

Final Year Project Report

Interim 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

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 Where other published and unpublished source materials have been used, these have been acknowledged.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: 7597 (Including all titles, code snippets and tables).

Student Name: Jack Wearn

Date of Submission: 04/12/2020



Signature:  . 

Table of Contents

Abstract.....	5
Chapter 1: Introduction	6
1.1 Project Specification, Aims and Objectives	6
1.2 User Interface Design	6
1.3 Summary of Related Literature	6
1.4 Summary of Milestones (Achieved and Planned)	7
Chapter 2: Web Development in HTML5	8
2.1 HTML, CSS, JavaScript and Other Basic Web Development Technologies	8
2.2 jQuery	10
2.3 HTML5 Canvas	11
2.4 Developing an Offline HTML5 Application	11
2.5 Application Caching	11
2.6 Service Worker	12
2.7 IndexedDB	12
Chapter 3: Open Street Map Data Representation	14
3.1 Raster/Image Tile Maps	14
3.2 Vector Tile Maps	14
3.3 Leaflet JS	14
3.4 Mapbox	15
3.5 Comparison Between Leaflet JS and Mapbox	15
Chapter 4: Proof-of-Concept Development	16
4.1 A “Hello World” Offline HTML5 Application	16
4.2 A “To-Do List” Application Using IndexedDB	17
4.3 Drawing Shapes Using HTML5 Canvas	20
4.4 Loading and Displaying Open Street Map Data	20
Chapter 5: Term One Project Development	23
5.1 Term One Demo Application	23
5.2 Project Diary/Work Log	24

5.3	Summary of Completed Work	25
5.4	Software Engineering Methodology	26
Chapter 6:	Assessment	28
6.1	Risks and Professional Considerations	28
6.2	Self-Evaluation for Term One	28
Conclusion	29
Bibliography	30
Appendix A: User Interface Design	32
Appendix B: HTML5 Canvas Example Outputs	36
Appendix C: Example of a Service Worker JavaScript File	38

Abstract

Recent developments in how websites are created has enabled the use of different tools and techniques to allow for a webpage to function offline as if a connection to the Internet is still present. For example, the creation of Service Workers has enabled for an effective offline experience. Combining this with the use of map data, allows for a user to interact with an online map application without the need for a stable connection to the Internet. During this project, the map aspect will focus on Areas of Outstanding Natural Beauty because there is a clear gap in the market of maps which specifically highlight these locations. Through the use of online documentation and articles surrounding offline capabilities of websites, proof-of-concept programs were created to test the different tools, programming languages and techniques in order to achieve a fluent transition between online and offline web pages. The proof-of-concept programs for the project include a “hello world” website which functions offline, displaying map data from OpenStreetMap 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 and ensure that there is no loss of performance when a user loses connection.

Chapter 1: Introduction

1.1 Project Specification, Aims and Objectives

The aim of the project is to create an offline mapping application, which uses the recent developments in HTML5 offline technologies. The application will be based on Open Street Map (OSM) data, which will be stored for use offline. The early deliverables will include the creation of proof-of-concept programs which will allow for a greater understanding of these key technologies in creating final product.

The proof-of-concept programs which form the early deliverables of the project will include a “hello world” offline HTML5 application, a “to-do list” application which will use indexedDB, drawing shapes using HTML5 canvas features and a web page which loads and displays OSM data.

Alongside the proof-of-concept programs, a number of reports will be created. The reports include basic web development in HTML5, advanced web-based technologies, developing an offline HTML5 application and OSM data representation. These reports will cover all aspects of the technology being used and how they will relate to the final deliverable of the project and improve the understanding of the different techniques, tools and languages.

The final deliverables of the project include a mapping application which works offline, allowing the user to load and display OSM map data while maintaining functionality while offline. For example, allowing the user to zoom and pan around the map. As an extension to the project, which has been included as part of the final deliverable is to allow for users to create and access accounts. This could then be extended further to allow for users to submit their own photographs of the AONB locations and leave comments for other users to see. Another possible extension is to allow for the user to search for specific locations on the map.

1.2 User Interface Design

Something to consider before the creation of the final product is the User Interface (UI) design which will be followed when creating the final web applications. There are a number of different ways to create a design for a website/webpage, however the most common is to create a wireframe design. When drawing out these designs, either by hand or using online software, there are a number of considerations that need to be taken into account. For example, ensuring that the users are placed in control of the interface, making it comfortable to interact with the product, reducing cognitive load and making user interfaces consistent [1]. Appendix A covers how to achieve these different points when creating a UI design and the initial design for the projects mapping application, using MockFlow [2].

1.3 Summary of Related Literature

During the course of the first term, a number of sources have now been discovered allowing for a better understanding of how this project will be created. The first of which was the MDN web docs by Mozilla [3]. The first iteration of offline web creation was performed with application caching, which Mozilla [3] have created in depth documentation on how to use. It was within this documentation that it was discovered that application caching is a deprecated asset in web development as it is no longer supported by a number of modern browsers. A proof-of-concept program was created to attempt to understand the current capabilities of application caching and

form the foundation for comparing with the new technology for offline web creation, service workers.

Prompted from the previous documentation, service workers were researched to understand this new feature that could be used for offline web applications. Again, Mozilla offered in depth documentation [4] on the feature, giving a large amount of background information on how they work and why they have been adopted by so many of the modern web browsers.

A useful resource, which was also identified in the project planning phase, was W3Schools [5]. They offer detailed tutorials and documentation on all elements of web production. For example, they offer tutorials on JavaScript, jQuery and HTML5 canvas. In the creation of the proof-of-concept programs, W3Schools was a useful resource to better understand the technologies involved.

1.4 Summary of Milestones (Achieved and Planned)

During the project plan phase, a set of term one milestones was set out as a guide of what should be completed throughout the term. This included researching for the creation of reports and proof-of-concept programs, the creation of proof-of-concept programs/demos and the writing up of the reports. These were all achieved, although some of the initial deadlines set out were either not achieved in the timeframe set or were completed sooner than what was expected. As seen from some of the diary entries from the first term, it was planned that the offline aspect of the project would be worked on/researched between the weeks of the seventh of October, when in reality it was still being worked on during the week beginning the 26th October, as seen from the extract of the project diary below:

“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. October 31, 2020” [6].

The second term will again have a set of milestones which will be worked towards, which will include the coding of the final deliverable for the project, alongside working on the final project report. The project diary will still be used to keep track of what has been worked on and what will be planned for the upcoming weeks.

Chapter 2: Web Development in HTML5

2.1 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) [7]. 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:

```
<!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 [8]. The code snippet below shows how a specific tag and class within a HTML document can be styles using in-head or an external CSS file:

<pre>/* Styling based on class */ .form-inline{ width: 50%; display: flex; justify-content: center; padding: none; }</pre>	<pre>/* Styling based on ID */ #submitbtn { background-color: white; border: white; border-radius: 5px; padding: none; }</pre>
--	--

To style an element in-line, you would need to specify the style required when creating the tag in the HTML document, which would look like the code snippet below:

```
<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 [9]. 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 [10]. 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 Bootstrap through jsDelivr would look like the following code snippet [11]:

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.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.

2.2 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 jQuery

jQuery is an example of a JavaScript library. Its main purpose is to simplify HTML DOM tree traversal and manipulation [12]. 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) [13]. 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.min.js"></script>
```

There is some basic syntax that is used when working with jQuery, which will look like the following code snippets from W3Schools [14]:

```
// 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 [15]:

```
$(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.

2.4 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 [16]. 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. 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 [17]:

```
<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.

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.

2.5 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.

2.6 Application Caching

Otherwise known as application caching, the technique works by saving all of the webpage information in local storage for use offline. In order to use the mechanism of application caching, a list of resources that will be available offline needs to be supplied to the web application. This is done in the form of a manifest. It should be referenced in the initial HTML tag, such as with the following code snippet:

```
<html manifest="/helloworld.appcache">
```

This points to a text file within the applications file structure. Within this manifest document, all of the resources that should be available while offline will need to be listed. This then informs the browser to gather these resources to be cached for use offline. The snippet below shows an example of what could be included within the manifest document:

```

CACHE MANIFEST
# Explicitly cached entries
index.html
# offline.html will be displayed if the user is offline
FALLBACK:
/ /offline.html
# All other resources (e.g. sites) require the user to be online.
NETWORK:
*
```

Some of the advantages of approaching offline web development by using application cache include increased speed and reduced server load [18]. Because the resources that are cached are stored locally, they will load up faster when the user loses connection to the internet and the browser only downloads the resources that have changed from the server, meaning that there is the reduced server load.

However, application cache is only available in secure contexts (HTTPS) and is a deprecated feature. What this means is that most modern browsers no longer support application cache, as shown in the Mozilla application cache documentation [18]. The recommended new feature to use for offline web development is service workers.

