ucc

Coláiste na hOllscoile Corcaigh, Éire
University College Cork, Ireland

Final Year Project

Name: John Creedon

Title: How Crowdsourcing Can Map Out Cleaner Routes for Dog Walkers

Supervisor: Gavin Russell

BA Digital Humanities & Information Technology

April 2024

Abstract

This project aimed to develop a visually appealing, user-friendly web mapping application for aiding the disposal of dog waste and the need thereof. The application is designed for crowdsourced updates from volunteers. All application content would be user-generated by logging the locations of dog waste, bagged waste, litter signs, and bins both public and dog specific. This user engagement would contribute to building up a comprehensive database; this data would in turn populate corresponding icons on a map. Knowing the location of bins and the prevalence of dog waste, owners can plan their dog walking routes more effectively. Accessing the location of nearby bins is now within arm's reach.

Utilizing the Leaflet library an interactive map was built entirely using HTML, CSS, PHP, and JavaScript. These combined to create a user-friendly and intuitive interface that connected to a robust server hosting the code and databases. PHP scripts and MySQL commands would enable users to register for, login to, and acquire the requisite permission to use the map editing facility.

Using geolocation and intuitive icons, such as a dog poo icon to represent dog waste and a bin icon for bins, users can log their location on the map. Saving these triggers an AJAX request to a PHP script to save the markers type, latitude, and longitude to a MySQL database. From this database another map available to all can replicate the location of these icons. These can be filtered by type and using a routing formula the nearest bin can be detected and directions given to its location.

The integration of these technologies in a seamless fashion has created a successfully deployed application. Drawing on the strengths of the crowd it has been able to gather information from as far afield as Boston, Thurso in the Scottish Highlands to the beachfront of Poetto in Sardinia.

Introduction

Overview of the Application

The project's aim was to develop a mobile application that enabled crowdsourced volunteers to log incidences of dog waste and the locations of bins in their areas. The application would serve as a practical tool in providing real-time, geolocated data that could guide users directly to bin locations. With this information, users would have the capability to plan their walking routes in advance, allowing them to dispose of dog waste responsibly. This would allow for the development and adoption of cleaner walking routes.

Personal Motivation

This project was borne out of my frustrations experienced as a dog walker. I have no problem cleaning up after my dog, although it can be quite inconvenient at times. The issue is the lack of facilities for disposing of dog waste. Are they in short supply or am I not aware of where they are? Judging by the amount of dog waste and bagged dog waste on the footpaths I am not the only one faced with this. My walking routes are often planned and sometimes dictated by my knowledge of where bins are located. The further away from the city center you go the fewer bins are available and unsurprisingly, dog waste is more visible. Despite having access to an abundance of information on my phone, I was surprised that there was no information on where my nearest bin was. The application would ideally show how it is possible to create a cleaner environment by highlighting the dual issues of dog litter and the lack of dog bins.

Academic Foundations

Throughout this project, my main goal was to effectively apply the diverse skills I had acquired during my three-year course in Digital Humanities and Information Technology at University College Cork. HTML, CSS, and JavaScript combined with an introduction to multimedia systems produced a visually appealing user interface. The programming knowledge gained from learning

Python, although not used, laid a foundation for JavaScript in introducing common concepts that proved invaluable in problem solving such as arrays, loops, objects, methods and functions. Databases were developed using MySQL and PHP, providing a backend to effectively store and retrieve user generated content. While considering content management systems like WordPress and Drupal, I opted to build the application from scratch. The application I created is testament to this and the confidence to do so was underpinned by this knowledge.

Open-Source Commitment

I committed to using free and open-source materials as much as possible in this project, aligning with principles of accessibility and transparency of the Digital Humanites. Initially, I secured the domain and web hosting through paid services, which was an important learning experience for me. This early financial outlay reinforced my dedication to minimizing further expenses and underscored the importance of a sustainable development model. As a result, I opted towards employing a combination of open-source and freely available solutions to ensure the application stayed free of charge. In the design process section of this report, I have detailed my choices of specific tools and methods, highlighting this commitment.

Crowdsourcing and Humanities

Drawing on the crowdsourcing aspect of digital humanities, this project engaged a community of users to contribute to the collection and verification of data. This method improved the data quality and application functionality. It was also hoped it would foster a sense of community and collective responsibility among users.

Identifying the User

To identify the key requirements and functionality to help in designing an application like this a user profile was created to replicate that of a typical dog walker. The design would be focused on their needs and wants in relation to locating bins for disposing of dog waste and knowing areas to avoid, especially those with a high incidence of dog waste.

The user profile was based on a middle-aged male aged 57 called Brian. He is a daily dog walker who was becoming frustrated by the sight of dog waste and abandoned bagged dog waste. His five-year-old Cockerpoo has been sick in the past from interacting with other dogs' waste. He has also become frustrated by the need to constantly reel the dog back from smelling the waste. He conscientiously cleans up his dogs' waste and disposes of it at the nearest bin known to him or brings it back home to dispose of there.

He needs reading glasses but does not wear them whilst walking. While he is comfortable using a smartphone, he prefers simple application interfaces and large, easy-to-read text

Brian wants an application that can find and direct him to the nearest bin from wherever he is at the time. He also wants to be able to update the application with bins and dog waste that he comes across to help others who find themselves in the same predicament.

The application would be designed with these needs in mind:

- Bin Locator - GPS-based map showing Brian's location and nearby dog waste bins. Clear, easy to see markings for the bins on the map.
- Assisted Navigation - Routing/walking directions to the nearest bin

- Crowd sourced reporting - To keep the bin locations and waste hot spots up to date, the application should allow users like Brian to report new bin locations or areas with high instances of dog waste.
- Accessible interface and streamlined experience – it should be easily accessible with large easy to read features. Navigation should be intuitive, cutting down on complex menus and settings.

Environmental Scan

To tailor the application to the needs of the intended end users an environmental scan was done of similar applications. The purpose of this was to see what features other sites and applications had, what worked well and what would not be suitable for the needs of the average dog walker. These sites rely on content generated and reported by the users and are often referred to as crowd-sourced or user-generated content platforms.

Poowatch

https://poowatch.co.uk/

Poowatch is a Worcester City Council initiative for litter reporting. It is a map-based website application that allows users to log the location of dog poo sightings. Users click on the map where they have seen the poo and press submit. This is sent to the council, updating its hotspot information which can then inform the relevant enforcement officers.

Pros

- It is a cross platform web application which can be used on all devices, mobile, desktop or laptop.
- Users' location pinpointed via GPS
- Location hotspots available on the map for public to view
- Aesthetically pleasing with humorous poo logo.

Cons

- Reporting can be done by anyone without verification
- Geo locked to the Worcester City Council boundaries

Pee.ie

https://www.pee.ie/

Pee.ie is a website that has the location of public toilets in Ireland. Using Google Maps, it shows where they are located and clicking on them gives a pop up with information and directions to them. This is all user generated content and toilet location can be updated by users submitting a form.

Pros

- Easy to use and intuitive
- Embeds well with Google Maps

Cons

- User verification nonexistent
- Users need to provide GPS coordinates, no instructions where to get them
- Map and side menu overlap on mobile
- Although free, constant alert reminder to "Buy me a Coffee"

Love Clean Streets

https://lovecleanstreets.info/

Love Clean Streets is a commercial application that allows registered users to report environmental issues to their local authorities. They can do this via the website or Apple and Android applications.

Pros

- Automatically detects location
- Support ticket given for follow up of issue
- Verified account needed for reporting
- Easy to use
- Photo can be uploaded to support report

Cons

- Too many menu options

Crowd4access

https://crowd4access.insight-centre.org/

Crowd4Access is a user generated collaborative website that allows registered users to update a map of their area with the accessibility of footpaths. This is done through the OpenStreetMap platform.

Pros

- Participation leaderboard using gamification to encourage engagement of crowdsourced volunteers.
- Well documented instructions

Cons

- Even with detailed and easy to understand instructions it is a lengthy process
- Not suitable for mobile use
- Registration on website not available

Takeaways and conclusions

The following conclusions and takeaways were made having created a user and performing the environmental scan. The following could be incorporated into the design of the application:

Simplicity and Accessibility:

- Place an emphasis on an intuitive and simple interface. Pee.ie embeds well with Google Maps and offers straightforward navigation. Given Brian's preference for large, easy-to-read text, this simplicity can also extend to visual design and text size.
- The application should have a clear and visually appealing design like Poowatch. Humorous and engaging elements add to the overall user experience.

Cross-Platform Compatibility:

- Ensure that the application works seamlessly across various devices (mobiles, tablets, desktops), this is highlighted by Poowatch and Love Clean Streets, this will cater to users regardless of their preferred device.

Location-Based Features:

- Incorporate GPS functionality to automatically pinpoint the user's location, like that used by Poowatch and Love Clean Streets. This will help in providing exact directions to the nearest bin.
- Integrate map-based interaction where users can easily view and interact with the location of bins and reported dog waste, similar to the systems used by Poowatch and Crowd4Access.

User-Generated Content and Verification:

- Encourage crowd-sourced reporting to keep the database updated with new bin locations and dog waste hotspots. A verification system would need to be implemented to prevent false reporting as can be done on Poowatch or Pee.ie.

Streamlined User Experience:

- Avoid complex menus and excessive options, as seen in Love Clean Streets, which could overwhelm users. Develop a more streamlined experience minimizing steps where possible.

Engagement and Community Involvement:

- Integrate features that encourage user participation, such as gamification, seen in Crowd4Access. This could include rewards or recognition for frequent contributors, this may foster a community around keeping clean and accessible walking areas for dogs.

Functional and Aesthetic Integration:

- Like Pee.ie, consider integrating well with existing services such as Google Maps for reliable mapping data, but ensure that the application’s interface does not overlap or hinder usability on different device sizes.

Features were found that would work well in the application and would be necessary for it to function as intended. From this a list of needs were gathered to aid in the design of the application.

- Editing/reporting process: allow verified users to add icons to the map and save them in real-time. The reporting processes such as filling out forms, taking photos and giving descriptions were too convoluted, time consuming and unwieldy for a dog walker.
- Verification of users to perform editing/reporting: Require quick registration and login to encourage participation and to deter spam and vexatious reporting.
- GPS location functionality: allows users to locate themselves for accurate reporting and discovery of nearest bins.
- Mobile friendly: design on PC with a priority on responsive for mobile use, which is the intended audience.
- Design aesthetic: friendly, humorous, and accessible
- Crowdsourcing: users to be sourced from volunteers to provide a wide, accurate and well-maintained database of bin locations

The application would include but not be limited to these features.

Development Approach

The design and development of the application was influenced by the user profile and environmental scan. It was decided it would be a browser-based mobile-centric design and the reasons are below.

Design Rationale

Browser-based: Ensures wide cross-device compatibility without requiring software installation or updates. More suitable for a crowdsourcing application due to accessibility and ease of maintenance.

Mobile Optimization: Tailored specifically for mobile devices to leverage their strengths and provide an optimal user experience. Portability is essential for dog walkers who need easy access and use while on-the-go. It leverages the GPS capabilities of mobile devices to track the user's location and provide directions to nearby dog litter bins.

Accessibility Focus: To avoid a digital obstacle course, it was designed with accessibility needs as much as possible, focusing on easy to view and locate features. It incorporates intuitive and well known touch gestures and interactions that are familiar to mobile phone users. There are very few prerequisites for inexperienced users other than a basic familiarity with using a mobile browser and touchscreen gestures.

Design Approach

The application and its functions were mapped out to give an overview of what was involved, how it would look and the constituent parts that it comprised of. The remit of the design was to be simple and user-friendly. The application would have the basic functionality to provide and update a map with bin locations. Lightweight and direct was key to the design. Each page was mapped out as part of the development process in a sequence of sketched diagrams that illustrate how a user would interact with it. This step was crucial for visualizing the end product and ensuring that all necessary features were accounted for. The designs were kept fluid, allowing for adjustments based on emerging needs and challenges.

Design Layout

A sitemap was created for the application and how it would flow from the homepage on landing. This site architecture provided a broad overview and laid the foundation for figuring out what functionalities, software and systems would be needed to create it. It was hoped that any issues or peculiarities could be identified at an early stage and mitigated.

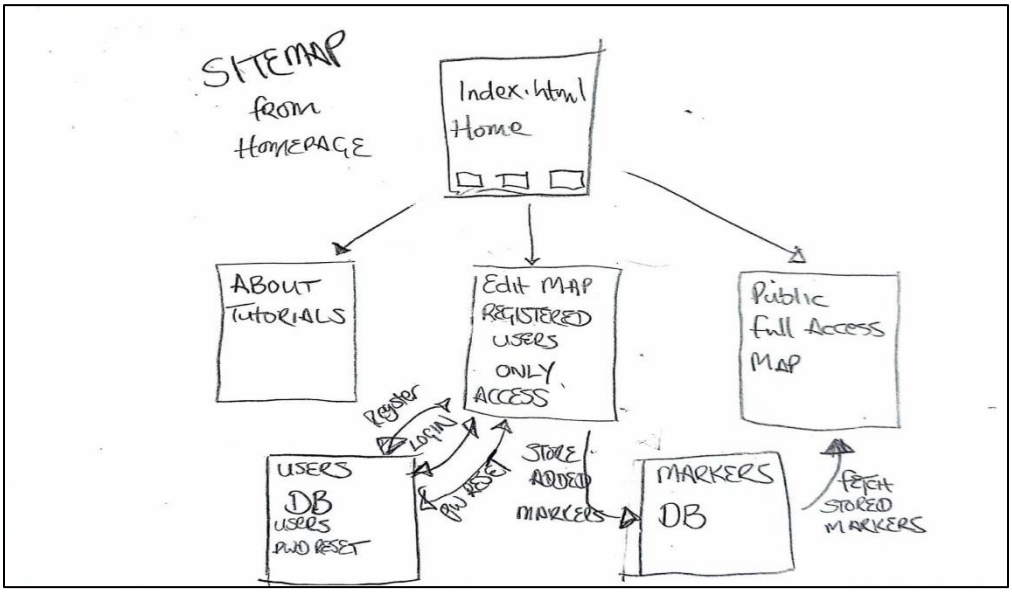


FIGURE 1: SITE ARCHITECTURE

The homepage would have three links leading from it.

- About: this would direct the user to a page that would have a blurb about the project and tutorial videos to instruct the user on how to use the site.
- Edit: this page would direct the user to a login page, allowing registered users to update the map.
- Map: this page would be fully visible to anyone who accessed it.

