an Open Street Map, MapBox. Will discuss further in initial project meeting. > Created a tasks tab within the app Notion, so that I could clearly write down everything that I will need to do and this will be updated/moved onto a Trello Board, so that I can keep on top of all tasks that need to be completed. Thursday 01st October: Tasks Completed: > I had an initial project meeting with Matthew Hague. I will cover this meeting in a separate post. > On this day I setup the Trello board and began populating it with the tasks that I want to get completed when undertaking this project. > I continued to work on my first report on basic web development, so that it would be ready for the next meeting. > Continued working on my proof of concept programs with anything new that has come up that I would like to add. > I submitted my project plan for marking. Friday 02nd: Tasks Completed: > After the meeting I had decided, a way of extending my project would be to allow for user login/sign up. I began working on a proof of concept program for this and looking into the different ways that this can be completed. > Continued work on my report on web development. Now have the first section (outlining HTML) completed. Issues: > Couldn't get the user log in/sign up to work as intended, will need to look more into different ways of getting the user data and storing it and manipulating the web page so that once a user has logged in it will display their specific profile.

Jack Wearn 1:59 pm on December 7, 2020

Permalink

Last week, I was working on the interim report. This was finished by the initial deadline of Friday last week. I also worked on some other proof-of-concept programs, which were uploaded alongside the interim report. This week, I will be delivering my presentation regarding the first terms work that has been completed.

Jack Wearn 7:57 pm on November 27, 2020

Permalink

This week I have been working on the different proof of concept programs to ensure that everything is working as it should be. More specifically, I have been working on the offline mapping application aspects, HTML5 canvas and indexedDB. I have also been working on the interim report to get it finished.

Jack Wearn 9:02 pm on November 23, 2020

Permalink

Last week I worked on the Interim Report and the demo. I now have the map showing a custom search box which looks through a JSON list to find different AONB locations. Once it has found the one that the user is looking for, it pans to the marker which has been added to the map. Today, I had a meeting with Matthew Hague, where I gave a demo for the project so far. We also discussed the interim report and some of the sections that will make up the report. We have arranged to meet before the report is due on the 3rd December.

Jack Wearn 9:39 pm on November 15, 2020

Permalink

This week I have been working on the demo which I will be presenting in the next meeting with Matthew Hague. I have been working on creating the landing page, which will display the map and a mock up of the login and register pages. All the pages now have a working footer and the map finds the users location on load. Next week I will be working on getting the map to show an area of outstanding natural beauty and working towards getting the map to work offline and have route navigation.

Jack Wearn 12:42 pm on November 9, 2020

Permalink

Today was the meeting with Matthew Hague. In this meeting we covered some of the work I have done over the past two weeks and looked at the interim report I have written so far and how to continue working on this towards the deadline. Discussed the offline aspects of the map and how I would go about implementing this. Also discussed the route mapping algorithms. Have arranged to have a demo in the next meeting in two weeks time which will show an unpolished version of the final project. Will continue working on this during the next two weeks.

Jack Wearn 12:36 pm on November 8, 2020

Permalink

This week I have been working on the interim report. Writing up sections of it including the web development section and beginning the mapping technologies section. I have also continued working on the offline aspect of the map by continuing creating the proof of concept program. There was also a lecture on how to write the report during the week which went over the best practices in writing a report for the third year project

Jack Wearn 4:49 pm on November 2, 2020

Permalink

I have begun writing the report for the interim review. This includes writing the abstract, a section on the aims and objectives and finished the section of basic web development practices and the proof of concept programs I have created to show off these technologies and languages. The next step will be to continue working on the proof of concept programs, namely the offline maps aspect, where I am looking to create a version of a plug in which allows for the downloading of a range of tiles and using them saved tiles offline. This will be continued over the course of the week and a report for this technology will be written.

Jack Wearn 11:54 am on October 31, 2020

Permalink

Continued working on the code for the actual project. Have begun working on a proof of concept program for the offline aspect of the map. This will continue into next week. Continued work on my projects reports, including an evaluation report of different mapping technologies and which I may choose to use in the end.

