Interactive Online Portal and Data Management Pipelines for an Interdisciplinary Citizen Science Project

Alexandra Griffin
2148000

December 2022

Project Dissertation Submitted to Swansea University in Partial Fulfilment for the Degree of Master of Science

Department of Computer Science, Swansea University

SUMMARY

This work describes the process of designing and implementing an interactive web application to be used for the management of EcoGardenHealth, a soil testing citizen science project conducted at Swansea University. The purpose of the work is to determine the most effective ways to use a web application to facilitate a successful citizen science project by streamlining the data collection process and encouraging user participation through ease of use. The characteristics of a successful citizen science web application were identified through a thorough literature review and then incorporated into a novel web portal for a project with no existing web application. Although the need for further improvements and user testing is indicated, the result of the work is an easy-to-use web application designed from the perspective of all users, including tangential stakeholders.

DECLARATION AND STATEMENTS

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Statement 1

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

Statement 2

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Alexandra Griffin

Alexandra Griffin

December 7, 2022

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Citizen science projects rely on data collection from a large number of research participants, typically located over a wide geographic area. Although data collection proved to be difficult to facilitate in the past, technological improvements have allowed home scientists to participate in citizen science efforts all over the world. However, many of these projects have not reached their full potential because they still rely on outdated data collection methods, which can prove cumbersome, insecure, and detrimental to the overall success of the research. Occasionally, those with custom web applications are still missing the mark for web design quality and secure data storage and, thus, could be missing out on potential participants.

The purpose of this work is to explore the ways in which computer science practices can streamline the data collection process of a citizen science project while providing secure data storage and engaging data visualizations for researchers and participants.

1.1 Background and Motivation

In order to outline the motivation of this project, it is important to provide an understanding of the EcoGardenHealth project. EcoGardenHealth is the inspiration for the design and development of the web application described in the current work, which provided the opportunity to pose questions surrounding the characteristics of a successful citizen science web application.

1.1.1 EcoGardenHealth Project Background

EcoGardenHealth is a soil testing citizen science project conducted by the Biosciences Department at Swansea University. The head researchers, Dr. Konstans Wells and Dr. Tamsyn Uren Webster, designed the project to explore the state of soil contamination in the Swansea area. The success of the project relies on the participation of Swansea residents, who are required to fill out a digital form and submit five physical soil samples to take part.

1.1.2 EcoGardenHealth Project Aims

The researchers involved in the EcoGardenHealth project aim to collect soil samples from throughout the Swansea area and perform testing to determine the levels of individual elements in each sample. They hope to provide individuals with their test results and use these datapoints to identify soil contaminant trends in the area. Because the project relies on citizen participation, another aim of the project is to attract and retain participants who will provide data and soil samples.

1.2 Problem Statement

At the beginning of this work, the EcoGardenHealth researchers had already established a workflow for their project. They received a few soil samples during the completion of this work and returned sample results to the participants. However, they recognize that the addition of a custom web application would ease the flow of the project and facilitate data collection.

1.2.1 Current Setup

In the current data collection setup of the EcoGardenHealth project, participants are recruited through direct emails and provide sample data through a Microsoft Form managed by the researchers. Citizen scientists are then instructed to devise a personal sample ID number by using their postcode and collection date. After the results have been calculated, they are distributed to participants via direct email after being manually typed into tables by the researchers.

1.2.2 Stakeholders

The primary stakeholders in the EcoGardenHealth project include the researchers and the research participants. From the researchers' perspective, the web application needs to provide an easy way to disseminate project information to potential participants, collect data from participants, provide results to participants, track soil samples, and store easily accessible data for the project. From the participants' perspective, the web application needs to provide accessible project information, as well as an easy and secure way to sign up and provide the initial data for participation. Secondary stakeholders include other researchers interested in the results of the project and anyone else who may be interested in EcoGardenHealth. These interested parties need the web application to provide a convenient way to learn about the project and an engaging visualization of the results.

1.3 Scope and Objectives

The objectives of this project are to design, implement, and deploy a custom web application for the EcoGardenHealth citizen science project according to the needs of the stakeholders, as well as to identify the characteristics of a successful citizen science web application and apply them to the current project.

The main objective for the creation of the web application is to provide a data management pipeline for EcoGardenHealth. Users should be able to securely submit participation data, accept the terms and conditions of the project, and view the results of their personal soil sample test results. Researchers should be able to access participant data and share soil sample test results. The application should also provide general information about the project and feature interactive data visualizations to generate interest in the results.

An additional objective of the current work is to provide an analysis of the process of creating a web application for a citizen science project through a computer science lens. Although there is sufficient literature describing the process of creating an application from the perspective of citizen scientist researchers themselves, there seems to be a gap in accounts from a computer scientist's perspective. Ultimately, this interdisciplinary project exists at the intersection of citizen science and computer science, and the perspectives of both disciplines are necessary for a thorough evaluation of the subject.

1.4 Methodology

A thorough literature review was conducted to discover what previous research has revealed to be the most useful features to include in a citizen science web application. Following an Agile project management strategy, the application was developed iteratively on the basis of predefined use cases and the results of the literature review, which were used to identify precise software requirements. A design phase was also conducted in which the user interface, program architecture, and database design were planned before implementation commenced.

The software itself was built using Django, a Python web development framework. PostgreSQL was used for the relational database, and Bootstrap, HTML, and CSS were used to create a user-friendly and visually appealing user interface. Finally, a geospatial data visualization was created using Plotly Express, Mapbox, GeoPy, and Pandas.

1.5 Achievements

Overall, the project was successful, with a working web application as the result. The application streamlines the data collection and dissemination process. It eliminates multiple steps from the current workflow, allowing the researchers to save time and achieve more secure data handling. The user interface is simple and easy to use and tailored for accessibility and responsiveness to reduce the number of potential participants who might be inadvertently excluded from the project because of accessibility or hardware limitations.

1.5.1 Tangible Achievements

This project achieves success from three different perspectives: general interested parties, participants, and researchers.

When the general public visits the web application, they are presented with Home, About, and Map pages. The Home and About pages provide static information about EcoGardenHealth, whereas the Map page displays current results presented in an interactive and engaging manner.