Access control would be needed for accessing the Edit Page. Users would need to login/register to gain access to edit the map. Their details would need to be stored in a secure database.

Once access is granted, they can edit the map, another database will need to be created to store their edits.

The Map page would display all the markers that had been added to the map. This would be gathered from the same database where the edits had been saved.

Map Design and Layout

How the two map pages would work was planned out in a wireframe. Ideally the same mapping software could be used for both with the basic features expected of an online map. In this instance the maps would need to have the following as standard

- GPS location services
- Ability to zoom in/out, pan using touchscreen gestures
- Different layers, street, and satellite views
- Ability to add/display markers to and from a database

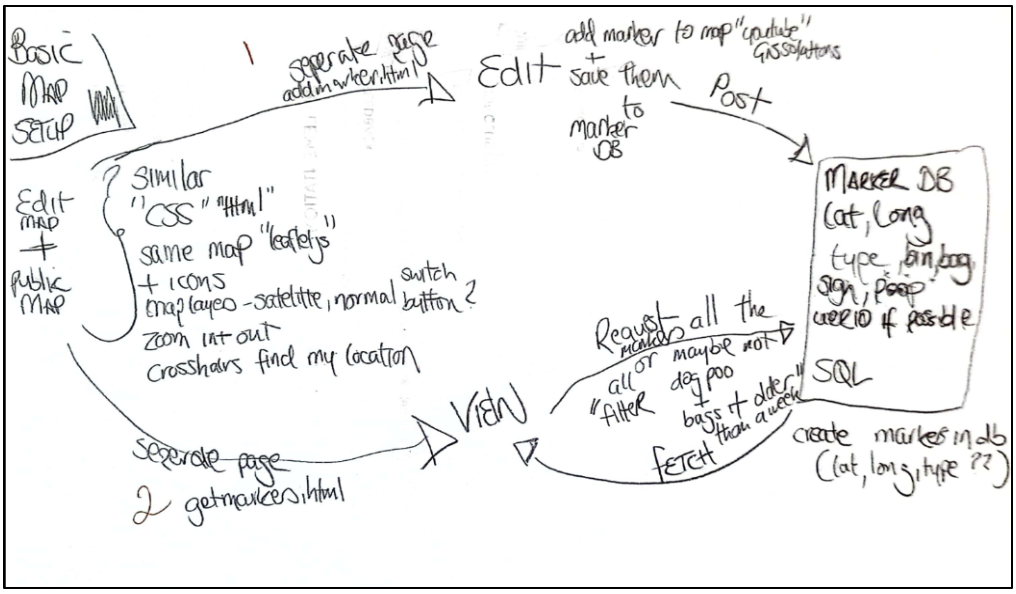


FIGURE 2: MAP DESIGN

The edit map would allow users to add to a markers database which would record the marker type and its latitude and longitude.

The view map would be able to pull all these saved edits from this markers database and display them on the map.

Database Design and Layout

Two databases were identified as necessary for the site.

- Users: would allow the registered details of the users to be saved. These would be used for account verification and access control to the editing map.
- Markers: would allow the storage and retrieval of editions to the map.

From this it was decided that the Edit map would be created first and the Markers database to save the additions made. The Map page would then be developed, firstly to display markers and then provide filtering and bin location services. The user registration/login section was developed when these two were deemed to be working as intended and with that a Users Database to store the details.

Software and Systems

Having identified the requirements for building the application the relevant software and systems needed to be identified. The rationale for selecting each is given with the following principles factored into their selection,

1. Ease of use and a low learning curve were prioritized in selecting software. Tools that could be used comfortably were chosen; this was aimed at saving time on downloading, installing, and figuring out its basic functionalities.
2. Familiarity: having previous experience with the software on the course was considered a bonus but not a necessity.
3. Free and open source: A preference was given to software that was free and open source, aiming for cost savings and an avoidance of providing credit card details.
4. Well supported community with readily available information online.

While the selected tools may not represent the absolute best available options, they effectively address the project's needs without introducing unnecessary complexity or the need for extensive evaluation or testing.

Website Host

Reclaim Hosting (Reclaim Hosting, n.d.) was selected both out of convenience, familiarity, and cost, having been used and paid for previously as part of the Digital Humanities Course. It is a one stop shop for hosting and managing an application like this. It manages the servers and security for the websites hosted on their platform. They handle the server infrastructure, ensuring that the hardware and software are properly maintained and updated. This includes managing the physical servers, network connectivity, server performance, and data storage.

A domain www.arsenalista.com had been registered and hosted with them previously to do assignments on WordPress and Omeka. It had been lying idle for a few months and they were able to do a top-level domain name change to the new URL at no extra charge. The speed of reply to questions on the viability of doing this and the process itself bode well for any future exchanges.

Reclaim uses cPanel, which is a widely used web hosting control panel that provides a graphical interface and automation tools designed to simplify the process of hosting a web application. Its graphical interface is straightforward and intuitive, making it accessible even for users with limited technical expertise.

It has a comprehensive suite of management tools available for managing domains, creating, and managing email accounts, uploading, and managing files, and backing up data. It provides popular content management systems such as WordPress and Omeka which can aid in deploying and managing the application.

cPanel provides interfaces for managing databases such as MySQL and PostgreSQL. Users can easily create, modify, and manage databases that can store and retrieve the expected user generated data.

Although not essential, the option of setting up email accounts associated with the domain name is particularly useful for user registration and communication.

Domain

The application would need a name that would be humorous, eye-catching, and indicative of its purpose. Poogle Maps was decided upon as a portmanteau of Google Maps, Poodle and Poop.

A domain name reflecting this https://www.pooglemaps.com was registered as part of the hosting process at a cost of $15. This investment offered several advantages over the alternative free service with a less manageable URL https://dev-pooglemaps.pantheonsite.io. It is easier to remember and lends a professional and polished image to the application. This will be useful in attracting and retaining users.

Website Building

WordPress was initially touted as the perfect solution for creating the application. It is a Content Management System (CMS) that can be easily installed on cPanel. It has an extensive library of themes and plugins. These pre-built themes and plugins were regarded as a potential aid in streamlining the development process and enhancing the application's functionality and appearance. However, several drawbacks associated with the WordPress platform were soon encountered which ultimately led to its abandonment.

WordPress was perceived as lacking intuitiveness, particularly for those with limited prior experience, the learning curve was deemed too high. The vast amount of information available online regarding WordPress proved overwhelming, making it challenging to locate relevant and reliable resources. While WordPress itself is free, many premium plugins and themes came with a cost.

Online research revealed it would be viable to create the entire website through coding from scratch, employing HTML, JavaScript, CSS, and PHP, and running it directly from the cPanel File Manager. This approach was adopted keeping to my initial principles, it was free, it utilized skills acquired on the course and there was a vast knowledge base online. Complete control could be wielded over every aspect of the site, from its structure and functionality to its appearance. To confirm this a basic html page called index.html was created, uploaded to the public_html section in File Manager and was on the internet at www.pooglemaps.com instantly.

Web Development Languages and Their Roles

- **HTML (Hypertext Markup Language):** HTML is the foundation of web pages. It defines the structure and layout of the content. HTML files are created with a .html extension.

- **CSS (Cascading Style Sheets):** CSS adds visual styling to the HTML elements. It controls fonts, colours, spacing, and layout. CSS files are created with a .css extension.

- **JavaScript:** JavaScript makes the website interactive. It allows the addition of dynamic behaviours, such as form validation, animations, and responsive features. JavaScript files are created with a .js extension.
- **PHP (Hypertext Preprocessor):** PHP is a server-side scripting language. It enables the performing of server-side tasks, interacting with databases, and generating dynamic content. PHP files have the .php extension.

Database Creation and Administration

cPanel has a section dedicated to database management. It utilizes phpMyAdmin which is a free, open-source web-based application that provides a user-friendly graphical interface for administering MySQL databases.

MySQL is an open-source relational database management system (RDBMS) that stores and manages data in a structured way. It allows the creation of databases, tables, and performing various operations like inserting, updating, and querying data.

The alternative to this introduced in the course was command line scripting which can offer flexibility and control for advanced users on specific and repetitive tasks. The GUI provided by phpMyAdmin is more beginner-friendly, accessible, and efficient for common database administration tasks. There is no need for direct server access or command line software.

Mapping

Leaflet was selected as it stood out as it is a lightweight and open-source alternative to mapping platforms like MapBox and Google Maps. Leaflet is entirely free; the others provide limited functionality on freemium plans. While lacking some advanced features out-of-the-box, Leaflet utilizes custom data sources and tile providers, giving it greater flexibility.

Leaflet offered an attractive choice for beginners seeking ease of integration and customization. While MapBox and Google Maps offer feature sets and proprietary APIs which would fulfil the remit of the application, these features can be created by the user. These features, although needing

a knowledge of JavaScript for creation, can be created by following online tutorials, videos and using the extensive library of free plugins.

A basic interactive map was created in HTML from the quick start section of the Leaflet website and was created and deployed online within 10 minutes (Agafonkin, 2024). This contrasted with the experience of needing to register, verify and signing in on MapBox and Google Maps. They were disregarded when credit card details were required for API key purchase.

Testing Software

XAMPP was chosen to ensure a thorough testing process for both the website and database management. This decision stemmed from the necessity to replicate the dynamics of a website and its associated databases when hosted on cPanel. XAMPP's comprehensive suite of tools, including the Apache web server, MySQL database, and phpMyAdmin, perfectly mirrors the functionalities encountered in a cPanel-hosted environment.

XAMPP emulates the conditions and performance of a live server setup. This would facilitate the necessary testing scenarios, such as website testing, database storage, creation, and retrieval. This selection enables the preemptive identification and the addressing of any potential issues or discrepancies, ensuring minimal disruptions or surprises when deploying in the real world.

XAMPP was downloaded, installed, and deployed with minimal fuss, it was deemed unnecessary to find any alternatives.

Artwork and Design

The following were used to create the icons, backgrounds, and logos on the website.

GIMP: is a free, powerful, and versatile image editor for creating icons, logos, and editing pictures with professional-level tools. This was already installed and an understanding of how it worked was already known.

Leonardo.ai: is an AI-powered platform focused on image generation, through user prompting. Its freemium level was sufficient for this project (Leonardo.ai, n.d.).

Samsung Image Clipper: is a Samsung Galaxy tool that allows objects or people to be extracted from images. Free to use and pre-installed as standard.

MS Paint: is a Windows program offering basic image editing tools for drawing, simple modification of photos, and creating straightforward 2d graphics.

Leaflet Basics

The basics of how Leaflet maps worked needed to be understood. For this project's purpose, there was a need to create an interactive client-side environment for the creation and display of user-generated icons.

A needs analysis showed a requirement for the following, a simple to use interactive map. Users would be able to select and add customized icons to the map and save them to a database. Subsequently a separate map view would display these saved icons.

Leaflet's comprehensive knowledge base, including tutorials, documentation, and extensive plugins, is supported by a sizable and active community. (Agafonkin, 2024).

Leaflet is an open-source JavaScript library that allows the creation of the map on a web page. It does not contain any map data or imagery. This needs to be provided from an external source.

This framework allows it to render and manipulate map data from other sources. Tiles are the most common source for rendering maps. Map tiles are small raster images that represent a fixed geographical area at a particular zoom level. They are loaded from a server and assembled in the user's browser, like how a large jigsaw is broken down into smaller, more manageable pieces. When the user pans and zooms Leaflet calculates the required tiles and fetches them from a tile server such as OpenStreetMap or Mapbox. This gives the illusion of a single continuous map. Map tiles do not have the coordinates embedded in them; Leaflet will infer their coordinates from the source directory they come from.

Creating The Map

A base map was created from the tutorial section of the Leaflet website (Agafonkin, 2024). This would be the template for creating the ‘Edit’ and ‘Map’ pages. Using Notepad ++ as the editor it was a matter of copying Leaflet files into the HMTL page.

- **Leaflet CSS File:** contains the styles and visual appearance of the map and its elements such as tiles, markers, controls, and popups.

```
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
      integrity="sha256-p4NxAoJBhIIN+hmNHrzRCf9tD/miZyoHS5obTRR9BMY="
      crossorigin=""/>
```

FIGURE 3: LEAFLET CSS FILE

- **Leaflet JavaScript File:** contains the core functionality of the Leaflet library. This provides the necessary methods, classes, and APIs for interacting, creating, and editing the map. It includes code for loading tiles, adding markers and user interaction such as panning and zooming.

```
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"
        integrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kNiyxNV1lvTlZBo="
        crossorigin=""></script>
```

FIGURE 4: LEAFLET JAVASCRIPT FILE

- **Map div:** this creates the area on the screen where the tile layer will be displayed.

```
<div id="map"></div>
```

FIGURE 5: LEAFLET DIV ELEMENT

- **Initialization:** this creates an instance of the map object and the container element ‘map’ that it is to be rendered in.The initial view is then set, the coordinates for Cork City (51.8985, -8.4756) have been included and a zoom level of 13.The base imagery tile layer is then sourced from the OpenStreetMap tile service URL. The .addTo(map) method adds this tile layer to the map instance making it visible on the webpage.

```
var map = L.map('map').setView([51.8985, -8.4756], 13); // Ensure you add the tile layer here
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: 'Map data &copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',
    maxZoom: 18
}).addTo(map);
```

FIGURE 6: CREATING THE MAP INSTANCE

After saving the HTML file in the **htdocs** directory of the XAMPP server and running it locally, a map centered over Cork City was successfully displayed.