2.7 Service Worker

A service worker is an event-driven worker, which acts as a proxy server between the web application, browser and network when one is available [19]. To begin with, a service worker will need to be registered, which is done by using the “`ServiceWorkerContainer.register()`” method. If this method is successfully executed, the service worker will be downloaded to the users client and attempt the its next lifecycle phase, which is installation. The next phase after installation is activation. A service worker can also be extended in web development for more advanced features other than offline website browsing, such as push notifications and background sync [20]. Appendix C shows an example JavaScript file which goes through the code required to get a service worker up and running on a web application. The code snippet below shows how to create the initial 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.');`

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

(The remainder of this code is shown in appendix C).

2.8 IndexedDB

Another tool to consider when developing an offline web application is IndexedDB. This is essentially a more powerful option to the traditional “localstorage” option. It allows for the storing of large numbers of objects and can retrieve this data using a number of robust data access

mechanisms [21]. A website can multiple IndexedDB databases that it uses and each of these different databases will need to have its own unique name/identifier. Each of these different databases can then store one or more object stores. When setting up the IndexedDB, the database would first need to be created. This is done using a “createDatabase()” function using JavaScript. The code snippet below is an example of how to create the IndexedDB database [21]:

```
function createDatabase() {  
    var openRequest = localDatabase.indexedDB.open(dbName);  
  
    openRequest.onerror = function(e) {  
        console.log("Database error: " + e.target.errorCode);  
    };  
    openRequest.onsuccess = function(event) {  
        console.log("Database created");  
        localDatabase.db = openRequest.result;  
    };  
    openRequest.onupgradeneeded = function (evt) {  
        ...  
    };  
}
```

Chapter 3: Open Street Map Data Representation

Open Street Map (OSM) is open-source map data, used in the production of web applications, mobile applications and hardware devices [22]. 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.

3.1 Raster/Image 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 [23]. 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.

3.2 Vector Tile Maps

Vector tile maps are used to deliver small chunks of geographical data to the web browser when the user is accessing the mapping application [24]. 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 LineStrings. 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 [25], 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 [25].

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.

3.3 Leaflet JS

Leaflet JS is a powerful open-source JavaScript library [26], 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.

3.4 Mapbox

Mapbox is a Software Development Kit (SDK) created to make the process of making mapping applications easier [27]. 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.

3.5 Comparison Between Leaflet JS and Mapbox

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 Mapbox's 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.

Chapter 4: Proof-of-Concept Development

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.

4.1 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:

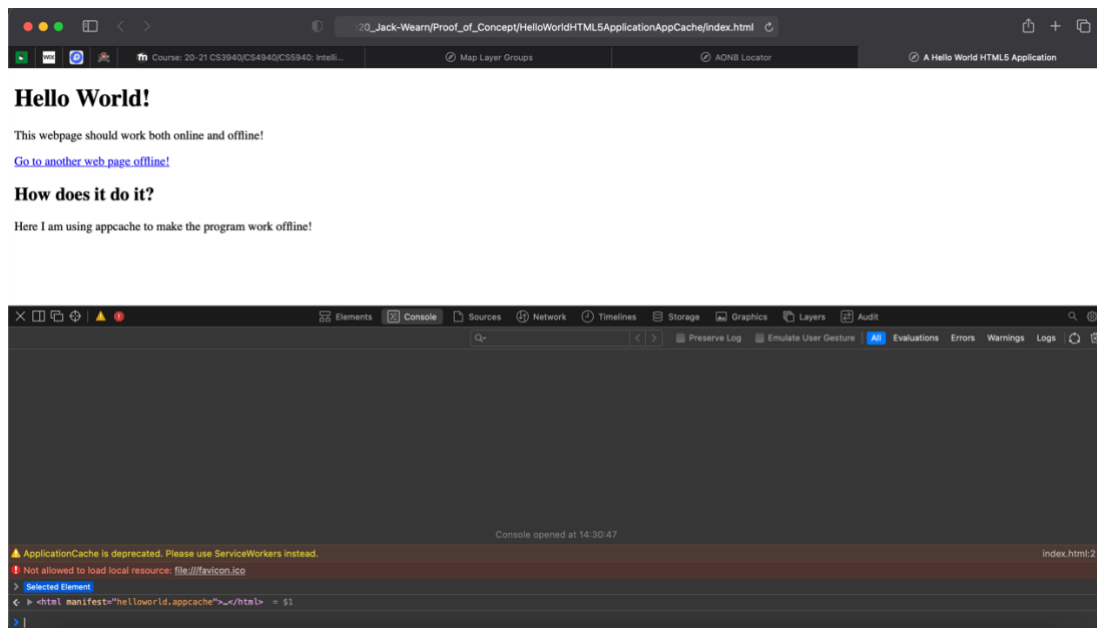


Figure 1. Screenshot displaying the error message displayed informing that application cache is deprecated and to use service workers.

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:

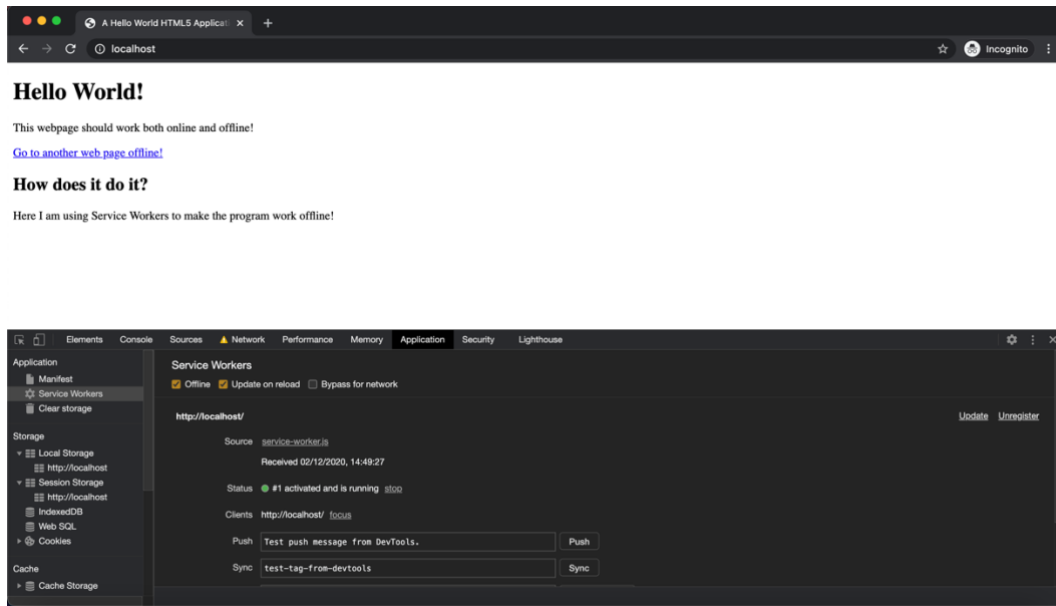


Figure 2. Screenshot displaying the working service worker for offline web application use.

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.

4.2 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 [30]. 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:

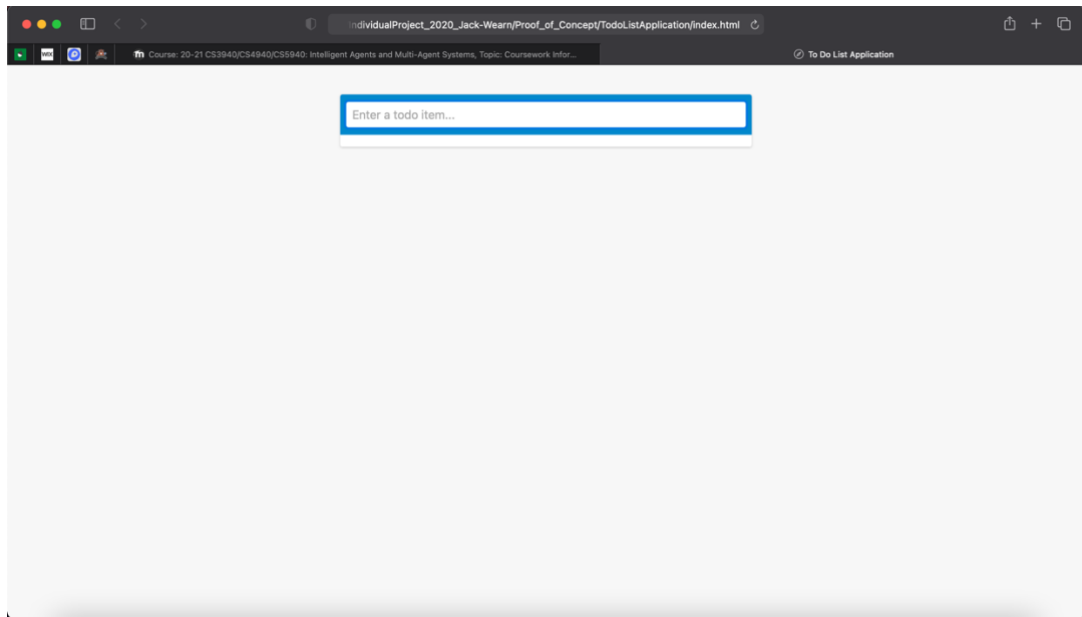


Figure 3. Screenshot showing the blank “to-list” application created.