Figure 1.1 EcoGardenHealth Application Home Page

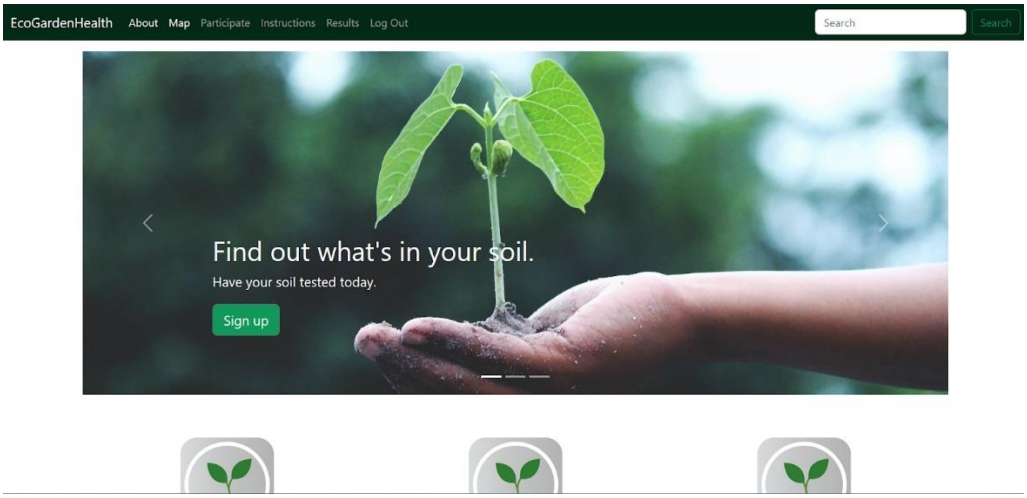


Figure 1.2 EcoGardenHealth Application About Page

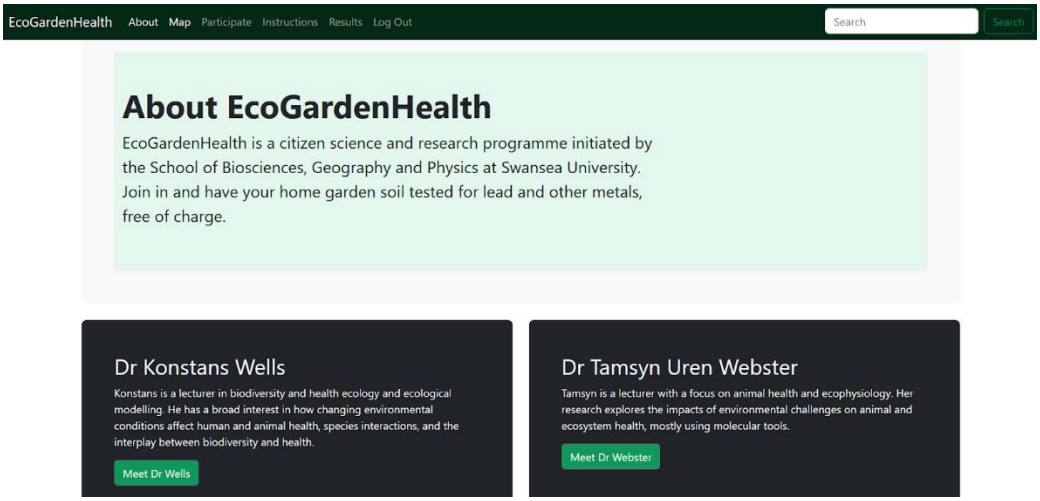
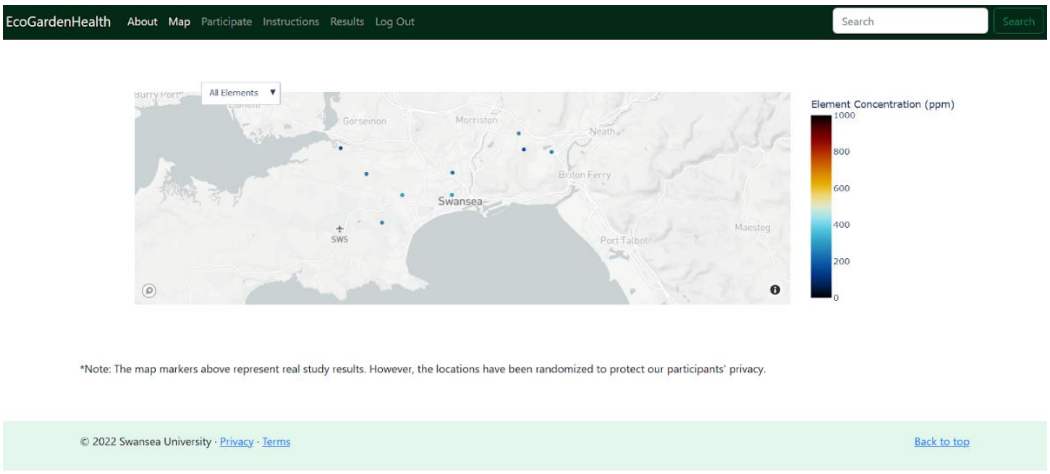


Figure 1.3 EcoGardenHealth Application Map Page



Participants are invited to create a user account, which reveals the participant's version of the web application. This version combines the pages of the general public view with the Participation Form and Participant Results pages. These pages allow participants to submit project data and enroll in the project and provide automatically populated soil testing results, when available.

Figure 1.4 EcoGardenHealth Application Participation Form Page

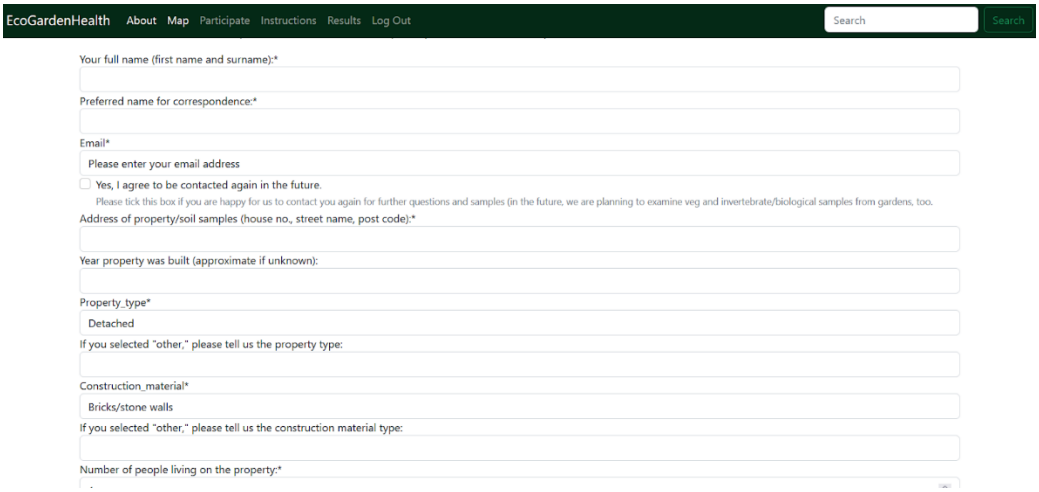
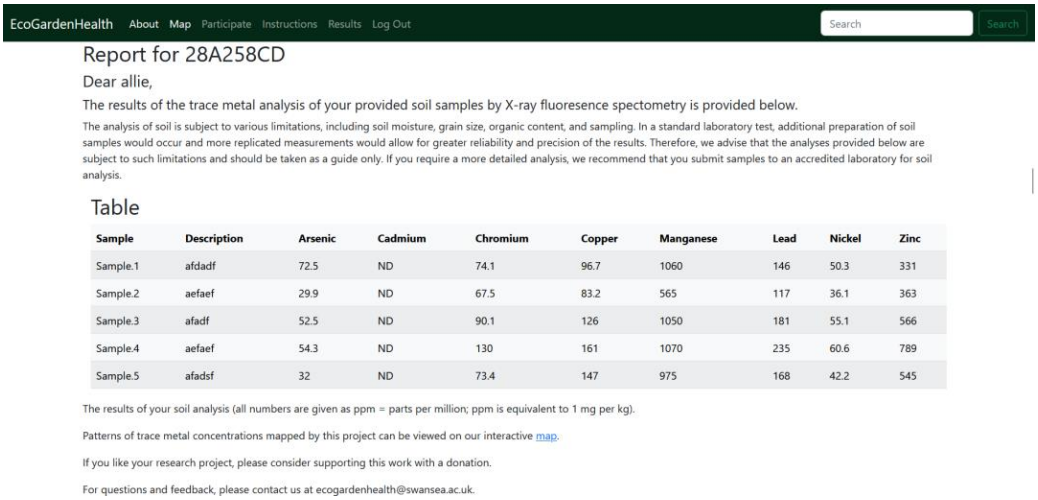


Figure 1.5 EcoGardenHealth Application Participant Result Page



Finally, the researchers are provided with an automatically generated soil sample ID system that allows them to effortlessly match participants with their soil samples. They also have a search feature that allows them to access any participant's data in an instant, a complete admin site that allows them to modify or delete data, and a result upload system that automatically parses data, stores it in the database, delivers it to the participants, and populates it on the map. A complete collection of screenshots from the completed application can be found in Appendix 1.