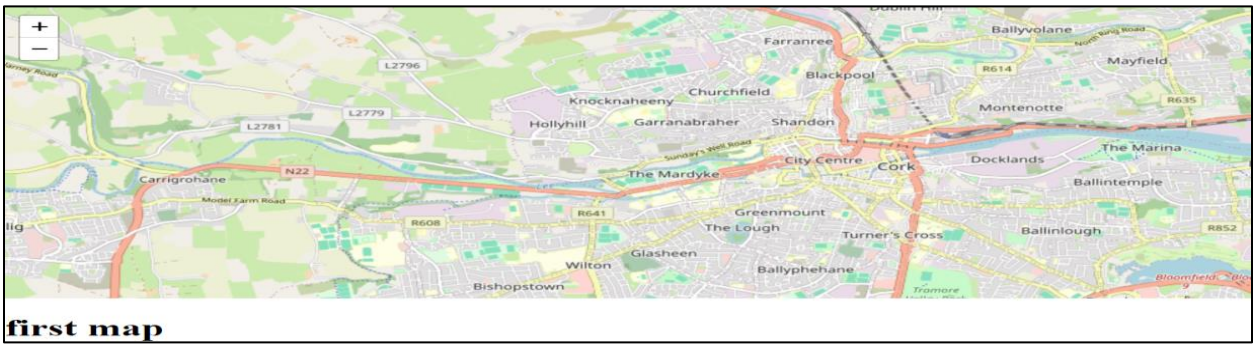


FIGURE 7: FIRST LEAFLET MAP CREATED

- **Locate me:** The ability to locate the users location is one of the key features of the application. A locate me button was created and placed under the zoom buttons in the map container. Using an event listener for the button to be clicked it calls the locate method provided in the Leaflet library. (Agafonkin , 2024) The button did not appear on the screen, it was discovered that its Z-index in CSS which determines how it is stacked amongst the elements on the screen (Stack Overflow, 2018). The index was increased to 1000 and revealed the button. Clicking on it worked and was able to locate me.

- **Satellite view:** To improve the accessibility of the maps a satellite view was added for those who would find it difficult to read the standard OSM layer provided. This works on that in Leaflet every time an item is added to the map using the addTo(map) method it creates a new layer. (Stack Overflow, 2016)

FIGURE 8: MAP WITH SATELLITE VIEW AND LOCATE ME BUTTON ADDED

Database Management

FIGURE 9: PHPMYADMIN LOGO

Reclaim Hosting provides a database section within cPanel which includes access to phpMyAdmin, this is a popular web-based interface for managing MySQL databases. The phpMyAdmin section is pre-configured and ready to use out-of-the-box, allowing users to interact with their MySQL databases and tables effortlessly.

For this project's purposes, the following elements of the phpMyAdmin section were used.

Database Selection: Upon accessing phpMyAdmin, users are presented with a list of MySQL databases available within their hosting account. They can select the desired database they wish to work with.

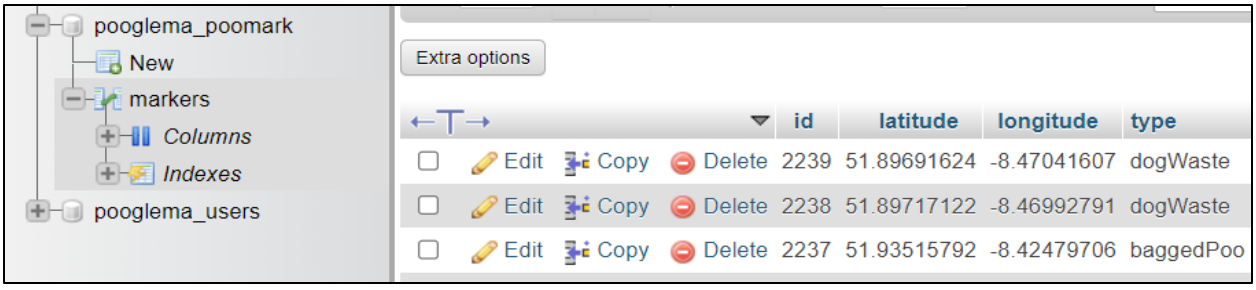


FIGURE 10: CREATE USERS TABLE

Table Management: Within the selected database, users can view and manage all existing tables. They can create new tables, define their structure (columns, data types, etc.), insert data, modify data, and perform various other table operations through an intuitive interface

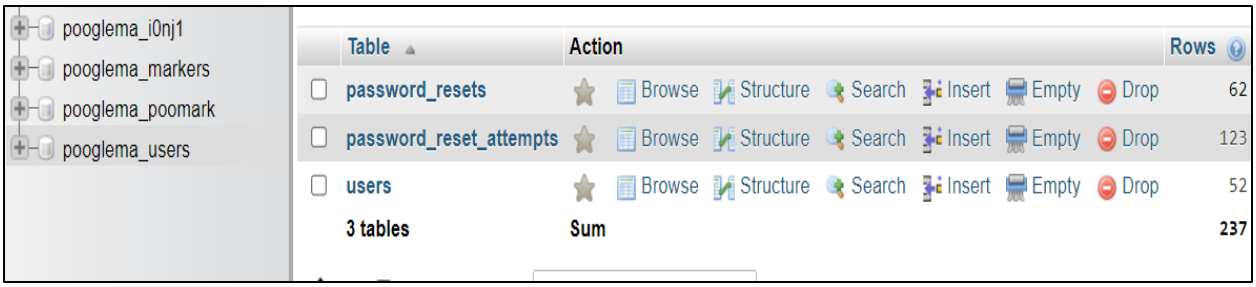


FIGURE 11: VIEW TABLES IN MYPHPADMIN

Data Import/Export: Users can import data into their databases from external sources or export existing data for backup or migration purposes. phpMyAdmin supports various import and export formats. (e.g., SQL scripts, JSON, CSV files)

Database Backups: The phpMyAdmin section often includes functionality for creating and managing database backups. Users can create backups of their databases, schedule automatic backups, and restore databases from existing backup files.

SQL Query Execution: phpMyAdmin provides a SQL query interface where users can execute custom SQL queries directly against their databases. This feature is useful for advanced operations, such as complex data manipulations or database schema modifications. This simplifies the process of managing MySQL databases and tables, eliminating the need to work with command-line tools

Two databases were set up to separate user data from map data rather than having two tables in one database. The rationale for this was to allow for the complete separation of personal identifying data from the mapping data. This would allow easier organization and maintain a modicum of order if the site needed to scale up and add more tables.

Database Creation:

Users Database

A database was created to store the details of the users on registration. This will store their first name, surname, and the password they provided. When they subsequently login to the system these details will be searched for and verified to allow access to the Edit map.

SQL queries were used to create both the databases and the tables needed.

The following creates a database called pooglema_users, it is case sensitive and cannot start with a number or a space.

```
CREATE DATABASE pooglema_users;
```

FIGURE 12: CREATE USERS DATABASE

Users Table

A users table was then created and placed inside the pooglema_users database. How the columns are declared is crucial as it determines the data types, constraints, and characteristics of the data that can be stored in each. The id column will automatically create an incrementing unique value of type INT (integer) enforced by the addition of the PRIMARY KEY constraint. VARCHAR is a variable-length character string with a maximum length of 50 characters or 255 characters. NOT NULL declares them as required fields. The email column is declared UNIQUE NOT NULL, ensuring that each email address stored in the database is unique. This prevents multiple users from registering with the same email address, maintaining data integrity, and facilitating effective communication and authentication processes.

```
CREATE TABLE pooglema_users.users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(50) NOT NULL,
  surname VARCHAR(50) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL
);
```

FIGURE 13: SQL QUERY FOR TABLE CREATION

Markers Database

A second database is created to store details provided by the user when editing the map. This will store non-identifying information.

```
CREATE DATABASE pooglema_poomarker;
```

FIGURE 14: CREATE MARKERS DATABASE

The markers table was created to store geographical markers, each associated with a specific type, bin, dog waste, bagged dog waste and dog litter sign. The table is designed within a database to ensure that each marker is uniquely identifiable and precisely located via latitude and longitude coordinates.

```
CREATE TABLE markers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    latitude DECIMAL(10, 6) NOT NULL,
    longitude DECIMAL(10, 6) NOT NULL,
    type VARCHAR(50)
);
```

FIGURE 15:SQL QUERY CREATE TABLE MARKERS

Edit Page

This section covers the actions that occur in the Edit Map page and how it was created. The focus was on how the users will be able to select and add custom icons to the screen, move and remove them if needed. A mechanism was then put in place to save these edits to a database.

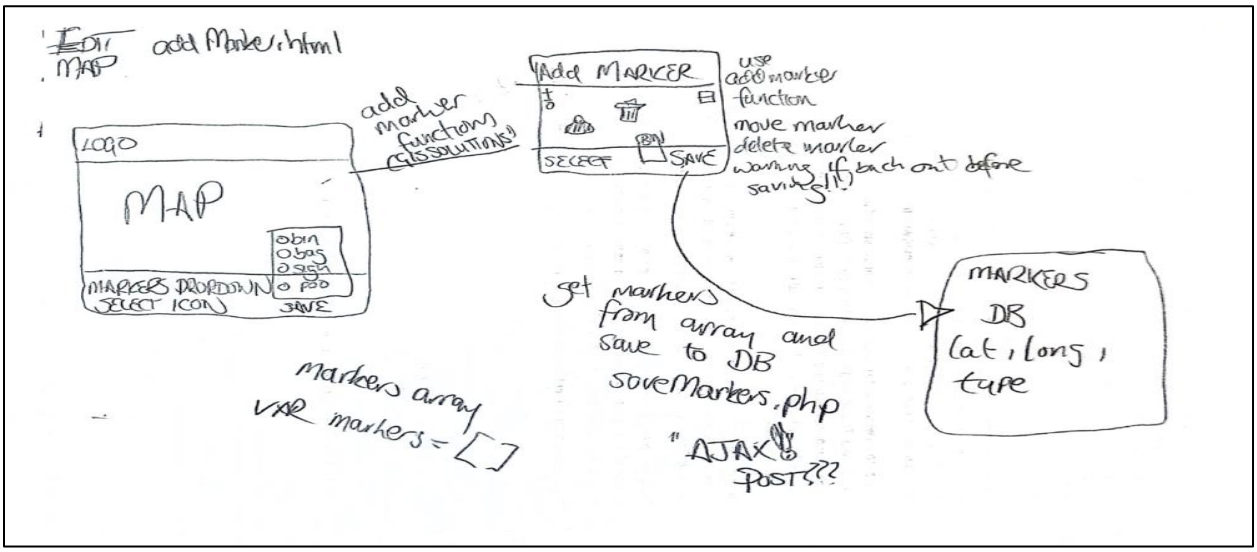


FIGURE 16: OVERVIEW OF MAP EDITING

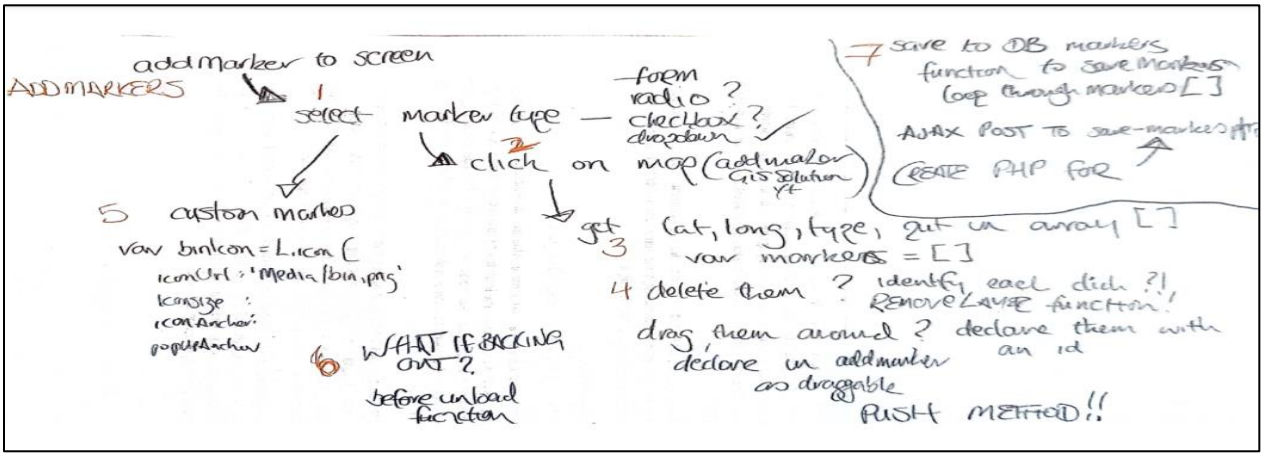


FIGURE 17: ADDING ICONS PROCESS

These designs allowed this process to be created emphasizing that the data will be saved and sent to a database.

The following instructions were inserted in a HTML file called addMarker.html (subsequently updated to addMarker.php) . The base of which contained the Leaflet maps template created in Leaflet Basics (see page 20).

Adding Icons To The Map

The next step was to add markers to the map (GIS SOLUTIONS, 2022a). Leaflet provides code to create a generic blue pin icon when clicking on the map (Agafonkin, 2024).

```
function addMarker(e) {

    // Create a generic Leaflet marker

    var marker = L.marker(e.latlng, {draggable:
true }).addTo(map);

    marker.bindPopup("Marker added at " +
e.latlng.toString()).openPopup();

}

map.on('click', addMarker);
```

FIGURE 18: ADDING GENERIC MARKER TO THE MAP

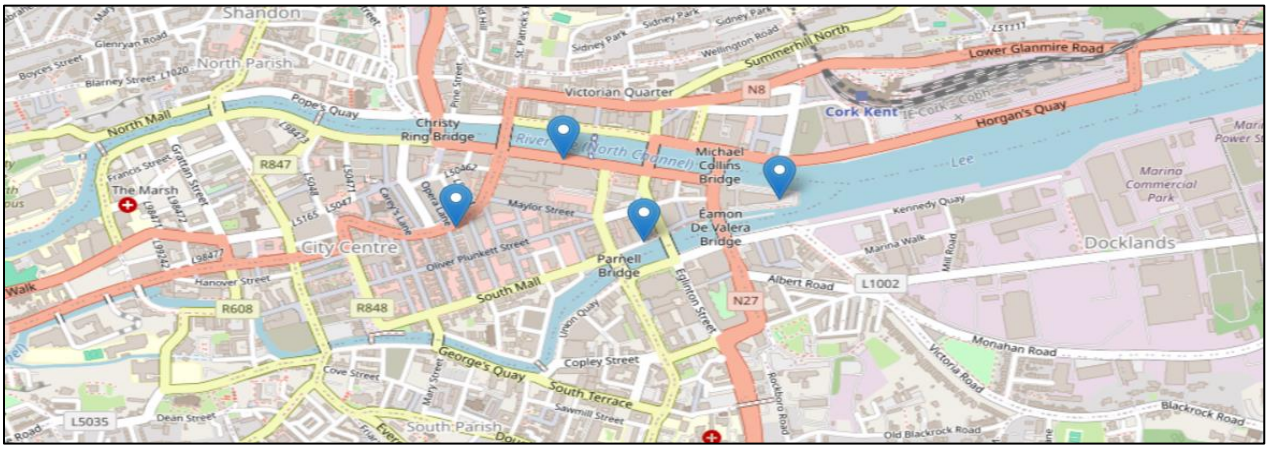


FIGURE19: MAP WITH ADDED ICONS

The icons were added to the map utilizing the on method which was listening for the click event which triggered the addMarker function. These icons are also draggable.