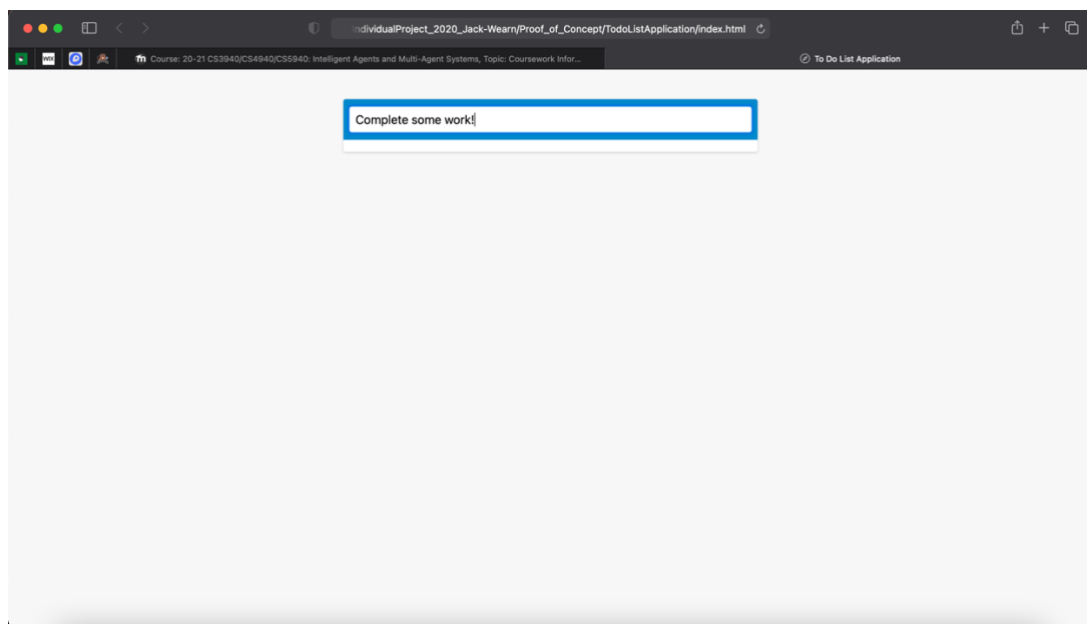


Figure 4. Screenshot showing how users would input data into the application.

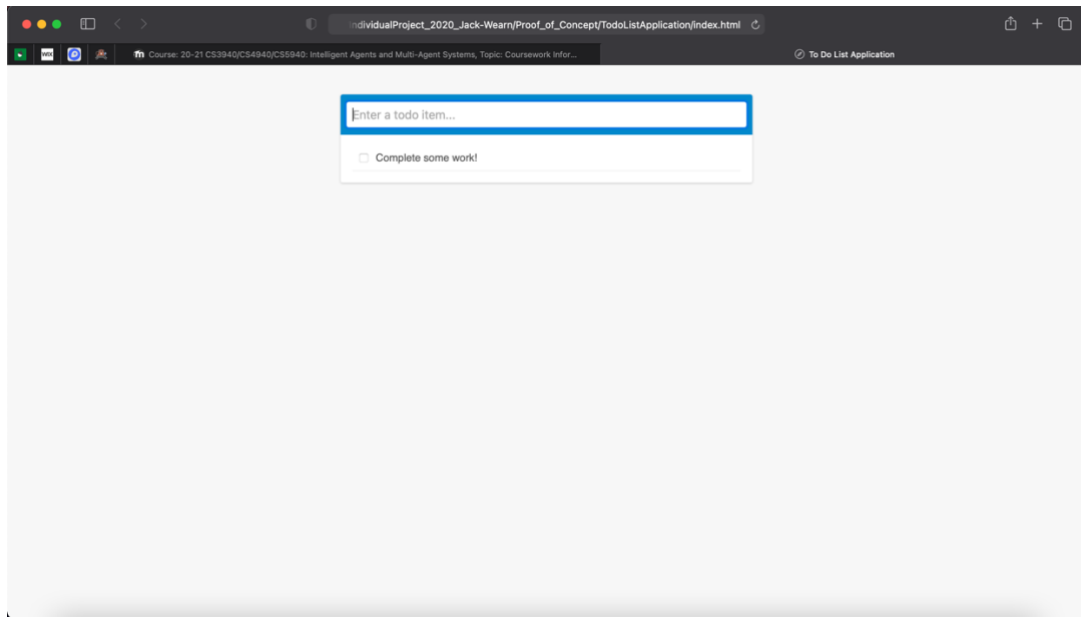


Figure 5. Screenshot showing the users input stored in a formal “to-do” list beneath the user input section.

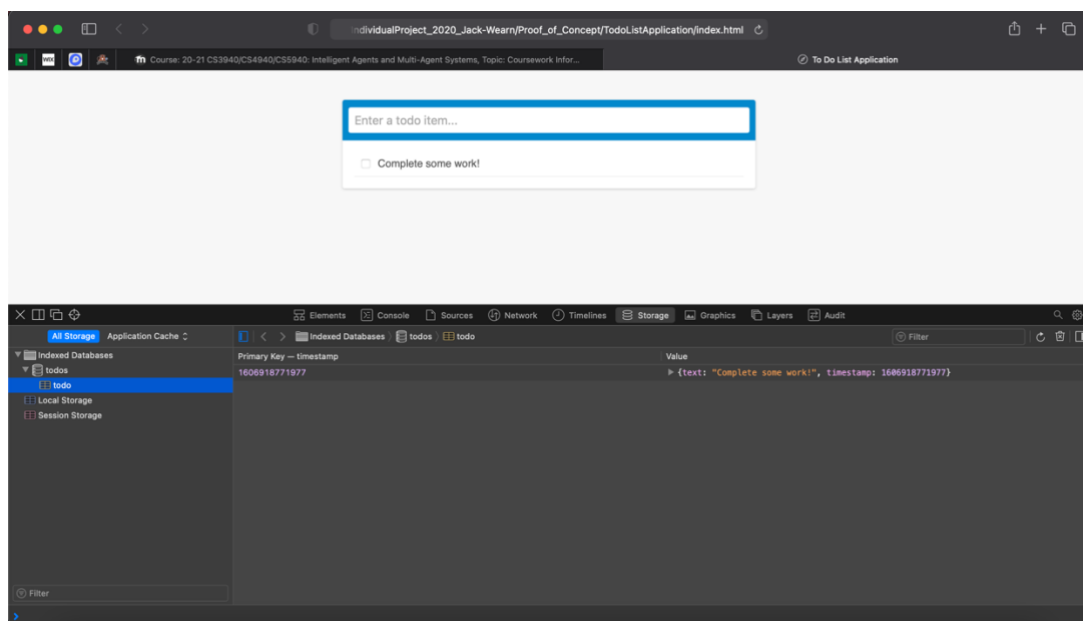


Figure 6. Screenshot showing what has been stored in the IndexedDB database, which is the users input.

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.

4.3 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 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:

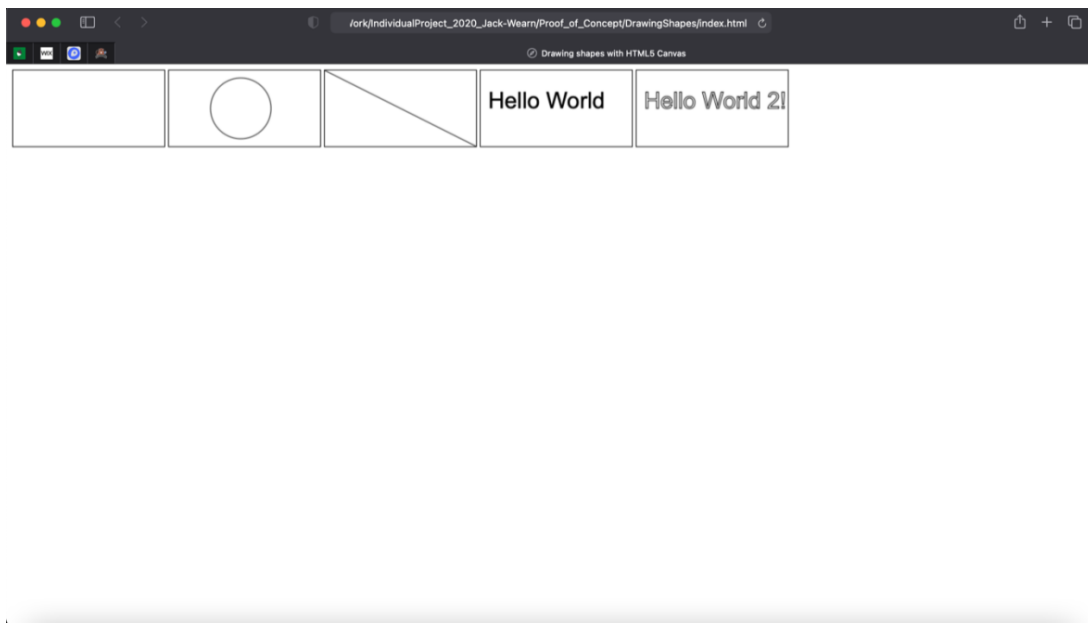


Figure 7. Screenshot displaying the output of the code for using HTML5 Canvas. More detail of the code used for this output can be seen in appendix B.