Figure 1.6 EcoGardenHealth Admin Search Result Page

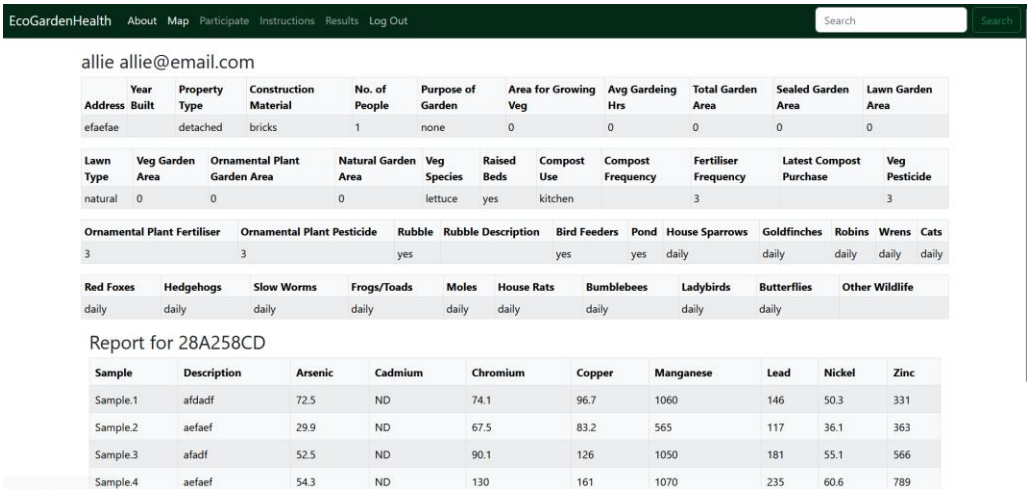


Figure 1.7 EcoGardenHealth Admin Home Page

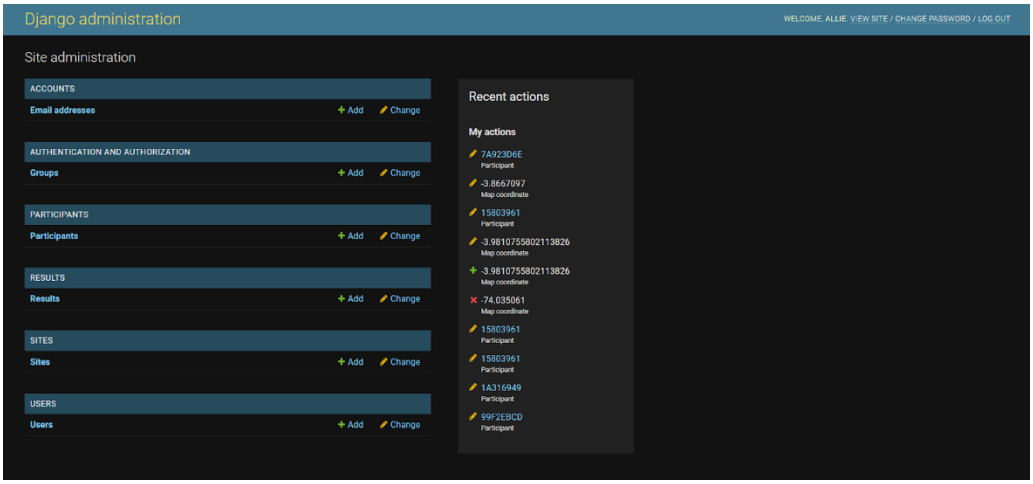
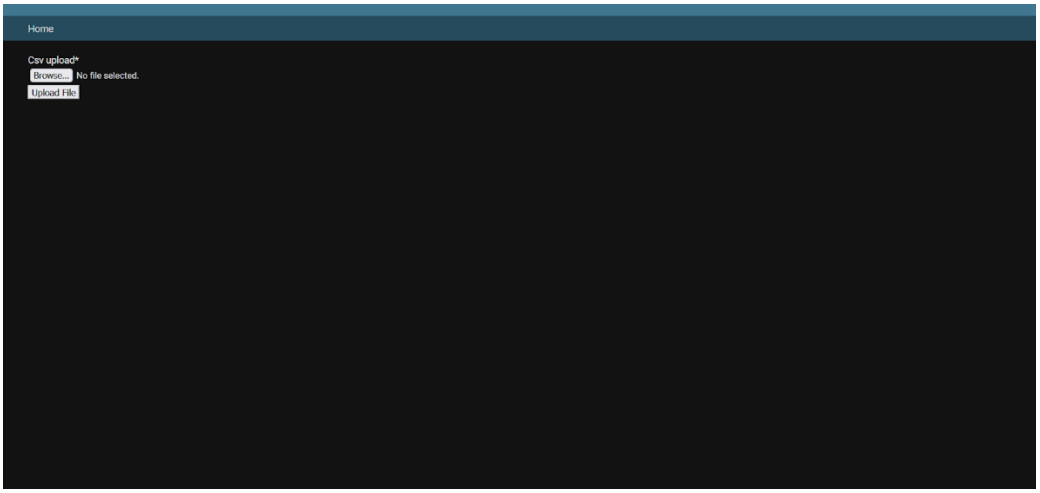


Figure 1.8 EcoGardenHealth Result Upload Page



1.5.2 Overview of the Development Process

I used an iterative Agile development process to implement one feature at a time. I began with the static pages of the general public view, added each participant feature, moved on to the researcher features, and finished the backend of the project with the map. I finished the project by styling the frontend to make the web application more attractive and user-friendly. I conducted extensive manual testing as I went to ensure each feature worked correctly on its own and as a whole after integration with the main project.

1.6 Outline

This work is divided into several chapters. Chapter 2 contains background information and a literature review. It covers related work, similar applications and projects, and the evaluation of other scientific web application development projects. Chapter 3 focuses on software requirement specifications, including functional and nonfunctional requirements and the use cases used to identify the requirements. Chapter 4 is an overview of the methodology used to implement the project, which includes the theoretical framework, technologies, and development tools. Chapter 5 describes how the project was implemented, including the structural, database, and user interface design, development, and the most notable parts of the implementation process. Chapter 6 examines the results of the project, comparing them with the software requirements set out at the beginning of the process. Finally, Chapter 7 provides the conclusion of the work, including a discussion of the overall outcome of the project, its limitations, planned improvements, and plans for future work.

2. BACKGROUND AND LITERATURE REVIEW

At the outset of this project, I conducted a thorough review of related work, which helped me find a direction for my own project. An analysis of similar citizen science projects to EcoGardenHealth and their web applications inspired the design and implementation of my own. Finally, a literature review of studies related to perfecting the design of citizen science applications further informed the design of the EcoGardenHealth application and inspired use cases that I would not have otherwise considered.

2.1 Similar Applications and Projects

To get an idea of where my project stands in the current body of work, I began by finding and examining similar projects to EcoGardenHealth and reviewing their web applications. By noting what seemed to work particularly well and what seemed to fall short, I was able to gain inspiration for my own application. I was also able to identify some pitfalls to avoid during the design and implementation phases.

The similar projects on which I focused included VegeSafe,[1] Soilsafe Aotearoa,[2] and a Soil Collection Program at the University of Oklahoma.[3] All three are soil collection citizen science projects, which provided highly relevant material to examine in regard to EcoGardenHealth. Two of the projects also involve the collection of soil samples for contaminant analysis, whereas one involves soil testing to identify potentially valuable fungi for medical research.

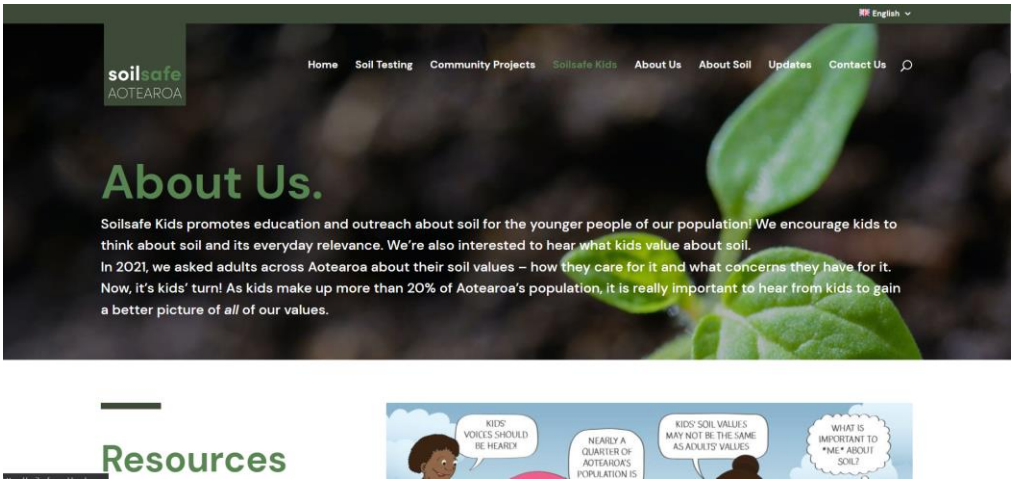
2.1.1 Evaluation

After evaluating all three web applications, I discovered some features that seemed to work well for the projects. The first is that all three websites included an interactive map that displayed soil testing results as geospatial datapoints. Although I already planned to include this feature in my web application, seeing it in use confirmed the impact that this type of data visualization has on a project of this type.

Figure 2.1 Soilsafe Aotearoa Map Page[4]

Some other positive features I found were a kids' page designed to encourage parents and schools to get children involved in the project on the Soilsafe web application and a discover page filled with FAQs and fun facts to cultivate interest and encourage participation in the University of Oklahoma project. Although the researchers are not interested in providing the content for these pages at present, it is worth noting them here as inspiration for future work.

Figure 2.2 Soilsafe Aotearoa Soilsafe Kids Page[5]



There were also less-positive features that I found on these web applications, including the lack of sophisticated participation forms and the open availability of participation forms without a user account creation barrier. Although the requirement to create a user account before the participation form can be accessed may discourage certain users from participating in the project, the measure provides a layer of security that is otherwise not present. Password-protecting the results and database submissions helps protect users' privacy and allows the results dissemination process to be completely automated, thus saving the researchers time.

2.2 Literature Review

The literature surrounding the intersection of technology and citizen science emphasizes the ways in which well-thought-out technological tools can enhance and facilitate citizen science projects. As written in the 2012 study by Newman et al., "[c]itizen science creates a nexus between science and education, that when coupled with emerging technologies, expands the frontiers of ecological research and public engagement."[6] This sentiment is echoed by Mazumdar et al. in their 2018 study: "[t]echnology development has steered the direction of citizen science and offered new mechanisms for engaging volunteers."[7] Skarlatidou et al. also suggest that collaboration amongst citizens, scientists, and human-computer interaction specialists can further these benefits.[8]

When searching for literature to guide the direction of this work and provide specific insight into the technologies and features to incorporate in the design and implementation phases, most of the articles I reviewed fell into one of two categories. The first was a series of articles written to examine and review citizen science applications created for specific projects. The second was a group of articles written to guide researchers or developers in creating their own citizen science applications or general web applications. Both types provided invaluable direction in the inception of this project.

2.2.1 Common Themes

In reviewing these articles, I found several common themes that were repeated consistently throughout the work. These included usability and design simplicity, with a focus on user-centered design, secure data-handling practices, and other design and content inspiration

Usability

Almost all of the literature focusing specifically on citizen science web applications explicitly stated the importance of user-focused design and the overall usability of any research and data collection tools.[9–14] Because citizen science projects heavily rely on community participation, it is necessary to make the data collection process as easy and enjoyable as possible for participants.