Create Customized Icons

Customized icons to depict the locations of bins, dog waste, bagged dog waste and litter signs would need to be added to the map. A bin icon was used to test this, a png file saved to the htdoc folder would be made available.

```
var binIcon = L.icon({
  iconUrl: 'http://localhost/bin.png'',
  iconSize: [32, 32],
  iconAnchor: [16, 32],
  popupAnchor: [0, -32]
});
```

FIGURE 20: CODE TO CREATE CUSTOM ICON

The addmarker function would need to be modified to include this

```
var marker = L.marker(e.latlng, {icon: binIcon, draggable: true}).addTo(map);
```

FIGURE 21: BINICON ADDED TO ADDMARKER FUNCTION

FIGURE 22: CUSTOM ICON ADDED TO MAP

Delete Icon From The Map

If the icons were placed in the incorrect spot they would need to be deleted. (GIS SOLUTIONS, 2022b). The addMarker function was amended to include a click event handler and a remove method for that event e.

```
var marker = L.marker(e.latlng, {icon: binIcon, draggable: true}).addTo(map).on('click',e=>e.target.remove());
```

FIGURE 23: REMOVE ICON ON CLICK

The map had three other icons to include and needed a way of selecting them. For mobile optimization, a dropdown menu was used. Custom icons were created for dog poo, dog litter signs and bagged poo.

A select div element was added to the footer of the page so they could be selected when they were encountered.The addMarker function was updated to include these conditions. Another

```
if (markerType === 'dogWaste') {
  marker = L.marker(e.latlng, {icon: dogWasteIcon, draggable: true}).addTo(map);
} else if (markerType === 'bin') {
  marker = L.marker(e.latlng, {icon: binIcon, draggable: true}).addTo(map);
} else if (markerType === 'dogLitterSign') {
  marker = L.marker(e.latlng, {icon: dogLitterSignIcon, draggable: true}).addTo(map);
} else if (markerType === 'baggedPoo') {
  marker = L.marker(e.latlng, {icon: baggedPooIcon, draggable: true}).addTo(map);
}
```

FIGURE 24: UPDATED ADDMARKER FUNCTION INCLUDING ALL TYPES

Saving Markers To The Database

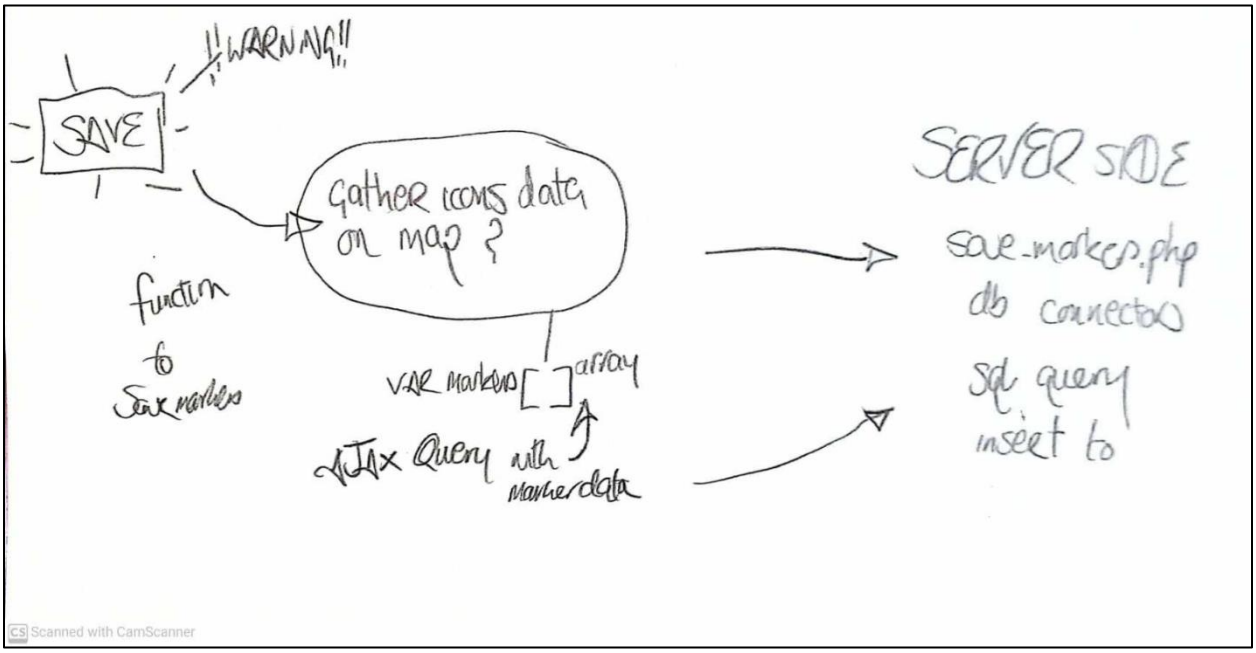


FIGURE 25: SAVE MARKERS DESIGN

It was planned that the process to save the markers to the database would involve both client-side and server-side scripting to manage data flow securely and efficiently.

Client-side Processing

User Interaction: a ‘Save Markers’ button was created in the footer section of the page. It would initiate the ‘saveMarkers’ process.

Confirmation: When the button is pressed, the ‘saveMarkers’ function is triggered, which includes a confirmation step. An alert is displayed asking the user to confirm if they wanted to save the markers or not. This helps prevent unintentional submissions.

Data Preparation: If the user confirms, the function then iterates through the markers array, this contains all the markers currently displayed on the map. Each marker's data, including its geographic coordinates (latitude and longitude) and its type (type) are collected and pushed into an array ‘markerData’

```
    markers.forEach(function(marker) {
      markerData.push({
        latitude: marker.latlng.lat,
        longitude: marker.latlng.lng,
        type: marker.type
      });
    });
```

FIGURE 26: ITERATION THROUG MARKERS DATA

AJAX POST Request: This ‘markerData’ array is then sent to ‘save-markers.php’ on the server via an AJAX POST request. AJAX allows this data transmission to happen in the background (asynchronously) without needing to reload the web page. The marker data needs to be converted to a JSON string so that it can be easily parsed and processed on the server side.

```
$.ajax({
  type: 'POST',
  url: 'save-markers.php',
  data: { markers: JSON.stringify(markerData) },
  success: function(response) {
    alert('Markers saved successfully!');
  },
  error: function(xhr, status, error) {
    alert('Error saving markers.');
  }
});
```

FIGURE 27: AJAX POST REQUEST

Server-side Processing

Connection: (XHTML, JavaScript, AJAX, PHP, n.d.), the connection to the database on the server side was made using the provided login details.

Processing Data: It begins with the POST request and starts decoding the JSON string back into PHP data structures.

Database interaction: SQL statements are prepared to check if a marker exists and to insert the new one if it does not (PHP GROUP, 2009)

Response: After processing, the server sends a response back to the client, which could be a success message or an error message, depending on the outcome of the database operations.

Errors:

This was unsuccessful on its first attempt; the database was checked, and nothing happened. There were no obvious errors, so these were added to the 'save-markers.php' file (Mendez, 2023).

```
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
```

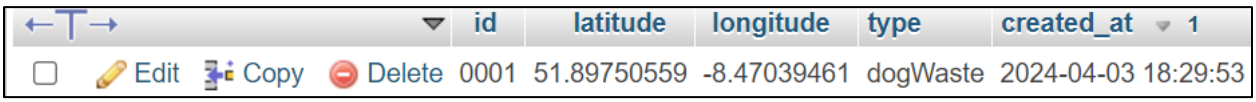
FIGURE 28: ERROR REPORTING CODES

Markers were again added to the map and saved. The database did not update again. The issue was discovered in the error log.

```
[03-Apr-2024 18:24:57 UTC] PHP Fatal error:  Uncaught ArgumentCountError: The number of elements in the type definition string must match the number of bind variables in /home/pooglema/public_html/save-markers.php:44
Stack trace:#0 /home/pooglema/public_html/save-markers.php(44): mysqli_stmt->bind_param('dsi', NULL, NULL, NULL, NULL)
#1 {main}  thrown in /home/pooglema/public_html/save-markers.php on line 44
```

FIGURE 29: ERROR LOG

From this it was seen that there was a mismatch between the number of placeholders in the prepared SQL statement and the number of variables being bound to them. It was pinpointed to line 44 and a d had been left out of the bind parameter call. This was rectified and the process repeated this time the details populated as expected in the database.



		id	latitude	longitude	type	created_at
☐	Edit Copy Delete	0001	51.89750559	-8.47039461	dogWaste	2024-04-03 18:29:53

FIGURE 30: DATABASE SUCCESSFULLY POPULATED AFTER ERROR CORRECTION

Map Page

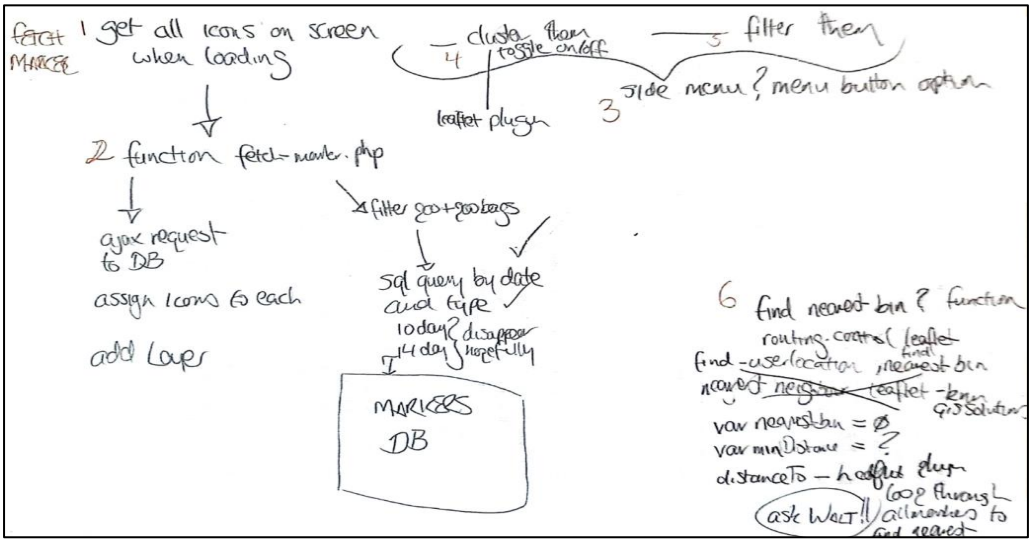


FIGURE 31: MAP PAGE PROCESS

This page was created to display all markers saved in the 'Markers' database. Marker icons will appear on the screen upon loading. To ensure the accuracy of the data, markers indicating dog waste older than 10 days and bagged dog waste older than 14 days will not be requested, as it can be assumed that it would not reflect current conditions.

The user could then access a side menu that allowed them different views of the data. These views could be filtered by icon type using checkboxes. An option would be available to toggle the clustering of the icons to allow quicker navigation as icons would not load individually but be clustered into groups of icons. The main functionality would be the “find nearest bin” option which allows the system to locate the nearest bin and provide directions to it.

The side menu was created with the ability to be closed if user clicked outside it.

```
window.addEventListener('click', function(event) {
  if (!slideMenu.contains(event.target) && slideMenu.classList.contains('active')) {
    slideMenu.classList.remove('active');
  }
});
```

FIGURE 32: CLOSE MENU EVENT

Fetch And Display Markers Function

This client facing page was created in the ‘fetchMarker.html’ file. This file contains the base map html template created previously. This would send a request to a file called ‘fetch-markers.php’ that would interact with the ‘markers’ database.

A Fetch Markers function was created to “fetch” this data. An Ajax request is sent to ‘fetch-markers.php’ using the GET method and expects a response in JSON format (Stack Overflow, 2017).

```
$(document).ready(function() {
    $.ajax({
        url: 'fetch-markers.php',
        type: 'GET',
        dataType: 'json',
        success: function(markers) {
```

FIGURE 33: AJAX REQUEST

This function then iterates through the received markers and assigns it an icon depending on type. A new layer is then added to the map. This then populated the screen with the relevant markers.

```
markers.forEach(marker => {
    var icon;
    switch (marker.type) {
        case 'dogWaste':
            icon = dogWasteIcon;
            break;
        case 'bin':
            icon = binIcon;
            break;
        case 'dogLitterSign':
            icon = dogLitterSignIcon;
            break;
        case 'baggedPoo':
            icon = baggedPooIcon;
            break;
        default:
            icon = L.Icon.Default();
    }
    var newMarker = L.marker([marker.latitude, marker.longitude], { icon: icon });
    newMarker.markerType = marker.type;
    allMarkers.addLayer(newMarker);
});
map.addLayer(allMarkers);
```

FIGURE 34: AJAX FUNCTION TO ADD RECIEVED ICONS TO MAP

The PHP file ‘fetch-markers.php’ was created. A connection is established to the MySQL database using the requisite server details, username, password, and database name. A PHP interval class was created to calculate intervals based on the current date and that of dog waste older than 14 days, and bagged dog waste 10 days and older, (ORACLE, 2024).

```php
// Current date
$today = new DateTime();
// 10 day timeframe
$intervalDogWaste = new DateInterval('P10D');
$dogWasteDate = $today->sub($intervalDogWaste)->format('Y-m-d');
// 14 day timeframe
$intervalBaggedPoo = new DateInterval('P14D');
$baggedPooDate = $today->sub($intervalBaggedPoo)->format('Y-m-d');
```

FIGURE 35: PHP INTERVAL CLASSES

These dates were then passed to the SQL query that would search the ‘Markers’ database for everything, with the condition that if it was dogWaste it had to be created in the last 14 days and 10 days if it was baggedPoo.

```
$sql = "SELECT * FROM markers WHERE

(type != 'dogWaste' OR (type = 'dogWaste' AND created_at >
'$dogWasteDate'))

AND (type != 'baggedPoo' OR (type = 'baggedPoo' AND created_at >
'$baggedPooDate'))";
```

FIGURE 36:SQL QUERY WITH PHP INTERVAL CLASS INCLUDED

An empty array ‘$markers’ is populated with the requested data, encoded as JSON and sent back to the ‘fetchMarkers.html’ page.