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 [29].

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.

4.4 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:

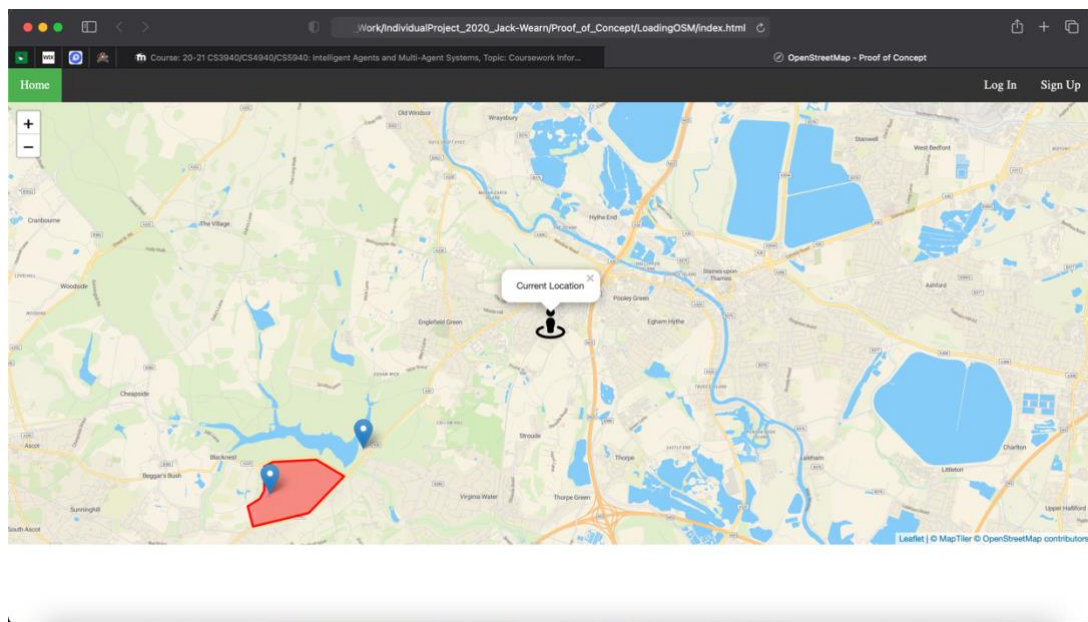


Figure 8. Screenshot showing the output of the map data using Leaflet JS to display the data.

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:

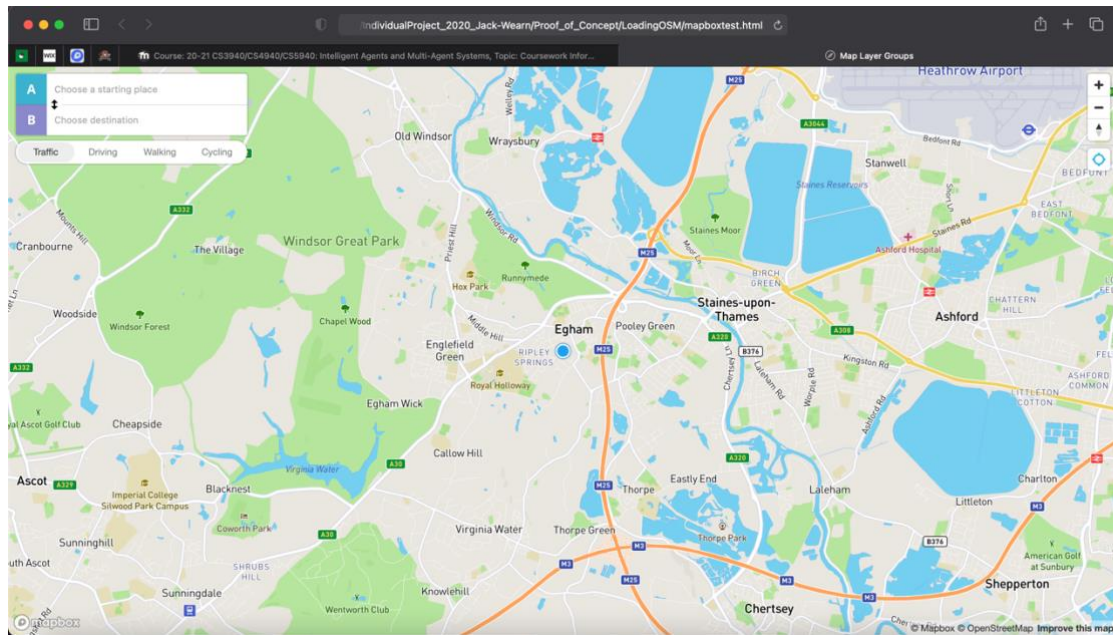


Figure 9. Screenshot showing the output of using Mapbox API to display the map data onto the web application.

Just like with Leaflet, Mapbox's other tools and extensions were tested 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.

Chapter 5: Term One Project Development

5.1 Term One 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 [6]. The screenshot below shows how the demo was created:

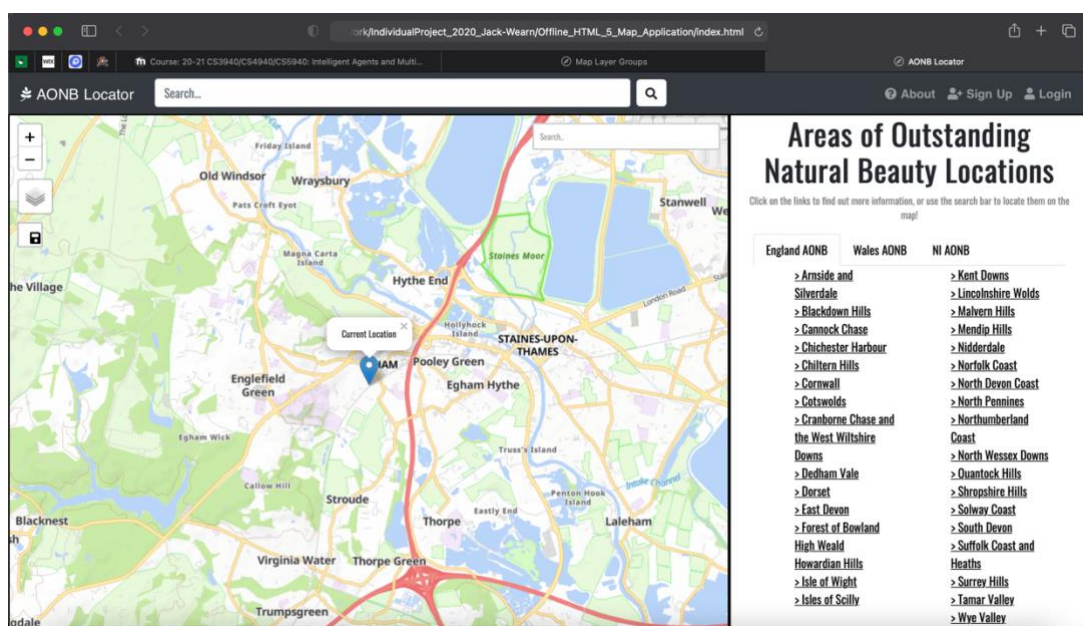


Figure 10. Screenshot displaying the final product of the demo which was created to test the knowledge gained through the creation of the proof-of-concept programs.

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:

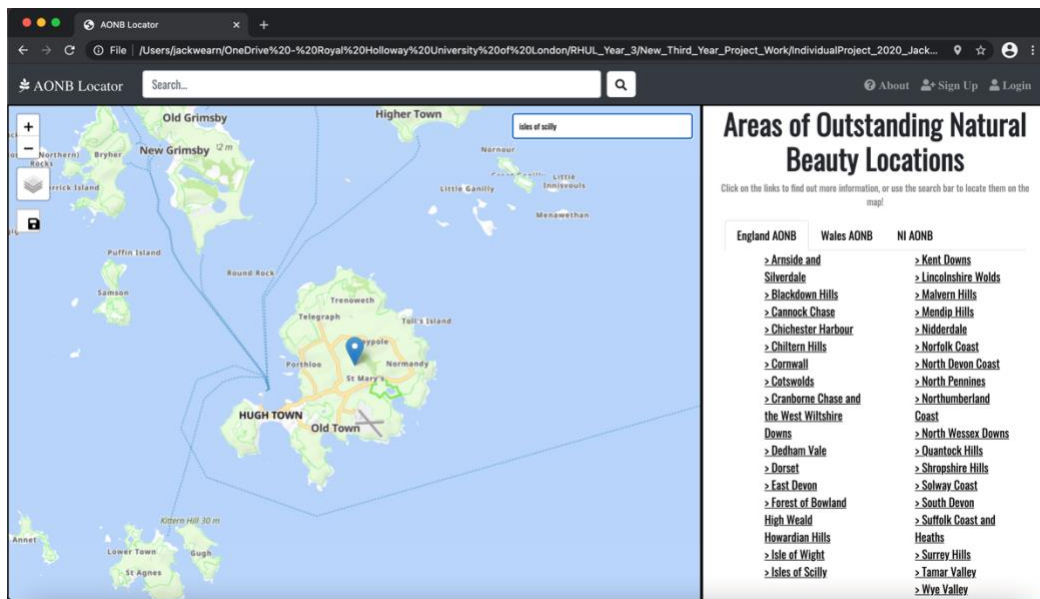


Figure 11. Screenshot showing what the map does when the user inputs a search for an AONB location.