A specific aspect of usability that was mentioned in multiple studies is accessibility.[9,10,15,16] Ensuring that the web application is accessible to users of all abilities is essential to ensuring that no one is excluded from participation. This includes optimizing the web application for screen readers and using a high-contrast color scheme for visually impaired users. It also includes offering translation capabilities for users whose first language may not be English. This is an especially important consideration for a project centered in Wales, which has a large population of Welsh speakers. Another accessibility suggestion is the optimization of the web application for a variety of screen sizes through the use of responsive web design to reduce limitations on mobile-only users.[13,17]

Several of the studies suggested incorporating rounds of user testing into an iterative development process to ensure the user-friendliness of the final product.[13] Each development iteration may be used to incorporate the proposed changes into the application. One study also suggests collecting both qualitative and quantitative feedback to give more structure to the results.[12]

Data Handling and Security

Because the collection of citizen science data includes collecting potentially sensitive information about the volunteer population, it is of the utmost importance that this data is handled with care. Multiple studies specifically mentioned data handling and security as an essential part of designing and implementing a scientific web application.[15,18]

The first and most basic security issue is the use of encrypted connections to protect user inputs through transport layer security. Another suggestion is to incorporate third-party login integrations that allow for secure password management with minimal maintenance requirements.[15] The same study also suggested secure data storage in the cloud to ensure up-to-date security measures while removing the responsibility of maintenance from the researchers. Finally, compliance with local regulations and data privacy laws, such as those outlined by the General Data Protection Regulation (GDPR) will help keep the project from encountering any legal issues.[19]

Other Features

Some other specific design feature suggestions revealed in the literature review include the inclusion of a static website that provides information in addition to a portal designed for sign-up and participation[13] and the integration of an interactive map.[10] These features allow the researchers to provide important project data to all relevant audiences and encourage user participation.

One study suggests collecting participant demographic data for the researchers to analyze to create engagement and retention strategies.[9] This study and others also suggested increasing engagement and retention by gamifying the data collection process.[9,11,19] This method is suggested to increase the project's appeal with younger generations.

2.2.2 Evaluation of Development Methodologies

Several of the studies included in the literature review included specific development methodology suggestions in addition to overarching design recommendations.[7,10,13,17,20] Newman et al., in their 2010 study, emphasize the importance of paying careful attention to the selection of programming languages, software packages, databases, and data structures.[10] They also recommend a flexible and general database design paired with object-oriented code to allow for scalability and the easy addition of new features as new use cases arise.

Some of the study authors also recommended specific technologies for use in web application implementation. Mazumdar et al. suggested the use of a web development framework, such as WordPress, Django, Wix, or Weebly for the main application paired with a map overlay created using a mapping tool like OpenStreetMap, Google Maps, OpenLayers, or Mapbox.[7] Other technology suggestions include Python and Django as a language and framework,[7,13] PostgreSQL for the database and PostGIS for spatial data management,[13] Bootstrap and React for responsive user interface components,[17] and UML-based web engineering for architecture design.[20]

2.2.3 Conclusion

This literature review provided a more thorough list of use cases and requirement specifications for the EcoGardenHealth web application. From the focus on accessibility and a user-centric design to the recommendations of specific technologies, the previous work offers actionable suggestions for current developers of citizen science web applications. Although not every recommendation was prudent for the EcoGardenHealth project, it provided an excellent background of research on the topic.

This review also revealed a gap in the research that will be addressed by the current work. All of the articles cited in this work were written by citizen science researchers who created web applications for their own scientific projects. However, there were no articles specific to citizen science applications that were written from the perspective of a computer scientist. Therefore, most were lacking in clear technical recommendations and specific implementation examples, both of which I have included in this work.

3. SOFTWARE REQUIREMENT SPECIFICATIONS

Following the principles of Agile software development, the first step in implementing the EcoGardenHealth application was to define use cases, which would become software requirements.[21] I defined 16 use cases from the perspectives of the researchers, participants, and other interested parties, including potential participants and researchers from other similar projects. Guided by these use cases, I then created a list of functional and nonfunctional requirements for the software.

3.1 Use Cases

The use cases I developed range from tasks as general as learning about the project to those as specific as editing participant questionnaire responses. Each one was drafted with a specific audience in mind. For example, participants need to sign up for the project, check their soil sample label ID number, and check the status of their results. The researchers need to store and organize project data, view participant data, and share results with participants. The full list of use cases, their actors, and their basic flow can be found in Appendix 2.

3.2 Functional Requirements

Functional requirements relate to specific functions software is required to demonstrate.[22,23] I developed a list of functional requirements for the EcoGardenHealth application by translating the use cases into specific software functionalities. For example, the use case of a participant needing to sign up to participate in the project was broken down into several functional requirements, including those surrounding signing up for a user account, filling out a participation form made up of data collection questions, and user tracking when the form is submitted. The use case of a researcher needing to view participant data resulted in a requirement for a search feature that allowed administrative users to search for result and participation form data. The full list of functional requirements can be found in Appendix 3.

3.3 Nonfunctional Requirements

Nonfunctional requirements refer to software requirements that are not attached to specific functions, such as security and user-friendliness.[22,23] My analysis of the use cases resulted in a short list of nonfunctional requirements for the EcoGardenHealth application. These included points such as visual appeal, scalability, easy maintenance, security, and compliance. Although these requirements are more difficult to quantify than their functional counterparts, they provide necessary guidance during the development process. A full list of nonfunctional requirements can be found in Appendix 4.

4. METHODOLOGY

The final step before beginning implementation was to determine which methodologies I would employ during the development process. This included choosing a theoretical development framework, determining which technologies would make up the tech stack of the application, and choosing development tools to use during the implementation process.

4.1 Theoretical Frameworks

Although I originally planned to use the Waterfall software development method for my project, I found that an Agile development framework better suited the work. I found it much more effective to develop the application iteratively and integrate and test as I went rather than building the entire project and leaving the testing until the end.

4.1.1 Agile Development

Agile development is a popular software development and project management framework used in many organizations.[24] It is characterized by several fast-paced development cycles that are used to add new features to a project in an iterative manner. The Agile lifecycle is divided into phases of conception, inception and design, development and iteration, deployment and release to market, maintenance, and retirement.

Concept

The concept phase is used to determine the scope of the project, quantify software requirements, and prioritize tasks. I completed this part of the cycle when I developed the use cases and software requirements for the project.

Inception and Design

The inception and design phase is used to design the project's architecture and create a mockup of the user interface. This phase will be addressed in Section 5.1, in which I describe the design of my project. I chose to use a Unified Modeling Language (UML) class diagram to design the architecture, an entity relationship diagram to design the database, and publicly available design mockups to provide inspiration for the user interface of the application.

Development and Iteration

The development and iteration phase is where the majority of the work takes place. This phase includes its own subphases of planning requirements, developing the software, testing the software, and incorporating any feedback into the implementation. This phase is meant to be repeated as many times as needed until the application is finished and all functionality is present. I describe the development phase of my project in Section 5.2.

Deployment and Release to Market

After the project has been implemented, the deployment phase includes a round of full system testing to ensure that everything works as planned. Any bugs that are discovered are addressed quickly. User training and documentation tasks are also completed during this phase. This is the phase I am currently in with the EcoGardenHealth project. This work is part of the documentation and user training process, and system testing is ongoing in preparation for deployment.

Maintenance

After the software has been deployed, it moves into the maintenance phase. Developers provide ongoing support to fix any new bugs and keep the system running as it should. I intend to provide maintenance support for the EcoGardenHealth project for a time after the application is deployed.

Retirement

The final phase of Agile software development is retirement, in which an obsolete system is replaced or removed from use. However, this phase is beyond the scope of this work.

4.2 Technologies

When choosing technologies to include in the tech stack of the EcoGardenHealth application, I gave preference to languages, libraries, and other systems that are well-established, secure, and relatively easy to maintain.

4.2.1 Backend Technologies

Backend technologies are those used to create the architecture, logic, and database of the project. The EcoGardenHealth backend was created using Django, Python, PostgreSQL, Docker, and several third-party Django extensions.

Django and Python

Django is a high-level Python web development framework that streamlines the process of developing web applications using the Python programming language.[25] It is free and open source with excellent documentation and an active community. The built-in features of the Django framework keep web applications secure, scalable, maintainable, and portable while eliminating many of the repetitive tasks required when writing a web application completely from scratch.

In addition to these qualifications, multiple studies examined in the literature review suggested the use of Django when creating a citizen science web application.[7,13] The authors of the studies qualified the framework as easy to use and maintain, which is ideal for the EcoGardenHealth project because it allows the researchers to make changes and complete maintenance tasks on their own, as needed.

Another reason to use the Django framework is its ability to translate Python code for web browsers. Python's syntax is similar to the English language, making it one of the easier programming languages to read and learn. It is also heavily used in data science and other scientific applications. For these reasons, I found it preferable to more complicated languages, such as Java or JavaScript.

Third-Party Tools

Because the Django framework is so widely used and admired, there are many third-party extensions available to enhance its functionality. Although these third-party tools can be convenient and enticing, they carry their own security and poor functionality risks. For this reason, I limited my use to three well-established, -used, and -documented extensions.

The first of these is psycopg2, which is a database adapter extension that allows developers to connect a PostgreSQL database to a Django application.[26] It is the most popular database adapter of its kind, is designed for applications with complicated data operations, and is efficient and secure. This extension was necessary to trade the built-in SQLite database connection for the desired PostgreSQL connection.

The second extension I chose for the EcoGardenHealth application is Django allauth, which acts as a replacement for the built-in authentication system present in a basic Django project.[27] This fully integrated authentication application allows for local and third-party social authentication and eliminates the need to program a custom authentication system.

The final extension used in the citizen science application is Django crispy forms, which is an automated Django form styler.[28] It allows developers to control form rendering in a visually appealing way that eliminates a large amount of repetitive code.