Apply Filter Function

A checkbox div was created to initiate the filtering of received icons by type. It was placed on the side menu.

The function for applying a filter was created similarly to the fetch marker's function.

Toggle Clustering Function

This was created by adding a CSS link to the Leaflet.markercluster plugin in the head of the page and creating a function using the code supplied (Nikulski, 2021).

```
function toggleClustering() {
  useClustering = document.getElementById('toggleClustering').checked;
  var newGroup = useClustering ? L.markerClusterGroup() : L.layerGroup();
  allMarkers.eachLayer(function (marker) {
    newGroup.addLayer(marker);
  });
  map.removeLayer(allMarkers);
  allMarkers = newGroup;
  map.addLayer(allMarkers);
}
```

FIGURE 37: CLUSTERING TOGGLE FUNCTION

Find Nearest Bin Function

A function needed to be created to calculate where the nearest bin was located. Leaflet has similar functions/plugins to find the distance between two points or a waypoint and current position. This function was created using Leaflets distance method iterating through each “markerType ===bin” until that with the lowest value was found. This would trigger the routing control plugin and create a route to it.

This worked as intended but the routing machine instructions produced a new pop up each time the function was called. This caused the screen on a mobile to become easily obscured.

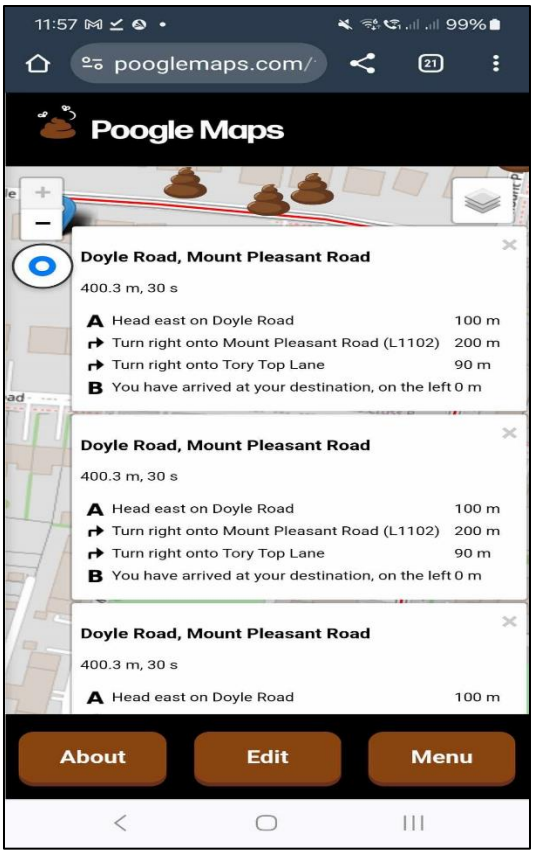


FIGURE 38: ROUTING INSTRUCTIONS ISSUE

The solution was to store instances of routingControl in a variable and check it to see if it was populated each time the bin finding button was pressed. If it was present, it would be removed using the Leaflets removeControl method.[1]

```
  if (routingControl != null) {

        map.removeControl(routingControl); // Remove the
previous route from the map

        routingControl = null;}
```

[2]FIGURE 39: REMOVE CONTROL METHOD

[1] The design and development of the MAP page was facilitated by the guidance and support received in understanding and demonstration on how to work through the coding logic by Leanne Walsh and Walter Wynne.

Login, Registration, and Password Management

It was decided after the environmental scan that some form of access restrictions would need to be added to prevent people from adding erroneous additions to the map. A simple registration, verification and login system was designed to deter those with only a passing interest from editing.

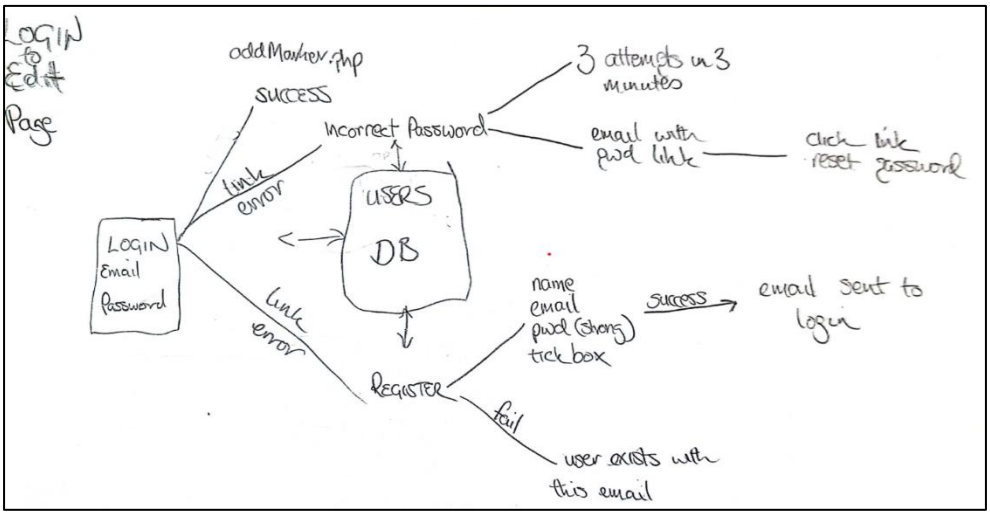


FIGURE 40:ACCESS CONTROL PROCESS MAP

The following would be needed to control access to the Edit page. A login system was needed, this would then link to a registration page and a reset password page. (www.w3schools.com, n.d.)

A users database would need to be set up that could store the users name, surname, email and password when first registering. These details would be subsequently checked to verify users on login (mmtuts, 2018b).

A users database was created previously in Database Management(see page 30).

A user would need to register their details, name, email address and provide a password and agree to the terms of the project before creating an account.

Registration Page

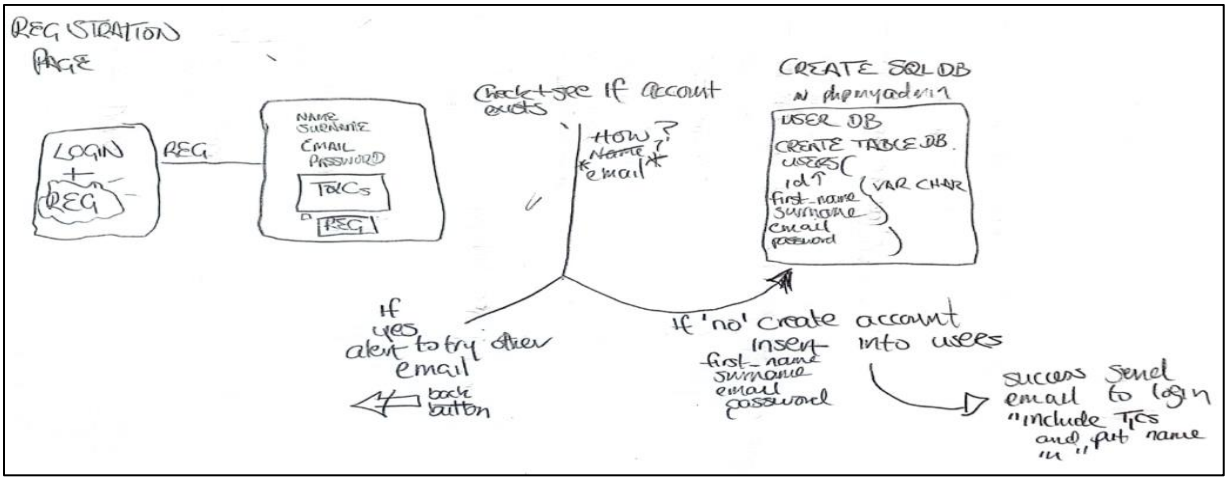


FIGURE 41: REGISTRATION PAGE PROCESS MAP

A register.html page was created (https://www.pooglemaps.com/register.html). A form div element was added with the details First Name, Surname, Email Address and Password as requirements. A container div element that would contain the consent for participation agreement in it was created. A checkbox element was also included that would need to be checked to confirm that the user had read and agreed to the participation terms.

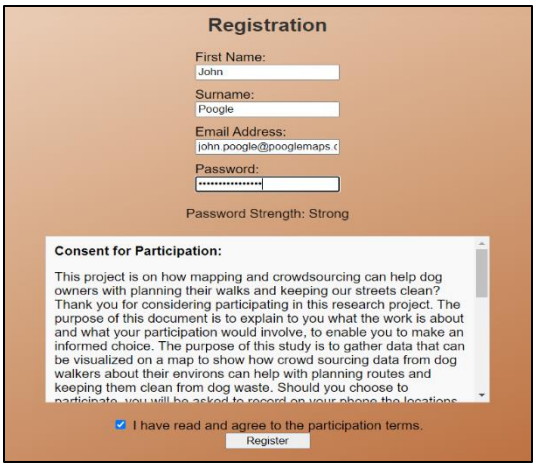


FIGURE 42: REGISTRATION FORM

As an extra layer of security a password checker zxcvbn was added, this is a JavaScript library designed to help evaluate the strength of passwords in a more realistic way than traditional password strength checkers. The register button will not appear until a requisite strength password is created and the checkbox ticked. (Wook, 2023).

A server-side PHP ‘register.php’ would also need to be created to sanitize, check for preexisting email incidences from the form submission and for inserting the details into the users database.

Registration php

The registration form element had been set up with the POST method and this script will check this first. Database connections and login details are checked.

A SQL statement was created to check the database to see if the email existed already and an appropriate response was issued if it did. A password function was also used to hash the password provided. A prepared statement to insert first name, surname, email and hashed password was created.

```
if ($existingUser) {
            echo "A user with this email address already exists. Please use a different email.";
        } else {
            $passwordHash = password_hash($password, PASSWORD_DEFAULT);
            $stmt = $pdo->prepare("INSERT INTO users (first_name, surname, email, password_hash) VALUES (?, ?, ?, ?)");
```

FIGURE 43: PREPARED INSERT STATEMENT

If this was successful a successful registration message would greet the user and ask them to check their email and login from there. The participation consent form with the users name included is sent to them for their records. This was tested and the registration was successful.

Login Page

To direct users away from the Edit page and require them to login with their user details. A function was created, placed on each page with a link to the Edit page. When the EDIT button was pressed the function was called and sent user to ‘login.html’.

```
function redirectToUpdateMap() {
    const isLoggedIn = localStorage.getItem("isLoggedIn");
    if (!isLoggedIn) {
        window.location.href = "login.html";
    } else {
        window.location.href = "addMarker.php";
    }
```

FIGURE 44: REDIRECT FROM EDIT TO LOGIN

A basic login page with email and password fields was created. This is connected to ‘login.php’.

Login php

The relevant connecition details are included along with checks to see if it is a post method. The email and password are taken from the form and a SQL statement is prepared to retrieve the users information id, password_hash from the users table. If a user is found the provided password is checked against the stored hashed password. If it is successful the user is directed to the ‘addMarker.php’ page. Otherwise, an error occurs asking user to register or reset the password.

```php
$stmt = $conn->prepare("SELECT id, password_hash, is_verified FROM users WHERE email = ?");
$stmt->bind_param("s", $email);
$stmt->execute();
$result = $stmt->get_result();

if ($result->num_rows > 0) {
    $user = $result->fetch_assoc();

    if (password_verify($password, $user['password_hash'])) {
        if ($user['is_verified']) {
            $_SESSION['user_id'] = $user['id'];
            $_SESSION['is_verified'] = true;
            header("Location: addMarker.php");
            exit();
        } else {
            $error_message = "Your email is not yet verified. Please check your email for the verification link.";
        }
    } else {
        $error_message = "Invalid email or password.";
```

FIGURE 45: LOGIN.PHP USER VERIFICATION PROCESS

Reset Password

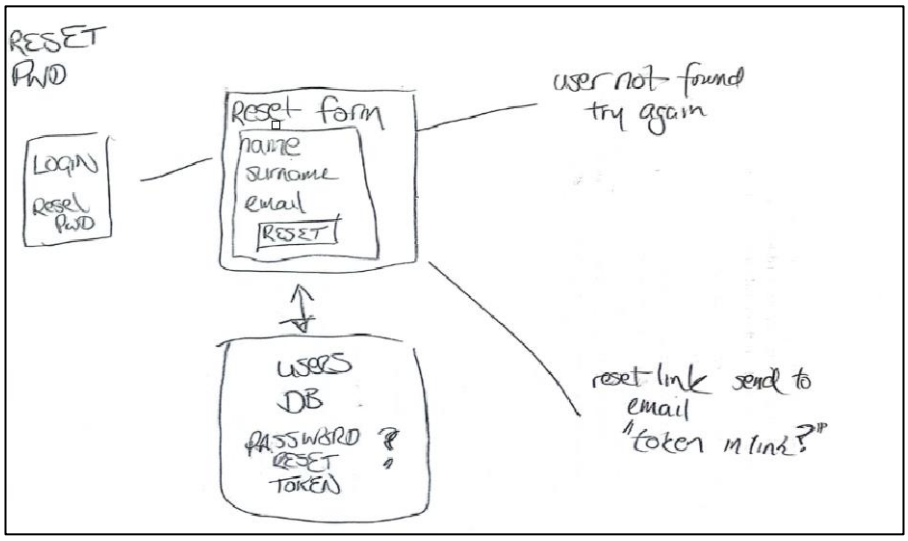


FIGURE 46: RESET PASSWORD PROCESS MAP

A self-service reset password facility was made available for the users in case they had forgotten theirs or wished to change it.

Following best practice, the process was broken into two parts (www.w3schools.com, n.d.).

- A password request is sent, and a reset token emailed to the user.
- The password reset token is verified and password reset allowed.

A new table ‘password resets’ in the ‘users’ database was set up with the fields user_id, token and expiration.

```
CREATE TABLE `password_resets` (

  `id` int NOT NULL,

  `user_id` int NOT NULL,

  `token` varchar(255) NOT NULL,

  `expiration` datetime NOT NULL);
```

FIGURE 47: CREATE PASSWORD RESETS TABLE

forgot_password.php

A password reset link was attached to the login page and directed users to a form at 'forgot_password.php'. Users are requested to enter their first name, surname and email address.

```
if ($user) {
    $token = uniqid('pwreset_', true);
    $expiration = date('Y-m-d H:i:s', strtotime('+1 hour'));
    $sql = "INSERT INTO password_resets (user_id, token, expiration) VALUES (?, ?, ?)";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([$user['id'], $token, $expiration]);
}
```

FIGURE 48: FORGOT_PASSWORD PHP SCRIPT

The script will confirm the user is registered, create a unique password reset token and place an expiration of one hour on it. The SQL query will then insert a new row into the 'password_resets' table.