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:

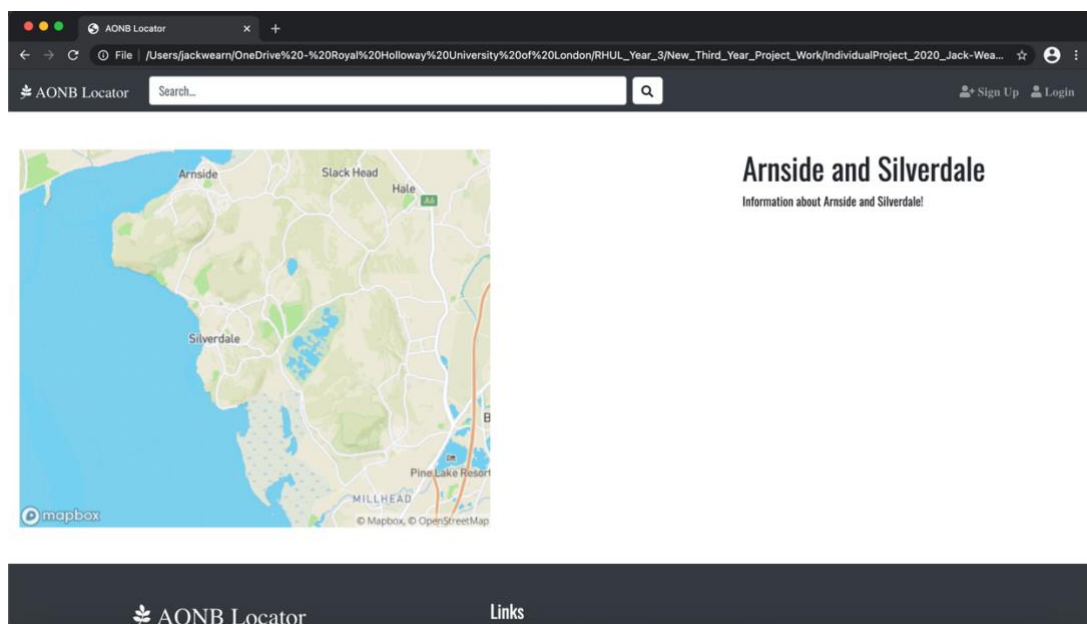


Figure 12. Screenshot displaying the template to be used for the information pages when clicked from the index web page.

5.2 Project Diary/Work Log

Throughout the course of this project, a diary will be updated regularly to give an idea of what has been worked on during the course of the weeks that this project is active. Throughout term one, these diary entries have been made weekly, outlining what tasks have been worked on each work along with any progress on the reports which were being worked on alongside the proof-of-concept programs/demos.

The project diary forms the basis of a work log. Using the diary to document progress each week allows for a better understanding of how the project is taking shape. It is also a convenient way of documenting any project related meetings, such as the initial diary post, which outlined what was discussed in the initial project meeting with the supervisor of the project. It is important to have this in written format, so that it can be referred to in later meetings and also to ensure that any tasks that were agreed to be completed are done so in the timeframe outlined within the meeting.

The diary entries do show an understanding of how the project is taking shape and allow for analysis of process. For example, as shown from the post on October 24, 2020 a new iteration of representing OSM data was being tested, as shown from the diary snippet below:

“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 user’s 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” [6].

This clearly shows the thought process going into both the creation of the proof-of-concept programs, but also decision making for the final project design.

5.3 Summary of Completed Work

Within each chapter of this report, it has gone into detail of some of the work and background theory that has been developed so far when developing this project. This section aims to summarise all of the work completed so far.

Figure 13. The table below shows a summary of the work that has been completed so far during term one of this project.

Work Completed	Summary of What Was Achieved
Proof-of-concept Programs/Demo	
Creation of the offline HTML5 application proof-of-concept programs.	After completing work on creating the two proof-of-concept programs related to offline HTML5 application creation, a better understanding of how a website can be displayed when a user is offline was achieved. Multiple methods of achieving this was also discovered which were otherwise not known.
Creation of the drawing shapes with HTML5 canvas feature proof-of-concept program.	After completing work on creating this proof-of-concept program, it gave a much better understanding of how to draw different elements onto a webpage. This will prove useful in the creation of the mapping application, as it will allow for more custom

	features to be implemented.
Creation of the “to-do list” application proof-of-concept program.	After completing this proof-of-concept program, a greater understanding of IndexedDB was achieved. A technology that had not previously been used but will prove to be useful when considering the offline aspects of the mapping application.
Creation of the displaying OSM data proof-of-concept program.	After completing these proof-of-concept programs, it allowed for a better comparison of how the OSM data will be represented on the final projects mapping application. It also allowed for experimenting with extra features, such as the search bar which was implemented.
Creation of the demo application.	By creating the demo application, it allowed for a better understanding of what the final project could look like when delivered. It displayed some of the key concepts of what the mapping application will be able to do, even with the most basic of functionality.
Reports	
Basic web development in HTML5 report.	
Advanced web technologies report.	
Developing an offline HTML5 application report.	
Open Street Map data representation report.	

All reports were then merged together within this Interim Review Report, to give an overall understanding of the project so far.

5.4 Software Engineering Methodology

The software engineering approach taken during the first phase of this project has followed an agile development approach. The initial concept was first planned, where it was decided that the map application would focus around AONB in the UK. The first iteration was then created. For this project, it was slightly different compared to some other software development projects, as the first iteration was the proof-of-concept programs. However, as with the agile approach, they were delivered and reviewed along with the project supervisor to create any needed changes, prompting a new iteration to be created. The agile approach can be seen in the image below, courtesy of Synopsys Editorial Team [31]:

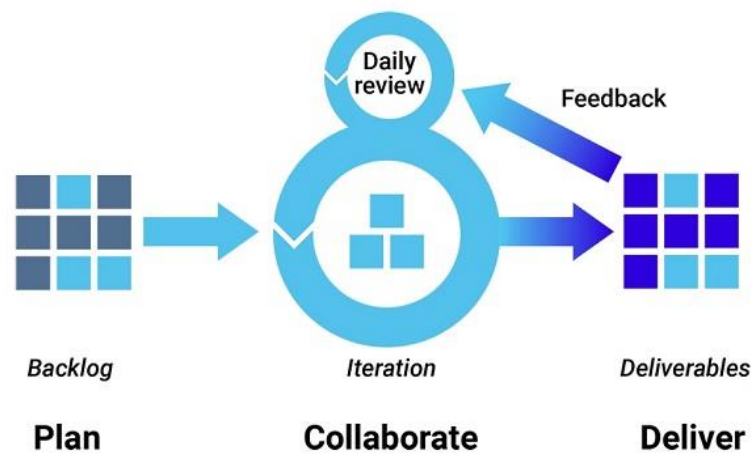


Figure 14. Example diagram of an agile development approach. Image courtesy of Synopsys Editorial Team [31].

However, when creating the demo of the project so far, from everything learnt in term one, the development followed a more waterfall development method/approach. For example, the requirements of the demo were first discussed in the first meeting on November 9, 2020 [6]. The user interface was then designed, showcasing all of the different elements that will appear on the website. This was then implemented in code as best as it could by sticking to the original designs and then presented in the meeting on November 23, 2020 [6]. The image below, again courtesy of Synopsys Editorial Team [31] showcases the waterfall development method/approach:

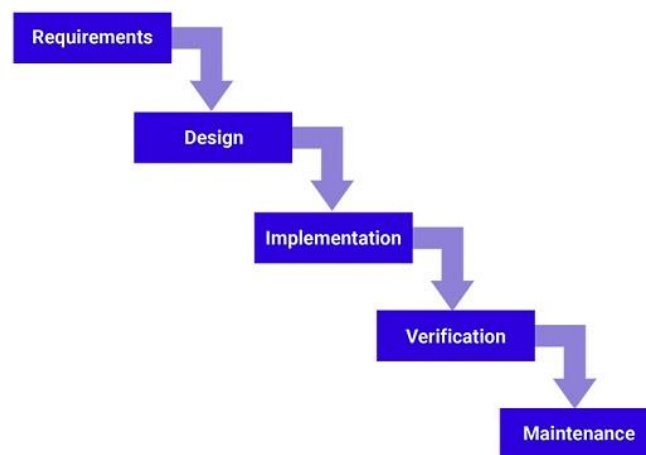


Figure 15. Example diagram of the waterfall development approach. Image courtesy of Synopsys Editorial Team [31].