PostgreSQL

PostgreSQL is an opensource relational database system with an excellent reputation and more than 35 years of active development.[29,30] It is one of the most widely used relational database systems available because it allows developers to safely store and scale complex data. It is reliable and robust and has an active opensource community. For these reasons, I chose it as the relational database used to store the data for the EcoGardenHealth web application.

Docker

One of the most common issues faced by developers is the need for a certain set of system configurations and dependencies to allow an application to run as intended. Docker is a system that allows applications to be encapsulated within a container that holds all of the dependencies and libraries necessary for the app to run smoothly.[31,32] Through the use of these Docker containers and images, the app can be run on any system instantly, without the need for extensive installations and configurations. It makes managing and deploying apps to live servers easy and secure. For these reasons, this project was created within a Docker container.

4.2.2 Frontend Technologies

Frontend technologies are used to create the client-facing user interface of a web application. When choosing the frontend tech stack of the EcoGardenHealth project, I opted for languages and libraries that would keep the frontend simple and easy to use while providing opportunities to optimize the application for accessibility and responsiveness.

HTML

HTML, or hypertext markup language, is the most basic building block of web development.[33] It provides an outline of web content while giving it meaning and structure. It also contains the links that connect pages to one another and annotations for text, images, and other content. Django uses its own form of modified HTML that simplifies the connection between an app's pages, which is what was used in the EcoGardenHealth project.

CSS3

CSS, or cascading style sheets, is the language used to modify the presentation of an HTML web page.[34] Because it identifies how elements should be rendered by the browser, it is used to alter the visual appearance of web pages. CSS was used to style the frontend of the EcoGardenHealth project through the Bootstrap CSS library.

Bootstrap 5.0

Bootstrap is the world's most popular CSS framework.[35,36] It makes creating responsive web applications effortless by using a unique grid-based organization for the elements of a web page. This library is free, opensource, and widely used throughout the web development world. Bootstrap templates contain a mixture of CSS, HTML, and sometimes JavaScript if animations are present. The pages of the EcoGardenHealth web application were designed by using modified Bootstrap templates.

4.2.3 Data Visualization

The final tech stack to consider when planning the EcoGardenHealth project was the one used to create data visualizations in the form of an interactive map. The use of Python as the main programming language of the project made the integration of several popular data visualization tools effortless.

Plotly and Plotly Express

Plotly is an interactive opensource plotting library used to visualize data on charts and maps.[37] It is indicated for geographic and scientific use cases, making it the perfect choice for the EcoGardenHealth project. Plotly Express is a built-in part of the Plotly library that uses functions to render entire figures at once.[38] Plotly Express paired with the Mapbox mapping technology[39] was what I used to create the interactive map on the citizen science web application.

GeoPy

In order to plot geospatial points on a dynamic map, the data provided by participants needed to first be converted into the correct data type. For a Plotly Express map, this means converting street addresses to latitude and longitude points. To perform this task, I used GeoPy, which is a Python client for geocoding web services that easily locates the coordinates of addresses and other geographic points.[40]

Pandas

The final step to populating the map was to feed the data from a database table into a dataframe, transforming it into a readable format for the Plotly figure. To do this, I used Pandas, which is a Python package that is designed to provide an easy way to work with relational data when performing data analysis in Python.[41]

4.3 Development Tools

The final step to set up for the implementation phase was to choose the tools I would use for the development of the application. Although the tools I chose are widely used and proven within the programming world, I mainly chose them on the basis of my own familiarity with how they work.

4.3.1 Visual Studio Code

The integrated development environment I chose to use was Microsoft's Visual Studio Code.[42] It is my code editor of choice because of its support for hundreds of programming languages, vast extension library, and convenient assistance, such as bracket matching and auto-indentation. After trying both VS Code and PyCharm at the beginning of the project, I settled on VS Code because it better recognized Django's modified HTML syntax.

4.3.2 Git and GitHub

A professional development environment is incomplete without version control. Making alterations to the main branch of a project without having a backup runs the risk of introducing a bug that crashes the entire program to devastating effect. For version control during the implementation of the EcoGardenHealth project, I used the ubiquitous opensource software, Git.[43,44] Finally, I stored my Git branches online by using GitHub to make the project available to access from any computer.[45,46]

5. PROJECT IMPLEMENTATION

Having the software requirements set out and the technologies chosen, I was able to begin the design and development phases of my Agile project. I began with the design phase and then moved through several cycles of the development phase, incorporating new features until the web application met all requirements.

5.1 Design Phase

The design phase began with identifying the type of application that would be best suited to the EcoGardenHealth project. After determining the application type, I designed the structure and architecture of the application and modeled it by using a UML class diagram. To define the structure and organization of the data storage, I created an entity relationship diagram. Finally, I created a mood board of user interface design inspiration to help direct my own user interface implementation.

5.1.1 Application Type

Considering the software requirements, the best implementation of the EcoGardenHealth project would be a dynamic application with a few static pages designed to provide information. Dynamic web pages feature information that is updated every time a user visits them, whereas static web pages remain the same every time they are visited.[47] Although the Home and About pages of the application would suffice as static pages, most of the pages would need to be dynamically updated through database queries.

A more specific implementation of the application would be as a project portal, which is a scientific gateway that is used for e-research and the dissemination of information among interested parties. The application must be designed to facilitate the sharing of data and resources without institutional restriction.[48]

5.1.2 Structural Design and Architecture

Planning the architecture and structural design of a complex software project is essential before beginning development. I chose to map the structure of the EcoGardenHealth application by using a UML class diagram.[49] This helped me define the models, views, and forms needed for the project, as well as organize the attributes and functions of each of these classes. It also created a visualization of how the different components of the software would interact with each other.

UML Class Diagram

UML class diagrams are based on the object-oriented programming principle of classes.[50] Each main component of a software project should, ideally, be given its own class, which, in turn, is given its own attributes and functions. The three superclasses in my diagram were Form, View,

and Model. Each of these is broken down into inheritance-based subclasses of the forms, page views, and models of the application. Every diagram is given its own attributes (the variables stored in each class) and functions (the operations performed by each class). The entire diagram is included in Appendix 5.

Figure 5.1 UML Diagram Subclass

5.1.3 Database Design

A well-functioning application is reliant upon a well-designed database. Organizing the way data is stored before developing the database reduces repetition, simplifies queries, and provides scalability. An entity relationship diagram is an effective tool to preplan the database design of an application. It displays the relationship of entity sets stored in a database and helps explain the logical structure. The three concepts modeled in this type of diagram include entities, attributes, and relationships.

Entity Relationship Diagram

I used a simple entity relationship diagram to plan the structure of the database used in the EcoGardenHealth project.[51] It features four entities, representing the four database tables in the project. Each entity features a list of attributes, which are the columns in each table (primary keys are marked in bold), and relationships, which are the way the tables are connected to each other. The full entity relationship diagram for the application can be found in Appendix 6.

Figure 5.2 Entity Relationship Diagram Entity and Attributes

model.map_coordinate	
participant	participant
address	varchar
latitude	varchar
longitude	varchar
rand_lat	float
rand_long	float
id	**int**

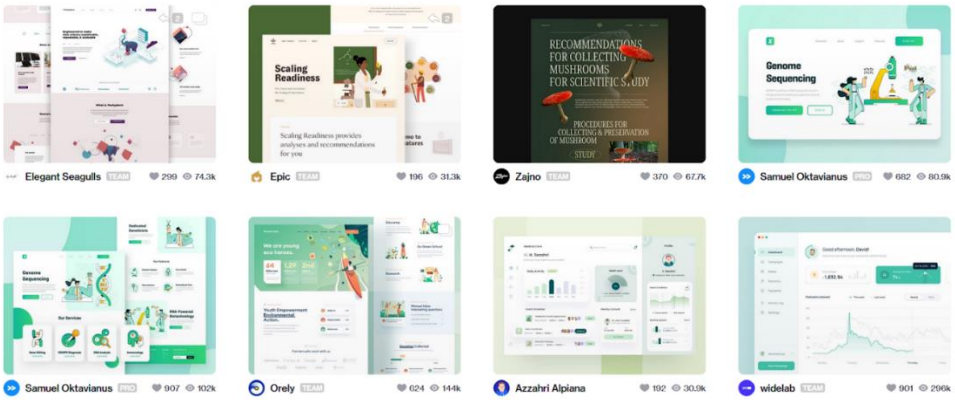
5.1.4 User Interface Design

Because I intended the design of this application to be user-centric, the user interface design played an important role in the design phase of the project. Although I lacked the skill and knowledge to create a full user interface mockup, I compiled a mood board filled with design inspiration to direct the implementation of the interface. This board contained mockups made by professional designers and made available to the public on Dribbble.com, a user interface design marketplace.[52]

Design Inspiration

The Dribbble collection I put together as inspiration for the user interface features 24 different website designs. I gravitated toward those with a scientific or natural theme and a green color scheme. The resulting EcoGardenHealth user interface was heavily influenced by these designs. The full Dribbble collection can be found in Appendix 7.

Figure 5.3 Dribbble Collection



5.2 Development Phase

The development phase of the project consisted of setting up the project for implementation and cycling through several development iterations, adding a new feature each time. After each feature was added, I tested it for functionality, addressed any existing bugs, then moved on to the next iteration.