This triggers an email to be sent to the user with the password reset token contained as part of a link. Part of this link is the process 'reset_password.php'

reset_password.php

https://www.pooglemaps.com/reset_password.php?token=pwreset_662971d3c205f2.06305846

FIGURE 49: EMAIL CONTAINING PASSWORD RESET LINK

Clicking on the link initiates the 'reset_password.php' script. The token value is checked against that stored in the 'password_resets' table and that it has not expired.

```
(SELECT * FROM password_resets WHERE token = ? AND expiration >
NOW())
```

FIGURE 50: SQL QUERY OF PASSWORD RESET TOKEN VALIDITY

If the token is valid a submit password form appears and starts the reset password process. This updates the users password with a new rehashed password

About Page

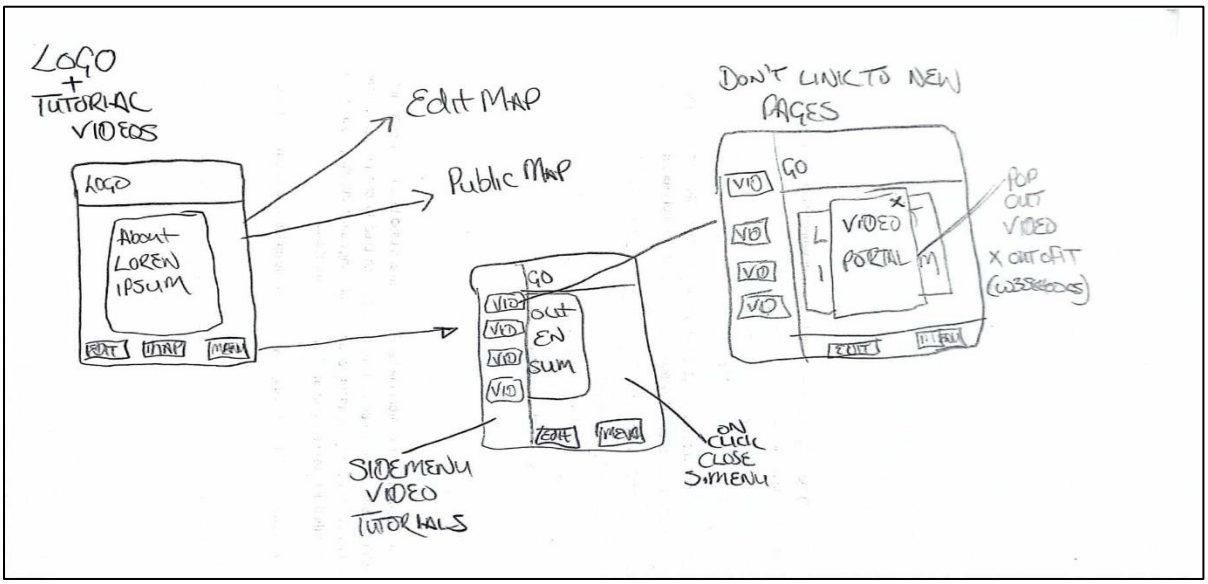


FIGURE 51: ABOUT PAGE PROCESS MAP

This page was created with the purpose of giving the user the context of what the application was for and how to use it.

A container element was imbedded in the body of the page with a recap of the project and its aims. A side menu replicated from the 'Map' page was included, here buttons were added for accessing pop up video tutorials.

Screen recordings of the main processes were performed and saved as mp4 files; Howto Register, How to Reset Password, How to Add Markers and How to Use Map. Buttons added to the sidemenu for each process would. The preference was to have the videos appear on the same screen rather than being embedded in a separate page.

A video modal was created that would populate and autoplay over the screen when the relevant button was pressed (Online Tutorials, 2022). This worked as expected after an issue involving the Z-index with both the side menu and modal not appearing. This had been an issue already resolved in the 'Locate me' section (see page 27). The Z-index is how elements are stacked and appear on screen, the index for each element on the page needed to be set in a hierarchy so that

one would show above the other when called. Elements with a higher index will appear on top of lower indexed elements.

Statistics

To provide statistics driven feedback and motivation a statistics page was added showing how many users had registered, and how many of each type: bin, dog waste, bagged dog waste and litter signs had been logged.

A PHP ‘get_tallies.php’ was created to interact with the database, count the rows containing each and send them back.

Two seperate databases needed to be accessed to gather this information, ‘pooglema_users’ that contained the ‘user’ table and ‘pooglema_poomark’ which had the ‘markers’ table. Login in credentials were the same but a seperate connection was created for each database. An SQL query was needed to count the amount of rows containing all the users and each marker type (GeeksforGeeks, 2021).

```php
// Establish connection for pooglema_users
$connUsers = new mysqli($server, $username, $password, "pooglema_users");
// Establish connection for pooglema_poomark
$connMarkers = new mysqli($server, $username, $password, "pooglema_poomark");
```

FIGURE 52: CONNECTING TO TWO SEPERATE DATABASES

```php
$queryUsers = "SELECT COUNT(*) AS totalUsers FROM users";
$resultUsers = $connUsers->query($queryUsers);
$totalUsers = $resultUsers->fetch_assoc();
```

FIGURE 53: SQL QUERY COUNT USERS

```php
foreach ($markerTypes as $type) {
    $query = "SELECT COUNT(*) AS count FROM markers WHERE type = '$type'";
    $result = $connMarkers->query($query);
    if ($result) {
        $row = $result->fetch_assoc();
        $markers[$type] = $row['count'];
    } else {
        $markers[$type] = 0;
```

FIGURE 54: SQL QUERY COUNT MARKER TYPES

This is sent back in JSON format to be displayed in the 'talliesContainer'.

This worked as expected after having initial errors with syntax, erroneous brackets and misspellings being the most common.

The statistics were displayed at 'pooglers.html', this would refresh every 10 seconds.

```html
<meta http-equiv="refresh" content="10">
```

FIGURE 55: PAGE REFRESH METHOD

This worked as expected and test users and markers were added to the system to confirm the statistics were being updated. The drawback to this was it was visually jarring with the refresh method sending a request every 10 seconds for the full page and its contents to reload causing a flickering of the screen. An alternative solution was found using the setInterval method (Mozilla.org, 2022). This was added to call the 'fetchUpdates' function every 10 seconds, this happened in the background and only updated the statistic numbers if changed.

```javascript
document.addEventListener('DOMContentLoaded', function() {

  fetchUpdates(); // Fetch initial data when page loads

  setInterval(fetchUpdates, 10000);

});
```

FIGURE 56: SETINTERVAL METHOD

Transition From Test Environment

The application was brought online and tested on the Reclaim Hosting live server. The reason was simple; it needed to be seen if it worked in the real world. Also, the responsive views and look would need to be fine-tuned to use on a mobile device. This would make sure that the elements involved in the site would display appropriately and function correctly, this was not possible to elicit from the test environment.

This process involved loading all the relevant PHP and HTML pages to the public_html section of the File Mager section in cPanel. The homepage was set as index.html and successfully loaded when the URL www.pooglemaps.com was accessed.

Databases and tables were created in phpMyAdmin using the same SQL queries as the test environment and each of the PHP scripts were updated with the new connection and database login details. Folders were created for CSS and Media files. This process was relatively straightforward, most issues were rectified by correctly updating the PHP scripts with the new login details.

The application underwent mobile testing in both indoor and outdoor environments. Indoor testing focused on the user inputs in registration, login and mapping being sent to and received from the relevant databases correctly. Outdoor testing evaluated location features and the overall user experience in a real-world setting i.e. whilst walking a dog.

Login, registration, password and mapping features worked as intended. This had been extensively checked in the test environment.

The outdoor testing also worked as intended but flaws and oversights in the initial design were encountered. These were then dealt with as outlined below

Issues Identified and Remedies

1. Login issue

Problem: Application needed to be logged into each time the EDIT button was selected

Identifying the issue: After logging in successfully to the editing section of the map and exiting to view the updated map and subsequently returning to the EDIT page the user was asked to re-

enter their login details. Links to the EDIT page were set up requiring login each time. There was no way for the system to discern that the user had logged in previously. Inspecting the Local Storage in the developers tools section of the browser confirmed that there was no session token for the system to verify (OWASP, 2013).

Remedy: Session id management was needed and '`session_start()`' was added to the start of 'addMarker.php', 'login.html' and 'saveMarkers.php' so that a token would be sent to the local storage. This was tested by logging in again, clicking on the homepage link at the top of the page and subsequently returning via the EDIT button. This was successful and allowed access to the 'addMarker.php' page. Local storage was checked, and the token was visible. Other pages were accessed, and the EDIT button was pressed to see if it allowed access as well. This proved successful and the issue was closed.

FIGURE 57: SESSIONID TOKEN STORED

2. Deleted Items Populating Database

Problem: When editing the map, markers that had been deleted before saving were saved to the database.

Identifying the issue: Four bins had been added to the map. Two were removed before saving, when the database was checked all four had been saved. Subsequent editing and saving confirmed that the system was removing the markers from the map but still sending them to the database. The database then populated the 'MAP' page with these markers.

The remove method worked to remove from the screen but did not consider that the marker data had also being sent to an array for storage (GIS SOLUTIONS, 2023a).

Remedy: Markers being added to the map needed to be given a unique id so that when they are being removed from the map they will also be removed by this identifier from the array. Each

marker was assigned a random unique number (Repšys, 2021). A filter was set up to remove them from the array.

```
marker.on('click', function () {
  map.removeLayer(marker);
  markers = markers.filter(m => m.id !== uniqueId);
});
```

FIGURE 58: FILTER FOR REMOVING MARKERS

Eight markers, two of each were added to the map and one of each subsequently removed before saving. This worked as it populated the ‘markers’ database with only the markers that remained.

3. Not knowing what bins had been logged previously

Problem: Unable to discern on the ‘EDIT’ page if a bin already existed without backing out and checking the ‘MAP’.

Identifying the issue: It was noticed that a couple of bins close to each other had been added to the map. They would not have populated if they had been placed on the exact spot, this is quite unlikely depending on zoom level and the marker being placed.

Remedy: to mitigate this it was decided to pre-populate the ‘EDIT’ map with previously saved bins. A function ‘fetchAndDisplayBins()’ was created that triggered the script ‘saved-bins.php’through an ajax request. This is identical to the process used to pre-populate the ‘MAP’ page albeit with only bin icons being retrieved and displayed.

When the ‘EDIT’ is next accessed it populated with bins from the database. To confirm this, a series of bins were added and saved. They were then seen to be visible when refreshing the ‘EDIT’ page. This then gave better visibility of bins already saved.

Design Aesthetics

The website's layout, look and accessibility were a vital part of its design. A humorous and easy to use design was adopted to attract and engage with potential volunteers.

The layout has a minimalist approach with only three options available on the homepage which simplified the navigation process. Options were provided in a slide menu that do not link to elsewhere; this enables the user to stay engaged and focused by keeping them within that section.

Logo: The logo was created using MS Paint and an image provided by an acquaintance [3], searches of creative commons images did not return what was required.

FIGURE 59: LOGO

Icons: were sourced from ‘Leonardo.ai’ and a picture taken of a dog waste bag.

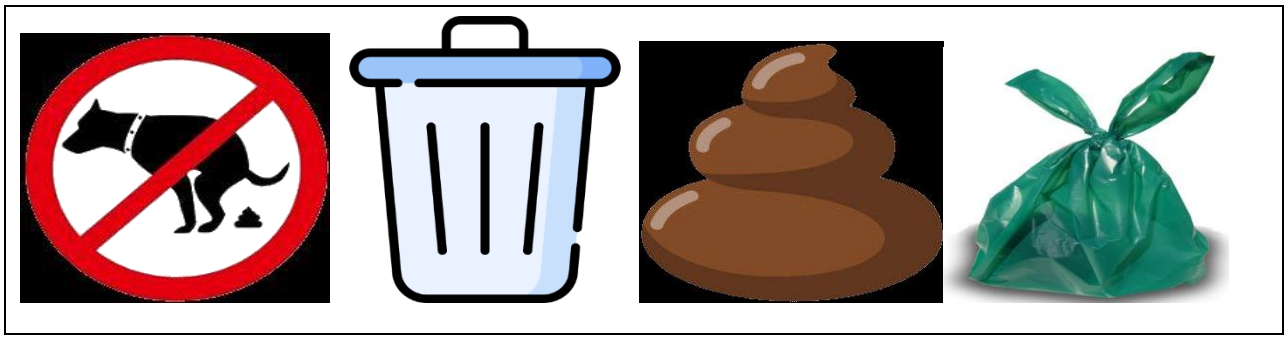


FIGURE 60: MARKER ICONS

[3] I would like to acknowledge Walter Wynne (WJ & Media) for providing images that were used in this project."

It was found that the icons displayed on the map with their backgrounds included. The dog litter sign, and the dog poop sign are images with visible backgrounds. The bin and bagged poo are those having had them removed. GIMP is an editing software tool that was able to remove the layers and give the background a transparent element. This allowed a more attractive image to be layered onto the map. This was a time-consuming and impractical solution for what seemed a basic process. An alternative was found using the object clip function found on Samsung Galaxy phones. When a picture is opened and long pressed the object is pulled from its background. This can be saved and pasted onto different backgrounds.