Jack Wearn 12:55 pm on October 24, 2020

Permalink

Began work on the second iteration of the project, which will use MapBox API to help with the mapping side of things. Created my own tile for the maps tileset which shows custom trails, walking routed, etc... Began working on a report which outlines the difference between Vector based maps and tile maps. I like the tile method for online maps, as they load quicker and smoother. But vector maps would be useful when using the map offline as they can be more easily downloaded/cached. Created a basic user login system, which will need to be deployed with the help of a database to keep track of the users credentials as it currently just accepts whatever the user has entered and doesn't check to see if the user does in fact already exist.

Jack Wearn 5:32 pm on October 18, 2020

Permalink

I have setup the GitHub and the required folder/repositories for this (Main project folder, reports folder and proof of concept folders). I have also begun implementing more features to the overall project, this includes the user login/signup function (first concept of this) and begun getting the other required technologies for this setup for the main project, this will be things like MySQL, PHP, etc... I have also begun a section of the report which will outline the user login/sign up function and how it will work in the project and what I aim for it to achieve in the final project. To do this, I have made a few variations of the proof of concept program with different ways that this can be completed in the project and will cover which one I eventually go with and why. I have also begun researching the different areas of natural beauty to see if there is any other way in which I can get there coordinates outlined on the map application I will be making, such as if there is an API or a JSON which would have different coordinates on it or if I will be building my own and mapping them that way.

Jack Wearn 1:59 pm on March 19, 2021

Permalink

This week I have been working to finish the final end product. I have now corrected the code to allow for individual comments for each of the information web pages. Over the next few days, all of these information pages should be finished and then will be setup with Heroku.

Jack Wearn 4:46 pm on March 15, 2021

Permalink

I have been working on the project report and the final product. Mainly creating the information web pages, which are now around 80% completed. The report is also nearly finished which will give time for final testing and finishing touches which can be implemented into the product. I also have a meeting with Matthew Hague to discuss the finalisations of the project.

Jack Wearn 4:55 pm on March 4, 2021

Permalink

I have been working on the final product and the final report for the third year project. I have now finished a majority of the product itself and am working on the final touches. The report still needs more work but the sections are being written in good time now.

Jack Wearn 11:41 am on February 19, 2021

Permalink

Continued work on the proof of concept programs and the final report ready to send the draft on 19th Feb.

Jack Wearn 3:59 pm on February 5, 2021

Permalink

This week I have been working on the proof of concept programs for user registration and logging in. Both the PHP and node.js one. PHP one is now finished and will look to finish the node.js one early next week. The report sections for these programs has also been started.

Jack Wearn 10:26 am on January 22, 2021

Permalink

Today I created a signup system with error handling checking based on the data inputted. An error I was experiencing was it always threw an inputError (user did not enter some data). However this was corrected by changing the data which was being used and correcting the typo I had done. It now puts the user data into the database.

Jack Wearn 4:44 pm on January 21, 2021

Permalink

Finished setting up the backend for the database, using PHP, PHPMyAdmin and MySQL, which will be used for the first proof of concept program. An issue which I ran into when setting this up was that I was unable to use the mysql commands in the command line to connect to the database. This was resolved by adjusting the command that I was using, using the command: "/usr/local/mysql/bin/mysql -u root -p". The next issue I ran into was that I was unable to log into phpmyadmin. This was resolved by creating the config file correctly. I am now able to create the database for the first proof of concept program.

Jack Wearn 4:38 pm on January 19, 2021

Permalink

today I have been working on the first proof of concept program for this term. This included creating the php files for a user registration/login feature. This will continue to be worked on over the course of the week.

Jack Wearn 5:22 pm on January 13, 2021

Permalink

Initial meeting with Matthew Hague today for term 2. Points that were covered in the meeting: Interim Review – discussed how it went and when I should be receiving feedback. The work that was completed over the Christmas break – where I had completed the template of the information pages on the website. Also discussed the final report – will continue working on the interim report to extend it for the final report. Will be sending regular drafts for feedback. Follow up meeting scheduled for next Monday, then every 2 weeks after that. Work to be worked on over the next two weeks: More frontend, design features. Backend work – Database + user accounts.

Jack Wearn 12:30 pm on December 11, 2020

Permalink

This week I delivered my presentation and participated in the presentation meeting. I have also continued working on the demo with some other features which were discussed in the presentation through the use of questions.