5.2.1 Project Setup

To begin, I set up the project for development. The first step was to create a new Django project after installing the framework within a virtual environment. After the project was created, I installed Docker and created a new Dockerfile within the Django project, which acts as a snapshot in time of what the project contains and a list of instructions to be built. This file ensures continuity no matter in what environment the project is run. The addition of a docker-compose file containing instructions for the container built based on the Dockerfile image is the final step for Docker configuration.

The next step of the setup process is to install the third-party database adapter, psycopg2, and switch the database from the default SQLite to PostgreSQL. Adding the PostgreSQL configuration to the Django project settings file and the docker-compose file completes the database setup. The last step is to begin version control by creating a new Git file and saving the project to the main branch.

5.2.2 Project Structure and Implementation

The Django framework encourages the grouping of related functionality into applications, which can be reused throughout the project. Each application then contains its own groups of related code in individual modules, called models, views, and templates. An additional URL file maps the location of each page in the application and is used to redirect HTTP requests to the correct view. When a URL is requested, the request is forwarded to the view, which reads and writes data from the model, and displays said data, as well as page information, via the template.[25]

Applications

The applications I created for the EcoGardenHealth project correspond to each major piece of the database design. They include users, participants, results, and map. Each of these contains related model (each one corresponding with a relational database table) and view files, as well as URL maps.

The custom users app is managed by the third-party Django allauth and contains the data needed for sign-up, sign-in, and password reset functions. The participants app contains the participation form, the model in which the participation information is stored, and views for the form and submission success pages. The results app contains the results model, where the

results information is stored, the results page view, and the search results view for when administrators search for participant information and results.

The final applications in the project feature views but no models. They include pages, which contains all of the application's static pages, templates, which contains all of the templates for the project, and static, which contains all of the images, CSS files, and JavaScript files used in the templates.

Models

Models in a Django project correspond with tables in a relational database. The four models in the EcoGardenHealth project also correspond to the four tables in the database. These models store important project data and contain functions and queries to populate foreign key data from other models in the application.

The users model stores user login information, including email address, username, and password. Each new model instance (corresponding to a database table row) is created upon new user registration. This model is connected to an SMTP email service[53] that allows users to modify their own model instances by resetting their passwords.

The participants model stores participation form submission data. Using the Django model form creation functionality, I generated a form straight from the model to ensure the correct information would be collected in the participation form. Each new model instance is created upon participation form submission. These instances can be viewed, modified, and deleted by administrators. After each new participation form submission, a random and unique soil sample label ID number is generated, which is used to track the associated soil samples throughout the application, as well as during the physical delivery of the samples.

The results model is used to store soil testing result data. Model instances are created when the administrators upload a CSV file containing soil sample results. This data is used to automatically update the participants' results pages, which are connected to the model via the soil sample ID number.

The map model was originally designed to store the coordinates of the soil samples and populate the interactive map. However, the original design triggered the creation of map model instances upon the submission of a participant form. This resulted in empty data points becoming visible on the map before the results were uploaded. It is also necessary to include the data intended to be displayed on the map in the map model, rather than trying to populate it by querying the results model. A temporary fix of adding the latitude and longitude to the results model has been put in place, but I intend to refactor the map model for use before deployment.

Views

Django views contain the bulk of the logic of the application and control what happens when pages are requested and viewed in an application. They define the models, templates, and URLs associated with each page of the website. The views in the EcoGardenHealth application correspond to the applications and models of the project.

The static page views include the homepage view, the about page view, and the instructions page view, which is an additional static page that provides soil sample collection and submission instructions for authenticated users.

The participant views include the participant form view and the participant form success view. The form view contains the participation form as well as the logic necessary to automatically generate a random and unique soil sample label number. The success view displays when the form has been submitted. It includes a database query that displays the user's most recently assigned soil sample label number based on a timestamp included in the participant model.

The results views include a results page view and a search results page view. The results view displays an authenticated user's soil sample test results in the order of the most recently submitted participant form submission. Each result table is labeled with the soil sample label number.

The map page view is the most complex of all the views in the project because it contains the entire block of Plotly Express code to build and style the interactive map, including several different map layers that can be toggled using the built-in dropdown menu. It also contains Pandas code to read the database and build a dataframe, which is used to populate the map with data.

Templates

Django templates are text files that define the structure of an HTML page. They are written in Django's template language, which is a modified version of HTML. Each page view is given its own template, which acts as a set of written instructions for how the page should be displayed in the browser. The templates also include styling and links to static files, such as CSS style files and images.

In the EcoGardenHealth web application, I included a base.html template file, which contains the navbar and footer that are persistent across all pages of the website. The base Bootstrap styling is also included in this file. All of the pages also have their own custom templates defining their structure and style.

5.3 Interesting Parts of Implementation

Although the majority of the implementation of the web application was routine and uneventful, some parts proved to be more of a challenge. The three most difficult features to implement were the participant result tracking system, the result uploading feature, and the map. All required comparatively complex logic and produced several bugs.

5.3.1 Participant Result Tracking

By far, the most challenging part of the implementation phase was finding a way to track each soil sample throughout the web portal, as well as physically in the real world. In the first attempt at implementation, the number was only generated upon the creation of a model instance with a function located in the participant model file.[54–56] However, this resulted in the same number being generated for every instance, leading to errors surrounding the unique field constraint in the model. I solved this bug by adding the same random sample ID generation function to the participant form page view, which ensured that a new number was generated every time a form was submitted.[57]

The second bug that arose surrounding result tracking was the discovery that the correct soil sample label number was not being displayed on the participation form success page if there were multiple forms associated with one user. I had originally implemented the template to iterate backward through the form instances associated with the user in question and display the first one, with the intent of displaying the most recently created model instance. However, the instances were not stored in a last-in-last-out fashion, as I originally thought. I solved this issue by adding a timestamp field to the participant model and writing a database query in the form success view to display the soil sample ID from the model instance with the most recent timestamp.

Through extensive manual testing, I later discovered that the soil sample number was regenerated every time an administrator altered a participant's form data in the admin dashboard, which was the third major bug associated with this system. I tracked this error to the number generation function in the model and solved it by adding an if-else statement to the function to prevent a new number from being generated if the instance was not brand new. Specifically, the function checks if the instance already has a primary key assigned before assigning a new soil sample label number.[58]

5.3.2 Uploading Results

Rather than creating a page on the main web application for researchers to upload soil sample test results, I wanted to allow them to upload the results straight to the built-in Django admin section of the application. This way, they could instantly see if the upload had been successful by checking the results admin page. To achieve this goal, I altered the results admin page to include a link to a separate page with a file field upload form. The form allows the administrator to browse files on their computer and choose one to upload.[59,60]

I then wrote a custom parser, which first checks if the file type is correct (CSV) and then reads the CSV file and parses the data into the proper fields of the results model to generate a model instance for each test result.[61,62] I matched the parser's logic with the specific CSV file format produced by the soil testing machine used by the researchers. I also included logic that separates each soil sample included in a CSV file into its own model instance.

5.3.3 Map Implementation

In the initial implementation of the map feature, I used a Django signal to trigger the creation of a map model instance every time a participant form was submitted.[63] The map model also contained logic using GeoPy to convert the participant's address to latitude and longitude points, randomize those latitude and longitude points within the Swansea area, and plot them on the map.[64] However, upon testing, I realized that this implementation resulted in the rendering of map points that had no data because they were being generated before the results had been uploaded.

Because of time constraints, I needed a quick fix, so I added latitude and longitude variables to the results model and used that to populate the map instead. In the meantime, I also found that it was much simpler to retrieve the data needed to populate the map from a single model rather than two models, making the results model an even more attractive location for the geospatial data. However, a better fix would be to change the signal to trigger the map model instance creation upon the creation of a result model instance and add variables to the map model to store all of the necessary data in the same location. I plan to make this change before deployment.

6. RESULTS

With the exception of a few minor refactoring and deployment preparation tasks, the application has been implemented completely. To determine the results of my project, I have compared the original requirements set out at the beginning of the work and compared them to the reality of the application. I have also examined the results of the testing completed throughout the development phase.

6.1 Evaluation

By comparing the EcoGardenHealth application's features with the functional and nonfunctional requirements I defined in Sections 3.2 and 3.3 and Appendices 2 and 3, I have been able to determine the success of the project overall. A matching feature and requirement set is considered a success, whereas a requirement without a matching feature is marked for future completion in a final development cycle iteration.

6.1.1 Functional Requirements

The current version of the web application meets the vast majority of the technical requirements I set at the beginning of the project. All of the main functionalities are present surrounding the collection of data, the presentation of information, user registration, participant form submission, soil sample tracking, result upload and display, and map functionality.

The functional requirements that have not been met are mostly nonessential features having to do with the accessibility of the application. One specific requirement that has not been met is the implementation of a Welsh version of the website. However, translation-friendly syntax has been added throughout the application, meaning that the entire website is optimized for third-party translation tools. I also have not added third-party authentication yet because a permanent URL is required to sign up for the service, and I do not yet know what the application's URL will be. As soon as I do, I will add this functionality.

The other functional requirements that have not been met involve testing, which will all be completed before deployment. Although I have added responsive features to the application, I have not tested it on screens of varying sizes and cannot be sure that it works properly. The application also fails to pass all written tests because the tests have not been completed yet.