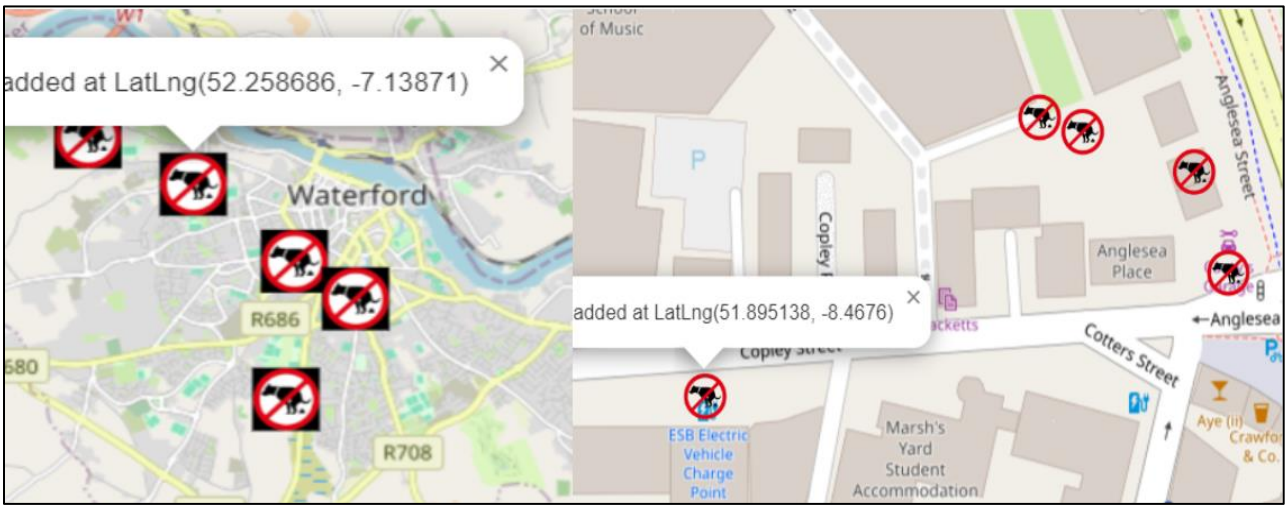


FIGURE 61: ICONS PRE AND POST LAYER REMOVAL

Images: the images used for the homepage were sourced from an AI image generator Leonardo.ai. (Leonardo.ai, n.d.). Using prompts, it was able to generate content that could not otherwise be created without a proficiency in art and graphic design. The main image and its background were sourced using prompts such as “draw a happy cartoon dog cleaning up poop in a park” and “draw a sunny multicoloured cartoon streetscape”.

FIGURE 62: AI GENERATED IMAGES

FIGURE 63: CLIPPED OBJECT ADDED TO STREETSCAPE

The dog was removed from its background, added to the streetscape and edited with the spray can feature in MSPAINT

CSS: was used to control the presentation and aesthetics of the web pages. It allowed the development of how the layout, colours, fonts, backgrounds, animations, and overall visual appearance of the elements were defined. By separating content from presentation, CSS enabled

consistent and efficient styling across the application, enhancing the user experience and creating a visually appealing and responsive web design.

CSS allowed the footer and headers to remain static and give the impression that the website was an application, or a device specifically built for this purpose.

It allowed the motto on the homepage to appear in a different font to that of the desktop version as one was more suitable than the other. This was achieved by assigning a font specifically to mobile devices using the @media query.

The buttons were specifically designed to be placed in the same place on each page to give a seamless transition between sections. They were given shade and a round appearance to give them depth and the appearance of reality. This was achieved by using a button generator which could be tweaked in real time to the desired effect (images.unsplash.com, n.d.).

By using black on the header and footer and variations of brown for the other elements, the application was given a warm, earthy, and natural aesthetic. This was hoped to make it more inviting and accessible.

FIGURE 64 BUTTONS BEFORE AND AFTER CSS STYLING

Implementation

Having designed, created, customized and tested the application it needed to be released to the public. This was built with the intention of providing dog walkers with a device that could identify and route them to the nearest dog waste bin.

Crowdsourcing has become a popular, cost effective and efficient way of gathering data and leveraging the collective knowledge and experiences of a wide range of peoples. These benefits are particularly apparent when recruiting, engaging with and deploying a mobile application (Wang et al., 2016).

Features had been created that were not planned for in the initial design stages. These were not released to the public as they were deemed extraneous for the application's planned purpose. These were administrative and data visualizing tools that did not translate well to a mobile screen. They were built leveraging Leaflet plugins with functions already created for adding and saving markers.

Distribution and marketing the application was free. Links to the website were included in texts, group chats and posted on social media. People were also encouraged to share the link with their friends and followers. Follow up posts were then sent daily to remind people to engage with the website.

An email address john.creedon@pooglemaps.com had been created and was set as a point of contact in the about section and on the account registration email. People were actively encouraged to reply via social media channels.

The first social media post was issued on the 10th of April, take was initially slow but 22 people registered in th first 24 hours. A follow up post was sent on the 14th of April by that stage 31 people had registered, a further 9 registered in the next 48 hours and in the first week 40 people had registered on the application. Icons were being successfully added to the map and were visible on the ‘MAP’ page

Feedback

User feedback is instrumental for web-based applications, especially one like this that allows users to record and find items on a map. Such applications rely heavily on user interaction and data accuracy, making feedback crucial for both functionality and user satisfaction. Feedback was gathered through various methods with email and social media interactions being the most common.

A proactive problem resolution mandate was set in place to keep user frustrations to a minimum. The importance of how these are acted on helps anticipate and solve problems before they become widespread, minimizing negative impacts on the user experience.

Issues that have been brought to light and how they were remedied will be shown here. To fully understand the nature of these issues, user actions were closely replicated as much as possible. A triage approach was adopted to get to the crux of the matter. This hands-on methodology provided firsthand experience, facilitating problem identification and subsequent resolutions.

Issues Identified and Remedies

1. Enabling Location Permissions

Problem: iPhone users experienced difficulties enabling permissions when loading the maps.

Identifying the issue: The application had been designed to assume a certain level of technical proficiency, which, in this case, was presumed to be more widespread among users than it was.

Remedy: A concise screen recording demonstrating how to update the settings was uploaded to the tutorial videos section of the About page. Interestingly, this recording was sourced from another user who had access to an iPhone, thereby ensuring the instructions were both authentic and practical for the intended audience.

2. Pop-ups not appearing

Problem: Pop-up alerts intended to provide brief descriptions of each page's purpose were not appearing consistently across the 'Edit', 'Map', and 'About' pages.

Identifying the issue: Users reported confusion about the purpose and functionality of each page, partly due to the absence of introductory pop-ups. The pop-ups were designed to trigger on the first visit to each page using a local storage mechanism to prevent repeat appearances, however they were only showing on one page and not the others.

Remedy: These pop ups are JavaScript window alerts (). They are designed to appear the first time that the page is landed on only. The local storage is then updated to show this. The next time the page is visited the local storage will be checked first, if the variable popupShown is present, no alert will show.

```
    if (!localStorage.getItem('popupShown')) {

    alert("Welcome, add markers to the map,\n if you make a mistake
before saving\n you can drag markers or tap on them to remove .");

        LocalStorage.setItem('popupShown', true);

}
```

FIGURE 65 ALERT POP UP CODE

All three pages shared the variable name ‘popupShown’ for their respective alerts. It was discovered that accessing the pages on different devices would only allow one page to show an alert. Was it that the alerts were only showing once or were they not working at all? How and where the alerts were stored needed to be examined.

Local Storage can be viewed by accessing the Application tab in the developer tools section of the site. The ‘popupShown’ was visible when sent to the local storage, it could be deleted and when accessing another page, it allowed the alert to display and populate local storage again as true.

It was noticed that the alerts were seen as one for the entire site and not per page. The solution was to modify each pop-up so that it would be independently tracked and displayed. They were changed respectively to ‘mapPopupShown’ for Map, ‘editPopupShown’ for Edit and ‘aboutPopupShown’ for About.

Testing: Deleting all three of the keys from the local storage and subsequently revisiting the webpages confirmed that the alerts showed once per page as intended. This modification ensured that each page's introductory alert would be managed separately, maintaining the intended user experience.

3. User not found

Problem: User not found issue on reset password

Identifying the issue: Users were being given the error, "User not found. Please check your details and try again." when requesting a password reset. Many reached out immediately flummoxed by the fact that it was exactly their names and email being used that they provided originally. Users were asked if they had used upper- or lower-case iterations of their names.

The users table in the 'phpMyAdmin' database was cross checked manually to confirm with the user their details, in each instance they matched the details they had input. A sample of names and email addresses were used on the reset password form to see if it was a once off occurrence or an issue with all the users. Out of six attempts two of the name and email address combinations returned user-not-found errors even though they were copied and pasted directly from the 'users' table.

Using the search on either of those two accounts returned an empty result set. Searches were done by first name, which returned the first account and not the second account. Searching by surname returned the second account, not the first, and searching by email address returned both. The details could be seen in the database, and they were searchable. The information was coming over from the 'register.php' script as intended.

The database was exported so that a manual check could be performed to see if the user table contained any irregularities. It was noticed that the two accounts affected had leading spaces after the first name in one and surname in the other. The users had set up their accounts unintentionally with an extraneous space included, the system did not recognize this as an error or a potential difficulty as it was taking instructions explicitly from the user as correct.

When entering these details with trailing spaces included in the password reset form a successful message appeared stating that "A password reset link has been sent to your email address.” The arrival of this reset password email with the user was confirmed.

A test account was set up to replicate the conditions described above and a similar outcome was achieved. This account was deleted from the database and reused again for surname and email address iterations of the test to confirm the issue had been identified fully. In each instance the extra spaces were being accepted by the system as part of the name, surname, and email address.

Remedy: On further inspection it was seen that there were more than two accounts affected by this. An SQL query was done on each of the columns in the table to identify the extent of the issue. The query was updated for surname and email address accordingly.

The query retrieved the first_name from each row of the users table and calculated the number of trailing spaces in each first_name. It does this by finding the difference between the original length of first_name and its length after removing any trailing spaces.

The function RTRIM (Right Trim) is designed to remove trailing whitespace to the right of a string.

```
SELECT

    first_name,

    LENGTH(first_name)     -     LENGTH(RTRIM(first_name))     AS
trailing_space_count

FROM users;
```

FIGURE 66 RTRIM FUNCTION

Finally, it gives a name (trailing_space_count) to this count of trailing spaces

In this instance Michael and Jennifer had recorded two spaces and one space extra in their first name field. The remaining names returned no extra spaces. five accounts were found using the search via surname and two more when querying by email address. A similar set of queries was run with LTRIM (left trim) and none were found to have extra spaces to the left of the identifier.

There weres even accounts affected in total and a clean-up was needed to sanitize the data going forward. For each of the headings first_name, surname and email the following query was run to remove any spaces to the left or right.

```
UPDATE users

SET first_name = LTRIM(RTRIM(first_name));
```

FIGURE 67 SQL QUERY TO CLEAR SPACES

After executing this query, all the first names in the users table that contained any leading or trailing spaces were removed. The initial trailing spaces count query was run again and returned no trailing spaces count across all three identifiers. The updated database was downloaded, the accounts that had been affected were manually checked and they were now rectified.

The test account was again used to register with a trailing space and created an account that was not found when the trailing space was not included. This was queried, found, sanitized, and deleted for future use. The problem had been identified, tidied up but not fully rectified as the issue was still being allowed to happen at the point of registration.

It was decided that a client-side sanitization of the user inputs should be done on the ‘register.html’ page. It ensured that the data going to the server side would be error free and consistent. The data had been accepted and processed previously on the server side without recognizing or looking for the error.

```
function trimAllInputs() {

    const ids = ['first_name', 'surname', 'email'];

    ids.forEach(id => {

        var input = document.getElementById(id);

        input.value = input.value.trim();

    });
```

FIGURE 68 SQL QUERY TO CLEAN ALL DATA

This function will iterate over the pre-defined list of input field ids, retrieve the input element, trim any leading or trailing spaces from it and update the input field with the trimmed value. This function is then called when the submit button is pressed on the registration form.

```
document.getElementById('registrationForm').addEventListener('su
bmit', trimAllInputs);
```

FIGURE 69 EVENTLISTENER TO PERFORM TRIM

Testing: the test account was used again to register, trailing spaces were included on first name, surname, and email address. Registration was a success, the database duly updated. Previous SQL queries for trailing spaces count were performed and zero issues were found. There were no issues logging in with the new account details or requesting a password reset. The tests were a success, and the issue has now been resolved fully.

4. Lack of verification when successfully registering

Problem: Users were able to login without opening the welcome email and clicking on the welcome login. This could allow users to set up accounts with emails other than their own. They would not need to validate receipt of the account being successfully registered at the email address provided.

Identifying the issue: Users found that they were able to login after registering without needing to open the welcome email and click on the login link included. This revealed an oversight in the planning process - there was no verification process to confirm that the email addresses provided by users were their own. The welcome email included a login link, but there was no actual verification step involved, as the link only directed users to the login page.

Using a test account, a new account was registered and confirmed that an email would be sent to the email address provided. The users database was checked to see if the account had been registered which it had been. Login was successful using the details just provided and markers were successfully added to the map.

The email address was then checked for the registration success email, this had arrived as expected at the correct account address. On further inspection it was confirmed that there was no verification incorporated in the email other than a semblance of one in the provision of a login link

I, John Doe, agree to take part in this study. Please click on the link to login: https://www.pooglemaps.com/login.html. Clicking on the link opened the login page and confirmed that there was no verification process in place.

Remedy: A verification process would need to be included going forward to enhance account security and accuracy of the data being held in the users database.

Like the password hash also created on account registration, a unique verification token would be created and stored in the database. This token will have a status that will change on verification of the welcome email. This status will then determine that the account is active, and grant access when logging in.

The users table was updated with two new columns using the following query.

```
ALTER TABLE users

ADD COLUMN verification_token VARCHAR(255),

ADD COLUMN is_verified TINYINT(1) DEFAULT 0;
```

FIGURE 70 SQL QUERY TO CREATE NEW COLUMNS

A decision was made to update all previously created accounts with the status is_verified to 1. There were only 22 accounts created at the time and if there were any issues they could be dealt with manually on the system.

```
UPDATE users

SET is_verified = 1

WHERE is_verified = 0;
```

FIGURE 71 SQL QUERY TO UPDATE USERS ACCOUNT RETROSPECTIVELY

The registration process needed to be updated to incorporate these new requirements and database changes. A random verification token was created using the random_bytes() and bin2hex functions, converting the 16 bytes of random data into 32 hexadecimal characters.

```
$passwordHash = password_hash($password, PASSWORD_DEFAULT);
$verificationToken = bin2hex(random_bytes(16));
```

FIGURE 72 ALERT POP UP CODE

The existing prepared statements are updated to expect the two new fields is_verified and verification_token

```
$stmt = $pdo->prepare("INSERT INTO users (first_name, surname, email, password_hash, is_verified, verification_token) VALUES (?, ?, ?, ?, ?, ?)");
```

FIGURE 73 ALERT POP UP CODE

When the statement is executed the placeholder question marks above are replaced by an updated array, now including a “0” as default and the randomly generated $verificationToken.

```
$registrationSuccess = $stmt->execute([$firstName, $surname, $email, $passwordHash, 0, $verificationToken]);
```

FIGURE 74 ALERT POP UP CODE

A verification link incorporating the $verificationToken was created and included in the email. Clicking on the link would open a PHP, ‘verify_email.php’ that would check the database if the token was valid and would update the is_verified status from “0” to “1”

```
// Check if the token is valid

$stmt=$pdo->prepare("SELECT*FROM users WHERE verification_token
=?");

$stmt->execute([$verificationToken]);

$user = $stmt->fetch();

if ($user) {

// Update the user's verification status

$stmt = $pdo->prepare("UPDATE users SET is_verified = 1 WHERE id
= ?");

$stmt->execute([$user['id']]);
```

FIGURE 75 PHP STATEMENT TO UPDATE USER TO VERIFIED

Testing: the test account was used again to register. An updated message added to ‘register.php’ shows "Registration successful"! A verification email has been sent, please click on the link to verify your email.”