Chapter 6: **Assessment**

6.1 Risks and Professional Considerations

There are a number of risks which were identified within the project planning phase of this project. However, as the project develops, more risks have been identified as potential risks to the project and should have serious professional considerations. Within this report, these risks will be summarised, along with the risk management that has also been identified.

One risk that was not planned for within the project plan was damage to the main hardware being used to create the project. Unfortunately, this did occur in the late stages of the project, with a computer device being accidentally damaged beyond repair. Although this was not planned for, or even considered before the incident, the management of this risk was well thought out and dealt with in a professional and timely manner. A replacement computer device was ordered and all data for the project was stored on the cloud. This meant that until the replacement machine arrived, the project could continue to be worked on using a different computer machine.

Another risk that has now been identified include the website data being too large to be cached, resulting in poor performance when using the web application offline. A way to plan for this is to ensure that when setting up the offline aspects of the website, only the essential resources are downloaded for the user, so they do not lose any necessary functionality but may no longer be able to access some of the less necessary resources.

Other risks will continue to arise and become relevant. Overall, during the course of this phase of the project, any risks that have arisen have been dealt with appropriately and any that could potentially harm the project have been identified and planned for.

6.2 Self-Evaluation for Term One

The work produced during the first term of this project has gone well. Overall, it was a productive period of the project with a lot of new information and knowledge gained which will prove useful in the creation of the final product. 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 will be best to use this when planning for what should be completed in the second term of this project. This means 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 will all be 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. During the start of the next term, some of the other features which will be included into the mapping application will be explored in more detail, such as the user accounts, along with proof-of-concept programs being created. However, overall, the progress made for this term has been positive.

Conclusion

A lot of work has gone into the first term phase of this project. The research conducted, along with the creation of the different proof-of-concept programs has allowed for a better understanding of how the project could be created and gain a deeper knowledge of some of the key techniques, technologies and programming languages which will go into the creation of the final project.

Over the course of the next phase of the project, a lot more work needs to go into creating the final, deliverable product. This will include creating the final website design, coding all of the required elements of the map aspect of the application and allowing for as close to full functionality of the map whilst interacting offline. This again will all be documented through the use of the project diary as this has proved to be a useful asset in during the first term phase.

Bibliography

- [1] “GOLDEN RULES OF UI DESIGN”, [ONLINE]. ACCESSED: [HTTPS://XD.ADOBE.COM/IDEAS/PROCESS/UI-DESIGN/4-GOLDEN-RULES-UI-DESIGN/](https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/) [DATE ACCESSED: 23/11/2020].
- [2] “MOCKFLOW, UI DESIGNER”, [ONLINE]. ACCESSED: [HTTPS://WWW.MOCKFLOW.COM](https://www.mockflow.com) [DATE ACCESSED: 23/11/2020].
- [3] “USING THE APPLICATION CACHE”, [ONLINE]. ACCESSED: [HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/Web/HTML/USING_THE_APPLICATION_CACHE](https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache) [DATE ACCESSED: 27/11/2020].
- [4] “USING SERVICE WORKERS”, [ONLINE]. ACCESSED: [HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/Web/API/SERVICE_WORKER_API/USING_SERVICE_WORKERS](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers) [DATE ACCESSED: 27/11/2020].
- [5] “ONLINE WEB TUTORIALS”, [ONLINE]. ACCESSED: [HTTPS://WWW.W3SCHOOLS.COM/](https://www.w3schools.com/) [DATE ACCESSED: 27/11/2020].
- [6] “PROJECT DIARY”, [ONLINE]. ACCESSED: [HTTPS://PD.CS.RHUL.AC.UK/2020-21/](https://pd.cs.rhul.ac.uk/2020-21/) [DATE ACCESSED: 29/11/2020].
- [7] “HTML, LIVING STANDARD”, [ONLINE]. ACCESSED: [HTTPS://HTML.SPEC.WHATWG.ORG/MULTIPAGE/](https://html.spec.whatwg.org/multipage/) [DATE ACCESSED: 02/10/2020].
- [8] LIE, H.W. AND BOS, B., 2005. *CASCADING STYLE SHEETS: DESIGNING FOR THE WEB, PORTABLE DOCUMENTS*. ADDISON-WESLEY PROFESSIONAL.
- [9] SEIBEL, P., 2009. *CODERS AT WORK: REFLECTIONS ON THE CRAFT OF PROGRAMMING*. APRESS.
- [10] “BOOTSTRAP (FRONT-END FRAMEWORK)”, [ONLINE]. ACCESSED: [HTTPS://EN.WIKIPEDIA.ORG/WIKI/BOOTSTRAP_\(FRONT-END_FRAMEWORK\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)) [DATE ACCESSED: 30/11/2020].
- [11] “GET BOOTSTRAP”, [ONLINE]. ACCESSED: [HTTPS://GETBOOTSTRAP.COM](https://getbootstrap.com) [DATE ACCESSED: 30/11/2020].
- [12] “JQUERY”, [ONLINE]. ACCESSED: [HTTPS://EN.WIKIPEDIA.ORG/WIKI/JQUERY](https://en.wikipedia.org/wiki/JQuery) [DATE ACCESSED: 30/11/2020].
- [13] “CONTENT DELIVERY NETWORK”, [ONLINE]. ACCESSED: [HTTPS://EN.WIKIPEDIA.ORG/WIKI/CONTENT_DELIVERY_NETWORK](https://en.wikipedia.org/wiki/Content_delivery_network) [DATE ACCESSED: 30/11/2020].
- [14] “JQUERY SYNTAX”, [ONLINE]. ACCESSED: [HTTPS://WWW.W3SCHOOLS.COM/JQUERY/JQUERY_SYNTAX.ASP](https://www.w3schools.com/jquery/jquery_syntax.asp) [DATE ACCESSED: 30/11/2020].
- [15] “JQUERY CLICK() EVENT”, [ONLINE]. ACCESSED: [HTTPS://WWW.W3SCHOOLS.COM/JQUERY/TRYIT.ASP?FILENAME=TRYJQUERY_CLICK](https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_click) [DATE ACCESSED: 30/11/2020].
- [16] “CANVAS ELEMENT”, [ONLINE]. ACCESSED: [HTTPS://EN.WIKIPEDIA.ORG/WIKI/CANVAS_ELEMENT](https://en.wikipedia.org/wiki/Canvas_element) [DATE ACCESSED: 30/11/2020].