6.1.2 Nonfunctional Requirements

Although nonfunctional requirements are not easily quantified, the application appears to meet the majority of these specifications, as well. The use of the Django framework and Docker images provides the application with scalability and easy maintenance. Data security and scalability are achieved through the use of a PostgreSQL relational database. Django also

automatically provides security features to web applications to protect against vulnerabilities such as SQL injection, cross-site scripting, cross-site request forgery, and clickjacking. (Django documentation) All participants are informed how their submitted data will be used before collection, thus meeting GDPR requirements. Although the web application is not deployment-ready, it will be before it is hosted.

The most difficult nonfunctional requirements of which to determine success are ease of use and visual appeal. These requirements are highly subjective and cannot be quantified easily. Findings in the literature review suggested user testing to determine the success of a user interface. However, extensive user testing falls beyond the scope of this project because of time constraints.

6.2 Testing

Per the Agile methodology, testing made up a large part of the development process of the EcoGardenHealth web application. I completed extensive manual testing every time I added new functionality to the project or changed existing features.

6.2.1. Developmental Testing

Throughout the iterative development process, I used the software requirements list to choose a new feature to implement. After implementation, I returned to the requirement and tested the functionality of the feature against the functionality it was meant to have. I manually moved through the workflows of hypothetical website visitors, users, and administrators and observed the behavior of the application. If a bug was discovered, I fixed it before retesting and moving on to the next feature implementation. I discovered countless unexpected bugs in this way, including those mentioned in Section 5.3.

In addition to manual testing, Django offers a built-in, easy-to-use automated testing system. Unit tests can be written directly in a file in each of the project's applications. The tests are run through the command line, and the results are displayed in the terminal. I wrote some Django tests as I went, but I am still in the process of completing all of the written tests. As I finish the last changes to the project, I will complete the entire test suite.

7. CONCLUSION

Although the EcoGardenHealth web application largely meets the requirements set forth by the administrators of the project, there are several improvements that can be made. By analyzing the successes and limitations of the work, I have discovered opportunities for specific improvements to the application. I have also considered the project's success in adding to the relevant literature and its value in future areas of study.

7.1 Summary

This project has resulted in a fully functional web application with all of the necessary features to assist in the management of the EcoGardenHealth citizen science project. It has also resulted in a work that has the potential to aid in the creation of future citizen science web applications and helps fill the gap in the current literature by providing an account of creating such an app from a computer science perspective.

7.1.1 Discussion

With the exception of a few minor fixes and additions that will be completed before deployment and have only been omitted because of time constraints, the application has met the overwhelming majority of the specified software requirements. It provides the tools necessary to implement, track, and maintain a citizen science soil testing project.

The application will also successfully remove several steps from the current cumbersome workflow that the researchers currently use. All of the data and soil samples are tracked automatically, with the exception of manually labeling the physical samples and the test results, which cannot be avoided. Participants are provided with their results automatically, with the data presented in an organized manner. Researchers can also upload their results as they are to have them automatically stored in the relational database. The interactive map is also updated with no extra effort from the researchers. The new workflow can be viewed in the application handbooks provided in Appendix 8.

Although this is a largely practical project, it is important to consider the success of the work as a whole and its value in the current academic landscape. In addition to creating a working application for a citizen science project, I aimed to provide an account of the process from a traditional computer science perspective and show the value of using software development tools to streamline the workflow and enhance the success of a citizen science project.

This work accurately represents the process of conceptualizing, designing, and developing a citizen science web application from a software developer's perspective. Although it focuses less on the overall success of the citizen science project itself than most of the current literature, it provides an in-depth look at the development process in a manner that can be easily replicated. It has the potential to provide value to the developers of future citizen science

web applications as well as those interested in the ways in which technology can enhance the success of citizen science projects.

7.1.2 Limitations

The biggest limitation I encountered during this project was a lack of time. With the fast-paced nature of the timeline, I was unable to devote as much time as I wanted to each phase of implementation. With more time, I would have liked to complete several rounds of user testing in between development iterations. It would have helped create a more user-focused application if I had been able to incorporate features based on user suggestions into the project. The results of user testing surveys would have also been a valuable addition to the work in an academic sense.

There were also several optional features I would have developed had there been more time to do so. These included a rework of the participant result page as an interactive dashboard rather than a list of tables, personalized data visualizations for each participant, and an email alert system that would notify participants when their results are ready. I would have also customized the admin side of the application to provide researchers with easier access to the information they need rather than mostly leaving the default Django admin site as is.

At the beginning of the project, I intended to add a Welsh translation API to the web application to make the website more accessible to Welsh speakers. However, because of time constraints, I settled on adding translation syntax to the project to optimize the content for third-party translation tools. The planned written test suite also suffered from a lack of time, but it will be completed before deployment. This limitation was also mostly augmented by extensive manual testing throughout development.

Although the database works well for the project and is largely designed in an organized manner, I would make some changes to its design, in retrospect. I originally planned to have a separate table for the map coordinate data but ultimately combined the map and results models for expedience. I plan to reconfigure the models to return to the original design, which makes more sense and better follows the principles of database design. Because of my lack of knowledge about database design at the beginning of the project, I failed to assign the soil sample label variable as the primary key for the participant model. Because this is the main foreign key used to link the entire database, it should be the primary key.

Finally, my lack of knowledge about frontend development greatly limited my ability to create a more visually appealing and interactive user interface. Because I only had a minimal working knowledge of HTML and CSS, I had to heavily rely on the Bootstrap library and example templates to build the user interface. In the early phases of the project, I considered using React to create the frontend and limiting the Django backend to a pared-down API role. This approach might have made it easier to create a more sophisticated user interface.

7.2 Future Work

The completion of this work has revealed several opportunities for future work, both practically within the EcoGardenHealth project and in an overall academic sense.

7.2.1 Next Steps

The most immediate next steps of the EcoGardenHealth project are preparing the web application for deployment and getting it hosted on the university's servers. This includes a thorough security review and a few planned improvements that will take place before deployment. After the application has been hosted, the project will move into the maintenance phase, which means that I will be in contact with the researchers to fix bugs as they arise and, potentially, add or change features, when necessary.

Planned Improvements

To ensure the most complete version of the web application is deployed, I plan to make a few improvements that were lost in the previous development cycles because of time constraints. The first is reworking the map coordinate database table to store the map data in a way that makes more sense. The second is adding third-party authentication to the login and registration features, which will be possible when I find out the URL at which the web application will be located. The final improvement is to finish writing the entire automated Django test suite for the application to ensure that there are no bugs that slipped through during the rounds of manual testing.

7.2.2 Suggested Future Work

Although this project is nearing completion, the EcoGardenHealth project is just getting started. The researchers might consider enlisting the help of other developers to make further improvements and additions to the web application. Those developers might consider conducting further research on the basis of this project, as well.

Future developers who work on the EcoGardenHealth application might consider taking the opportunity to improve the application. Some of those suggested improvements include implementing an interactive dashboard to display participant soil sample result data visualizations in a more engaging manner, reworking the entire user interface to make it more sophisticated and less reliant on the Bootstrap library, and adding some customizations to the admin site to make it more user-friendly for the researchers.

Some other development work suggested by findings in the literature review include the addition of a mobile application, the addition of photo upload capability, and the implementation of a gamification feature. A mobile application might increase the reach of the EcoGardenHealth project to new audiences and make participation easier. Allowing participants to upload photos with their soil sample data might provide the researchers with more context

about the soil samples they receive. Finally, gamification is suggested in several studies as a means to engage potential participants, specifically younger demographics.

The biggest opportunity for future work revealed by the current project and the studies included in the literature review is the implementation of several rounds of user testing followed by several rounds of iterative development to incorporate the user suggestions into the application. Not only would give the application a more user-centric design, but it would also provide an excellent basis for further literature on the topic.

REFERENCES

1. *VegeSafe.* (2021) Macquarie University 360 Dust Analysis. https://www.360dustanalysis.com/get-started/soil/australia

2. *Soil testing.* The University of Auckland Soilsafe Aotearoa. https://soilsafe.auckland.ac.nz/soil-testing/

3. *Participate.* (2021) The University of Oklahoma Citizen Science Soil Collection Program. https://whatsinyourbackyard.org/participate/#scroll

4. *Map my environment.* Soilsafe Aotearoa. https://iupui-earth-science.shinyapps.io/MME_Global/

5. *About us.* Soilsafe Kids. https://soilsafe.auckland.ac.nz/soilsafe-kids/

6. Newman, G., Wiggins, A., Crall, A., Graham, E., Newman, S., & Crowston, K. (2012). The future of citizen science: emerging technologies and shifting paradigms. *Front Ecol Environ, 10*(6), 298–304.

7. Mazumdar, S., Ceccaroni, L., Piera, J., & Holker, F. (2018). Citizen science technologies and new opportunities for participation. In: *Citizen Science: Innovation in Open Science, Society and Policy*. London, UK: UCL Press; 303–320.

8. Skarlatidou, A., Hamilton, A., Vitos, M., & Haklay, M. (2019). What do volunteers want from citizen science technologies? A systematic literature review and best practice guidelines. *Journal of Science Communication, 18*(1).