The ‘users’ database is checked and confirms that the account has been set up, ‘verification_token’ created and ‘is_verified’ set at ‘0’.

To replicate the original problem, logging in was tried again before clicking on the link and was successful even though the verification process has been set up. The ‘login.php’ has not been updated to also check for the ‘is_verified’ status.

The prepared SQL is updated to also check for the user's verified status and to check the email that was input matches.

```
$stmt = $conn->prepare ("SELECT id, password_hash, is_verified
FROM users WHERE email = ?");
```

FIGURE 76 ALERT POP UP CODE

The users password is checked using the PHP ‘password_verify’ function and if found to be verified, the user is redirected to ‘addMarker.php’ and exits the script execution.

```
(password_verify($password, $user['password_hash'])) {

    if ($user['is_verified']) {

$_SESSION['user_id'] = $user['id'];

$_SESSION['is_verified'] = true;

header("Location: addMarker.php");

exit();
```

FIGURE 77 ALERT POP UP CODE

The test account was deleted from the database again and used to reregister once more. Registration successful, email sent, and database checked to confirm creation and ‘is_verified’ is set at ‘0’. Login is attempted and the following warning appears, “Your email is not yet verified. Please check your email for the verification link.”

Clicking on the verification link in the email subsequently updates the ‘is_verified’ status to ‘1’.

Login is successful and user is now able to access the ‘addMarker.php’ editing section of the website.

Conclusions

This project was initiated to develop an application that would leverage crowd-sourced volunteers to record the location of dog bins, dog related waste and signage. The aim was to empower dog walkers in selecting and defining cleaner walking routes by having the knowledge available for the proper disposal and avoidance of dog waste. It was also aimed at applying the knowledge gleaned from studying the Digital Humanities course at University College Cork.

A functional application that fulfilled these requirements was successfully designed, created, tested, and deployed. Volunteers were actively sourced and engaged through various channels to contribute data to the application. They were given the ability to record and view data that could be updated in real time in a user friendly and efficient manner. This would allow them to make informed choices on how and where to dispose of and avoid dog waste.

One of the key benefits of sourcing data from the crowd for this project is that it allowed the application to extend beyond a single geographic area. By engaging volunteers from multiple locations to contribute information, the application was able to provide a more comprehensive and widespread mapping of dog waste bins, waste areas, and relevant signage. There are 51 registered users of the application who have provided hundreds of data points.

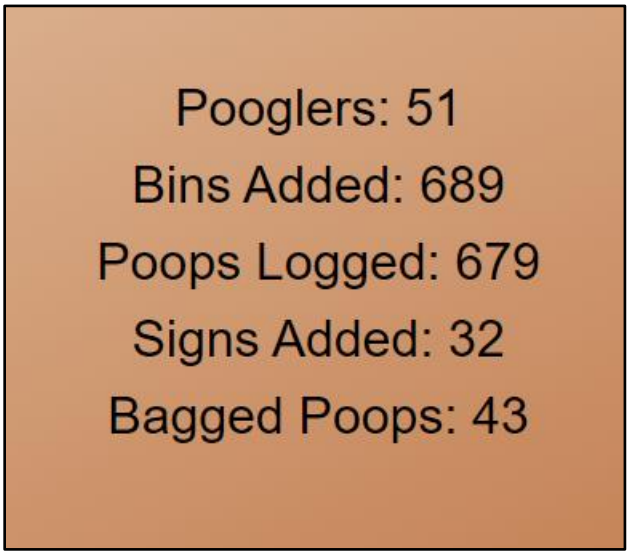


FIGURE 78: USER STATISTICS

While the main focus of the project has been the creation of the application, the successful sourcing of volunteers from the crowd could have been further improved or expanded. Successful crowdsourcing involves engaging more proactively in recruiting and retaining volunteers (Xu, Wu and Hamari, 2022). Recruitment involves continuous targeted media campaigns and communication. Retention through robust feedback and support mechanisms keep the user engaged and interested.

Due to time constraints and the purview of the project this was deprioritized to providing a basic sample of crowdsourced users and data. Elements that could have been included are dedicated user areas for updating personal details, viewing statistics and route planners. Leaderboards to foster healthy competition and engagement.

Features were created but not included in the application as they were beyond its original scope. They were also seen as unwieldy to use on a mobile device. These could be used for the extraction and analysis of the markers directly from the map without needing direct access to its database.

These included:

- A dual screen synchronous map that utilized the Leaflet.sync plugin and a filtering mechanism (Waagmeester, 2024). One map displayed a heatmap of the icons while the second displayed the icons.
  https://www.pooglemaps.com/heatmap.html

- A map that allowed users to draw lines around an area using the Leaflet.draw plugin (GitHub, 2018) and extracting the data to a CSV file (www.javatpoint.com, n.d.). https://www.pooglemaps.com/poly-fetch.html

- A map that could be populated with the data uploaded from a CSV file. https://www.pooglemaps.com/upmarker.html

The application could incorporate these features and swap out the types and icons used in the datasets. This would allow for the creation of similar maps and data that could be analyzed to show incidences of other factors. This could include and not be limited to reporting potholes, vacant and

derelict properties or even incidences of illegal dumping. This could aid in the deployment of enforcement and deterrent measures.

References

Reference list

Agafonkin, V. (2024). *Quick Start Guide - Leaflet - a JavaScript library for interactive maps*. [online] leafletjs.com. Available at: https://leafletjs.com/examples/quick-start/ [Accessed 26 Mar. 2024].

Agafonkin., V. (2024). *Leaflet on Mobile - Leaflet - a JavaScript library for interactive maps*. [online] leafletjs.com. Available at: https://leafletjs.com/examples/mobile/#:~:text=Leaflet%20has%20a%20very%20handy [Accessed 23 Apr. 2024].

DEV Community. (n.d.). *jQuery Datatables Ajax PHP and MySQL using PDO Example*. [online] Available at: https://dev.to/codeanddeploy/jquery-datatables-ajax-php-and-mysql-using-pdo-example-39of [Accessed 5 Apr. 2024].

GeeksforGeeks (2021). *How to count rows in MySQL table in PHP ?* [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/how-to-count-rows-in-mysql-table-in-php/ [Accessed 25 Apr. 2024].

GIS SOLUTIONS (2022a). *Leaflet on click add custom marker*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=EkBEJ59Ak1I [Accessed 23 Apr. 2024].

GIS SOLUTIONS (2022b). *Leaflet tutorial - How to add a Custom Marker*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=GmH7qkdUy3Q&t=320s [Accessed 23 Apr. 2024].

GIS SOLUTIONS (2023a). *Leaflet Delete Marker on click*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=o041giJsXZQ [Accessed 23 Apr. 2024].

GIS SOLUTIONS (2023b). *Leaflet Find Current Location*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=PGGUfnYD_xo [Accessed 23 Apr. 2024].

GIS Solutions (2022). *Leaflet on click add standard marker*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=nuvictDdCHI [Accessed 23 Apr. 2024].

GitHub. (2018). *Leaflet/Leaflet.draw*. [online] Available at: https://github.com/Leaflet/Leaflet.draw [Accessed 26 Apr. 2024].

GitHub. (2022). *Installation*. [online] Available at: https://github.com/dropbox/zxcvbn.

Hamid, H. (2023). *A Comprehensive Comparison: Leaflet, OpenLayers, Mapbox GL JS, Google Maps JavaScript API, ESRI…*. [online] Medium. Available at: https://norcomdentaire.medium.com/a-comprehensive-comparison-leaflet-openlayers-mapbox-gl-js-google-maps-javascript-api-esri-430c98b657fe#:~:text=Like%20Leaflet%2C%20it%20supports%20tile [Accessed 21 Apr. 2024].

images.unsplash.com. (n.d.). 🚀 *CSS button generator | 2023 custom button CSS & Get code*. [online] Available at: https://www.htmlcssbuttongenerator.com/css-button-generator.html.

Leonardo.ai (n.d.). *Leonardo.Ai*. [online] app.leonardo.ai. Available at: https://leonardo.ai/ [Accessed 1 Apr. 2024].

Liedman, P. (2024). *perliedman/leaflet-routing-machine*. [online] GitHub. Available at: https://github.com/perliedman/leaflet-routing-machine [Accessed 24 Apr. 2024].

Mendez, J. (2023). *Display All PHP Errors: Basic & Advanced Usage*. [online] Stackify. Available at: https://stackify.com/display-php-errors/#:~:text=Configure%20PHP.ini%20to%20display%20all%20errors&text=The%20display_errors%20directive%20must%20be [Accessed 23 Apr. 2024].

mmtuts (2018). *How To Create A Complete Login System In PHP | Procedural MySQLi | 2018 PHP Tutorial | mmtuts*. *YouTube*. Available at: https://www.youtube.com/watch?v=LC9GaXkdxF8.

Mozilla.org. (2022). *setInterval() - Web APIs | MDN*. [online] Available at: https://developer.mozilla.org/en-US/docs/Web/API/setInterval.

Nikulski, E. (2021). *Leaflet/Leaflet.markercluster*. [online] GitHub. Available at: https://github.com/Leaflet/Leaflet.markercluster.

Online Tutorials (2022). *Animated Video Popup on Click using Html CSS & Javascript | How To Create Responsive Video Modal*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=cw21m2S5PXQ [Accessed 25 Apr. 2024].

ORACLE (2024). *MySQL :: MySQL 8.0 Reference Manual :: 12.7 Date and Time Functions*. [online] dev.mysql.com. Available at: https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html.

OWASP (2013). *Session Management · OWASP Cheat Sheet Series*. [online] Owasp.org. Available at: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html.

PHP GROUP (2009). *PHP: mysql_query - Manual*. [online] Php.net. Available at: https://www.php.net/manual/en/function.mysql-query.php.

Qiu, S., Psyllidis, A., Bozzon, A. and Houben, G.-J. (2019). Crowd-Mapping Urban Objects from Street-Level Imagery. *The World Wide Web Conference*. doi:https://doi.org/10.1145/3308558.3313651.

Reclaim Hosting. (n.d.). *Reclaim Hosting – Take Control of your Digital Identity*. [online] Available at: https://www.reclaimhosting.com/.

Repšys, P. (2021). *Simplest possible way to generate unique ID in Javascript*. [online] Medium. Available at: https://paulius-repsys.medium.com/simplest-possible-way-to-generate-unique-id-in-javascript-a0d7566f3b0c [Accessed 25 Apr. 2024].

Stack Overflow. (2016). *How to add satellite view in leaflet?* [online] Available at: https://stackoverflow.com/questions/35487887/how-to-add-satellite-view-in-leaflet [Accessed 23 Apr. 2024].

Stack Overflow. (2017). *Get JSON data from PHP function using AJAX*. [online] Available at: https://stackoverflow.com/questions/42596191/get-json-data-from-php-function-using-ajax [Accessed 24 Apr. 2024].

Stack Overflow. (2018). *Adding clickable button to Leaflet map*. [online] Available at: https://stackoverflow.com/questions/49372018/adding-clickable-button-to-leaflet-map [Accessed 23 Apr. 2024].

Stack Overflow. (2021). *How to use leaflet markerclusterGroup?* [online] Available at: https://stackoverflow.com/questions/49333263/how-to-use-leaflet-markerclustergroup [Accessed 24 Apr. 2024].

w3schools (n.d.). *PHP AJAX and MySQL*. [online] www.w3schools.com. Available at: https://www.w3schools.com/php/php_ajax_database.asp.

Waagmeester, J.P. (2024). *jieter/Leaflet.Sync*. [online] GitHub. Available at: https://github.com/jieter/Leaflet.Sync [Accessed 26 Apr. 2024].

Wang, Y., Jia, X., Jin, Q. and Ma, J. (2016). Mobile crowdsourcing: framework, challenges, and solutions. *Concurrency and Computation: Practice and Experience*, 29(3). doi:https://doi.org/10.1002/cpe.3789.

Wook, A. (2023). *Introduction | zxcvbn-ts*. [online] zxcvbn-ts.github.io. Available at: https://zxcvbn-ts.github.io/zxcvbn/guide/ [Accessed 24 Apr. 2024].

www.javatpoint.com. (n.d.). *JavaScript create and download CSV file - javatpoint*. [online] Available at: https://www.javatpoint.com/javascript-create-and-download-csv-file [Accessed 26 Apr. 2024].

www.tutorialspoint.com. (n.d.). *LeafletJS - Markers*. [online] Available at: https://www.tutorialspoint.com/leafletjs/leafletjs_markers.htm.

www.w3schools.com. (n.d.). *How To Create a Register Form*. [online] Available at: https://www.w3schools.com/howto/howto_css_register_form.asp.

www.w3schools.com. (n.d.). *WebSecurity ResetPassword Method*. [online] Available at: https://www.w3schools.com/asp/met_websecurity_resetpassword.asp.

www.xul.fr. (n.d.). *Ajax Tutorial*. [online] Available at: https://www.xul.fr/en-xml-ajax.html [Accessed 23 Apr. 2024].

XHTML, Javascript, AJAX, PHP. (n.d.). Available at: https://www.massey.ac.nz/~nhreyes/MASSEY/159339/Lectures/Lecture%2018%20-%20AJAX-PHP.pdf.

Xu, H., Wu, Y. and Hamari, J. (2022). What determines the successfulness of a crowdsourcing campaign: A study on the relationships between indicators of trustworthiness, popularity, and success. *Journal of Business Research*, 139, pp.484–495. doi:https://doi.org/10.1016/j.jbusres.2021.09.032.

Acknowledgements

I would like to take this opportunity to thank Water Wynne and Leanne Walsh who helped me with the design, development and coding of this project. Their inputs, guidance and experience in web development and programming was invaluable.