- [17] “HTML CANVAS, DRAW A CIRCLE”, [ONLINE]. ACCESSED: [HTTPS://WWW.W3SCHOOLS.COM/HTML/HTML5_CANVAS.ASP](https://www.w3schools.com/html/html5_canvas.asp) [DATE ACCESSED: 30/11/2020].
- [18] “USING THE APPLICATION CACHE”, [ONLINE]. ACCESSED: [HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/WEB/HTML/USING_THE_APPLICATION_CACHE](https://developer.mozilla.org/en-US/docs/Web/HTML/Using_the_application_cache) [DATE ACCESSED: 30/11/2020].
- [19] “SERVICE WORKER 3API”, [ONLINE]. ACCESSED: [HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/WEB/API/SERVICE_WORKER_API](https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API) [DATE ACCESSED: 30/11/2020].
- [20] “SERVICE WORKERS: AN INTRODUCTION”, [ONLINE]. ACCESSED: [HTTPS://DEVELOPERS.GOOGLE.COM/WEB/FUNDAMENTALS/PRIMERS/SERVICE-WORKERS](https://developers.google.com/web/fundamentals/primers/service-workers) [DATE ACCESSED: 30/11/2020].
- [21] “USING THE HTML5 INDEXEDDB API”, [ONLINE]. ACCESSED: [HTTPS://DEVELOPER.IBM.COM/TUTORIALS/WA-INDEXEDDB/](https://developer.ibm.com/tutorials/wa-indexeddb/) [DATE ACCESSED: 30/11/2020].
- [22] “OPEN STREET MAP”, [ONLINE]. ACCESSED: [HTTPS://WWW.OPENSTREETMAP.ORG/ABOUT](https://www.openstreetmap.org/about) [DATE ACCESSED: 30/11/2020].
- [23] “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/](https://bachasoftware.com/what-is-tile-and-differentiate-between-raster-tile-and-vector-tile/) [DATE ACCESSED: 30/11/2020].
- [24] “VECTOR TILES”, [ONLINE]. ACCESSED: [HTTPS://WIKI.OPENSTREETMAP.ORG/WIKI/VECTOR_TILES](https://wiki.openstreetmap.org/wiki/Vector_tiles) [DATE ACCESSED: 30/11/2020].
- [25] “RASTER VS. VECTOR: PROS AND CONS OF BOTH MAP TILE TYPES”, [ONLINE]. ACCESSED: [HTTPS://WWW.GEOAPIFY.COM/RASTER-VS-VECTOR-MAP-TILES](https://www.geoapiify.com/raster-vs-vector-map-tiles) [DATE ACCESSED: 30/11/2020].
- [26] “LEAFLET, AN OPEN-SOURCE JAVASCRIPT LIBRARY FOR MOBILE-FRIENDLY INTERACTIVE MAPS”, [ONLINE]. ACCESSED: [HTTPS://LEAFLETJS.COM](https://leafletjs.com) [DATE ACCESSED: 30/11/2020].
- [27] “MAPBOX, THE BUILDING BLOCKS”, [ONLINE]. ACCESSED: [HTTPS://WWW.MAPBOX.COM/ABOUT/COMPANY](https://www.mapbox.com/about/company) [DATE ACCESSED: 30/11/2020].
- [28] “MAKING A SIMPLE SITE WORK OFFLINE WITH SERVICE WORKER”, [ONLINE]. ACCESSED: [HTTPS://CSS-TRICKS.COM/SERVICEWORKER-FOR-OFFLINE/](https://css-tricks.com/serviceworker-for-offline/) [DATE ACCESSED: 01/12/2020].
- [29] “INTERACTIVE MAP OF SHOPPING MALL ON HTML5 CANVAS”, [ONLINE]. ACCESSED: [HTTPS://SOFTWARECOUNTRY.COM/ARTICLES/INTERACTIVE-MAP-OF-SHOPPING-MALL-ON-HTML5-CANVAS/](https://softwarecountry.com/articles/interactive-map-of-shopping-mall-on-html5-canvas/) [DATE ACCESSED: 01/12/2020].
- [30] “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](https://blog.teamtreehouse.com/create-your-own-to-do-app-with-html5-and-indexeddb) [DATE ACCESSED: 01/12/2020].
- [31] “TOP 4 SOFTWARE DEVELOPMENT METHODOLOGIES”, [ONLINE]. ACCESSED: [HTTPS://WWW.SYNOPSYS.COM/BLOGS/SOFTWARE-SECURITY/TOP-4-SOFTWARE-DEVELOPMENT-METHODOLOGIES/](https://www.synopsys.com/blogs/software-security/top-4-software-development-methodologies/) [DATE ACCESSED: 01/12/2020].

Appendix A: User Interface Design

The screenshot below shows an example of how the mapping application has been designed so far, using an application called MockFlow [2]:

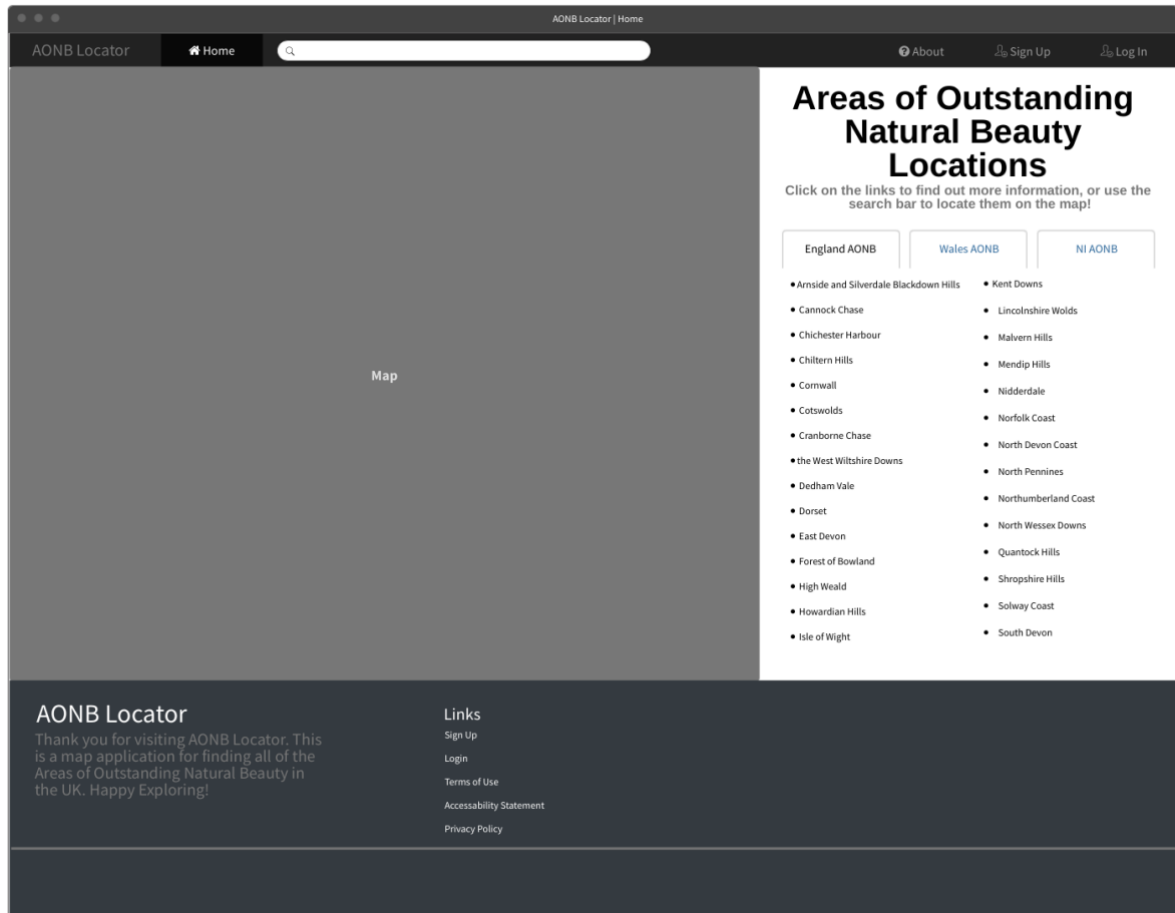


Figure 16. Screenshot showing the initial design for the index (home) page of the mapping application, created using MockFlow [2].

The screenshot below shows an example of the initial design for a user registration web page, which could be used in the final deliverable project. All designs were created using MockFlow [2].

AONB Locator | Sign Up

Home About Sign Up Log In

Sign Up

Email

Confirm Email

Password

Confirm Password

Sign Up

AONB Locator
Thank you for visiting AONB Locator. This is a map application for finding all of the Areas of Outstanding Natural Beauty in the UK. Happy Exploring!

Links
[Sign Up](#)
[Login](#)
[Terms of Use](#)
[Accessibility Statement](#)
[Privacy Policy](#)

Figure 17. Screenshot showing the initial design of a user sign up page, designed using MockFlow [2].

Below is a screenshot of the initial design for the user login web page/system. This will follow a similar pattern to the registration web page, however this could be subject to change if included in the final project.

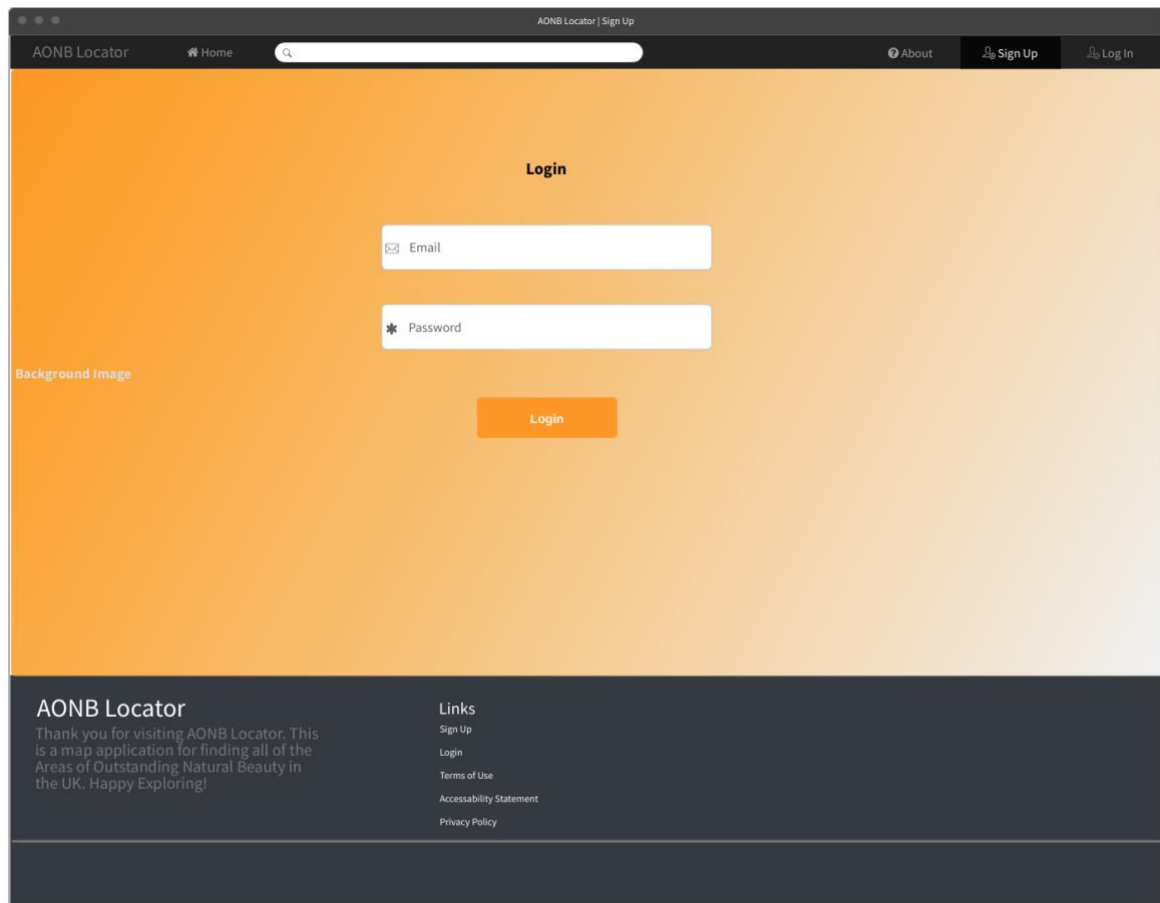


Figure 18. Screenshot showing the initial design for a user login system/web page, designed using MockFlow [2].

Once you have a number of web pages designed, an application map is able to be generated. By using MockFlow [2], this map was able to be created through the software, to produce the map displayed in the screenshot below.



Figure 19. Screenshot showing an example of how some of the webpage may interact with each other in term of the map of the application itself, created through the use of tools on MockFlow [2].

Appendix B: HTML5 Canvas Example Outputs

There are a number of different shapes, colours and elements which can be drawn using HTML5 canvas. Below are some examples of what can be drawn by using the HTML5 canvas feature, along with the code snippets which get them to work. These have been based off the tutorials supplied by W3Schools [5].

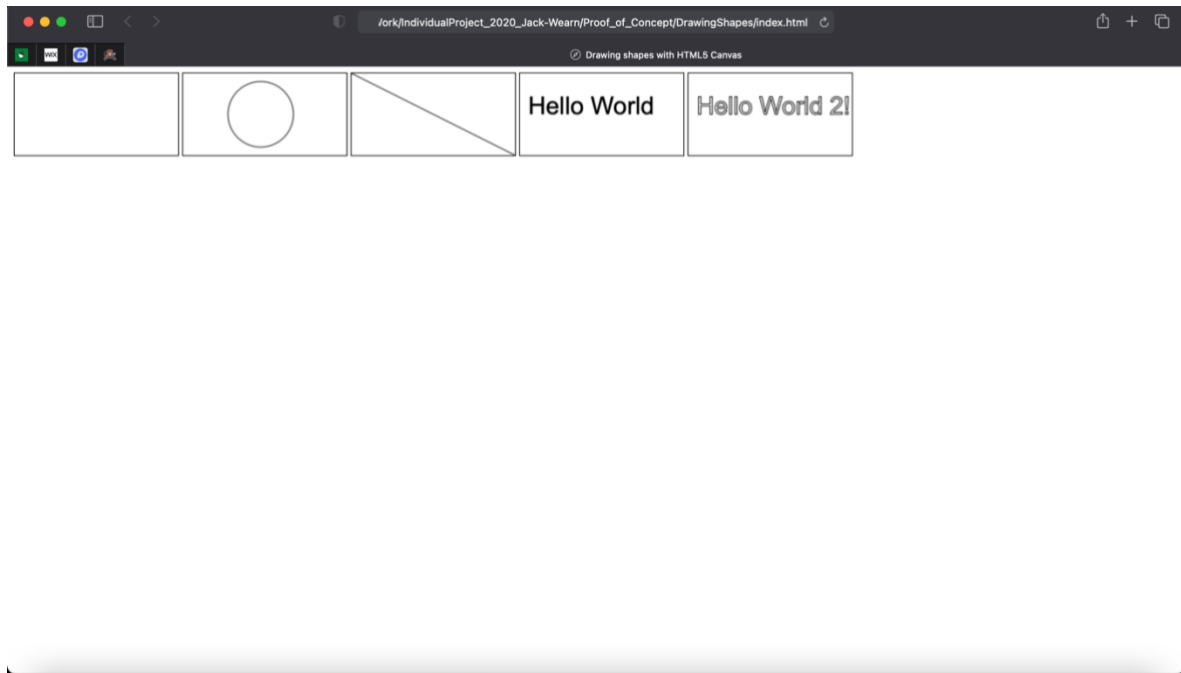


Figure 20. Screenshot of the output of multiple different canvas drawing methods, such as drawing an empty canvas, a circle, line, text and stroke text on the canvas.

The code for the above examples is shown below, each of the different outputs is labelled with a comment in the HTML and JavaScript code:

```
<!-- Drawing an empty canvas to the screen -->
<canvas id="Empty" width="200" height="100" style="border:1px solid
  #000000;"></canvas>
<!-- Drawing a canvas with a circle to the screen -->
<canvas id="circle" width="200" height="100" style="border:1px solid
  #000000;"></canvas>
<!-- Drawing a canvas with a line to the screen -->
<canvas id="line" width="200" height="100" style="border:1px solid
  #000000;"></canvas>
<!-- Drawing a canvas with plain text to the screen -->
<canvas id="text" width="200" height="100" style="border:1px solid
  #000000;"></canvas>
<!-- Drawing a canvas with stroke text to the screen -->
<canvas id="text2" width="200" height="100" style="border:1px solid
  #000000;"></canvas>
<script>
// Drawing a canvas with a circle to the screen
var c = document.getElementById("circle");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
```

```
// Drawing a canvas with a line to the screen
var x = document.getElementById("line");
var xy = x.getContext("2d");
xy.moveTo(0, 0);
xy.lineTo(200, 100);
xy.stroke();

// Drawing a canvas with plain text to the screen
var c = document.getElementById("text");
var ctx2 = c.getContext("2d");
ctx2.font = "30px Arial";
ctx2.fillText("Hello World", 10, 50);

// Drawing a canvas with stroke text to the screen
var c = document.getElementById("text2");
var ctx3 = c.getContext("2d");
ctx3.font = "30px Arial";
ctx3.strokeText("Hello World 2!", 10, 50);
</script>
```

Appendix C: Example of a Service Worker JavaScript File

The below code is based off of a tutorial for service workers by Nicolas Bevacqua from CSS-Tricks [28]. The first function directly below this paragraph is stored in one file (script.js) and the remaining code is stored in a separate file (service-worker.js).

```
"use strict";
console.log('WORKER: executing.');
```

```
var version = 'v1::';

var offlineFundamentals = [
  '',
  'script.js'
];

self.addEventListener("install", function(event) {
  console.log('WORKER: install event in progress.');
```

```
  event.waitUntil(
    caches
      .open(version + 'fundamentals')
      .then(function(cache) {
        return cache.addAll(offlineFundamentals);
      })
      .then(function() {
        console.log('WORKER: install completed');
      })
  );
});

self.addEventListener("fetch", function(event) {
  console.log('WORKER: fetch event in progress.');
```

```
  if (event.request.method !== 'GET') {
    console.log('WORKER: fetch event ignored.', event.request.method,
      event.request.url);
    return;
  }

  event.respondWith(
    caches
      .match(event.request)
      .then(function(cached) {
        var networked = fetch(event.request)
          .then(fetchedFromNetwork, unableToResolve)
          .catch(unableToResolve);
        console.log('WORKER: fetch event', cached ? '(cached)' :
          '(network)', event.request.url);
        return cached || networked;

        function fetchedFromNetwork(response) {
          var cacheCopy = response.clone();

          console.log('WORKER: fetch response from network.',
            event.request.url);

          caches
            .open(version + 'pages')
            .then(function add(cache) {
              return cache.put(event.request, cacheCopy);
            })
        }
      })
  );
});
```

```

        .then(function() {
            console.log('WORKER: fetch response stored in cache.',
event.request.url);
        });
        return response;
    }

    function unableToResolve () {
        console.log('WORKER: fetch request failed in both cache and
network.');
```

```

        return new Response('<h1>Service Unavailable</h1>', {
            status: 503,
            statusText: 'Service Unavailable',
            headers: new Headers({
                'Content-Type': 'text/html'
            })
        });
    }
}

});
});

self.addEventListener("activate", function(event) {
    console.log('WORKER: activate event in progress.');
```

```

    event.waitUntil(
        caches
            .keys()
            .then(function (keys) {
                return Promise.all(
                    keys
                        .filter(function (key) {
                            return !key.startsWith(version);
                        })
                        .map(function (key) {
                            return caches.delete(key);
                        })
                );
            })
            .then(function() {
                console.log('WORKER: activate completed.');
```

```

            })
        );
    });
});

```