9. Sun, C. C., Hurst, J. E., & Fuller, A. K. (2022). Citizen science data collection for integrated wildlife population analyses. *Citizen Science for Future Generations*, 8.

10. Newman, G., Zimmerman, D., Crall, A., Laituri, M., Graham, J., & Stapel, L. (2010). User-friendly web mapping: lessons from a citizen science website. *International Journal of Geographical Information Science*, *24*(12), 1851–1869.

11. Skarlatidou, A., Ponti, M., Sprinks, J., Nold, C., Haklay, M., & Kanjo, E. (2019). User experience of digital technologies in citizen science. *Journal of Science Communication*, *18*(01).

12. Karamanis, N., Pignatelli, M., Carvalho-Silva, D., Rowland, F., Cham, J. A., & Dunham, I. (2018). Designing an intuitive web application for drug discovery scientists. *Drug Discovery - Today*, *23*(6), 1169–1174.

13. Batsaikhan, A., Hachinger, S., Kurtz, W., Heller, H., & Frank, A. (2020). Application of modern web technologies to the citizen science project BAYSICS on climate research and science communication. *Sustainability*, *12*(18), 7748.

14. Garett, R., Chiu, J., Zhang, L., & Young, S. D. (2016). A literature review: website design and user engagement. *Online Journal of Communication and Media Technologies*, *6*(3).

15. Saia, S. M., Nelson, N. G., Young, S. N., Parham, S., & Vandegrift, M. (2022). Ten simple rules for researchers who want to develop web apps. *PLoS computational biology*, *18*(1), e1009663.

16. Burnett, J. L., Dale, R., Hou, C. Y., Palomo-Munoz, G., Whitney, K. S., Aulenbach, S., ... & Wellman, T. P. (2021). Ten simple rules for creating a scientific web application. *PLOS Computational Biology*, *17*(12), e1009574.

17. Shahzad, F. (2017). Modern and responsive mobile-enabled web applications. *Procedia Computer Science*, *110*(2017), 410–415.

18. Hansen, J. S., Gadegaard, S., Hansen, K. K., Larsen, A. V., Møller, S., Thomsen, G. S., & Holmstrand, K. F. (2021). Research data management challenges in citizen science projects and recommendations for library support services. A scoping review and case study. *Data Science Journal*, *20*(1).

19. *Data protection.* GOV.UK. https://www.gov.uk/data-protection

20. Mubin, S. A., Nelson & Poh, M. W. A. (2018). Web applications design methodology. *2nd global conference on computing & media technology.*

21. *Use cases.* Usability.gov. https://www.usability.gov/how-to-and-tools/methods/use-cases.html

22. *Why is the difference between functional and non-functional requirements important?* (2012). ReQtest. https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

23. *Functional vs non-functional requirements: main differences & examples.* The App Solutions. https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/

24. *The agile software development life cycle: agile guide.* Wrike. https://www.wrike.com/agile-guide/agile-development-life-cycle/

25. *MDN web docs: Django introduction.* Mozilla. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction#in_this_module

26. *psycopg2 – Python-PostgreSQL database adapter.* PyPi. https://pypi.org/project/psycopg2/

27. *Welcome to django-allauth!* django-allauth. https://django-allauth.readthedocs.io/en/latest/

28. *Forms have never been this crispy.* django-crispy-forms. https://django-crispy-forms.readthedocs.io/en/latest/

29. *PostgreSQL: the world's most advanced open source relational database.* PostgreSQL. https://www.postgresql.org/

30. *About.* PostgreSQL. https://www.postgresql.org/about/

31. *Developers bring their ideas to life with Docker.* Docker. https://www.docker.com/why-docker/

32. Palmer, S. *Docker deployment: how to deploy a web app using docker web server?* DevTeam.Space. https://www.devteam.space/blog/how-to-deploy-a-web-app-with-docker-containers/

33. *MDN web docs: HTML: HyperText markup language.* Mozilla. https://developer.mozilla.org/en-US/docs/Web/HTML

34. *MDN web docs: CSS: cascading style sheets.* Mozilla. https://developer.mozilla.org/en-US/docs/Web/CSS

35. *Introduction.* Bootstrap. https://getbootstrap.com/docs/5.0/getting-started/introduction/

36. *About.* Bootstrap. https://getbootstrap.com/docs/4.1/about/overview/

37. *Getting started with Plotly in Python.* Plotly. https://plotly.com/python/getting-started/

38. *Plotly Express in Python.* Plotly. https://plotly.com/python/plotly-express/

39. *Maps and location for developers*. Mapbox. https://www.mapbox.com/

40. *Welcome to GeoPy's documentation!* GeoPy. https://geopy.readthedocs.io/en/stable/

41. *Package overview.* Pandas. https://pandas.pydata.org/docs/getting_started/overview.html

42. *Why did we build Visual Studio Code?* Visual Studio Code. https://code.visualstudio.com/docs/editor/whyvscode

43. *Local branching on the cheap.* Git. https://git-scm.com/

44. *Why use Git for your organization.* Atlassian. https://www.atlassian.com/git/tutorials/why-git

45. *The home for all developers – including you.* GitHub. https://github.com/

46. Juviler, J. (2022). *What is GitHub? (And what is it used for?).* HubSpot. https://blog.hubspot.com/website/what-is-github-used-for

47. *Understand web applications.* (2021). Adobe. https://helpx.adobe.com/dreamweaver/using/web-applications.html

48. Allan, R. (2009). Virtual research environments. In: *Portals to Science Gateways.* Chandos Information Professional Services; 145–159.

49. *Create diagrams without touching your mouse.* Gleek. https://www.gleek.io/

50. *Class diagrams.* (2021). IBM. https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams

51. *Draw entity-relationship diagrams, painlessly.* DBDiagram.io. https://dbdiagram.io/home

52. *Soil testing project collection.* Dribbble. https://dribbble.com/alliefg/collections/5840981-Soil-Testing-Project

53. *Prepare for takeoff.* SendInBlue. https://www.sendinblue.com/

54. *Generate secure random numbers for managing secrets.* Python. https://docs.python.org/3/library/secrets.html

55. *UUIDField – Django models.* Geeks for Geeks. https://www.geeksforgeeks.org/uuidfield-django-models/

56. *Generate unique id in Django from a model field.* Stack Overflow. https://stackoverflow.com/questions/16925129/generate-unique-id-in-django-from-a-model-field

57. *Save random integer into form of Django.* Stack Overflow. https://stackoverflow.com/questions/67900153/save-random-integer-into-form-of-django

58. *Django - how to avoid double saving after defining the 'save' method on the model?* Stack Overflow. https://stackoverflow.com/questions/69409731/how-to-avoid-double-saving-after-defining-the-save-method-on-the-model

59. Very Academy. *Django admin CSV file upload.* YouTube. https://www.youtube.com/watch?v=BLxCnD5-Uvc

60. *Customize the Django admin with Python.* Real Python. https://realpython.com/customize-django-admin-python/

61. *Importing a CSV file into a PostgreSQL DB using Python Django.* FAQ Code 4 U. https://www.faqcode4u.com/faq/282844/importing-a-csv-file-into-a-postgresql-db-using-python-django

62. Barbacena, F. *Use Python scripts to insert CSV data into Django databases.* Towards Data Science. https://towardsdatascience.com/use-python-scripts-to-insert-csv-data-into-django-databases-72eee7c6a433

63. Freitas, V. (2016). *How to create Django signals.* Simple Is Better Than Complex. https://simpleisbetterthancomplex.com/tutorial/2016/07/28/how-to-create-django-signals.html

64. *Python: generate random latitude and longitude.* (2014). Hadoop Guru. http://hadoopguru.blogspot.com/2014/12/python-generate-random-latitude-and.html

BIBLIOGRAPHY

1. Lomas, A. (2021). *A 15-step guide on how to build a web app.* Net Solutions. https://www.netsolutions.com/insights/how-to-build-a-web-app/

2. Seiz, C. (2021). *How to develop a web application: your 10-step guide.* Zibtek. https://www.zibtek.com/blog/how-to-develop-a-web-application-your-10-step-guide/

3. Crestodina, A. *27 research-backed web design tips: how to design a website that works.* Orbit Media. https://www.orbitmedia.com/blog/web-design-tips/

4. Markovic, I. (2020). *The web development team workflow you should expect.* TMS. https://tms-outsource.com/blog/posts/web-development-team-workflow/

5. Lutski, Y. (2017). *What are your stages when creating a web app as a full stack developer?* Hashnode. https://hashnode.com/post/what-are-your-stages-when-creating-a-web-app-as-a-full-stack-developer-cixisahxm0090uy53vbg8ydh6

6. *Citizen science.* Khan Academy. https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:computing-innovations/xcae6f4a7ff015e7d:crowdsourcing-innovations/a/citizen-science

7. Vincent, W. (2020). *Django-allauth tutorial.* LearnDjango. https://learndjango.com/tutorials/django-allauth-tutorial

8. *Plot CSV data in Python.* Plotly. https://plotly.com/python/plot-data-from-csv/

9. *Scatter plots on Mapbox in Python.* Plotly. https://plotly.com/python/scattermapbox/

10. *Get started with Bootstrap.* Bootstrap. https://getbootstrap.com/docs/5.2/getting-started/introduction/

11. *Bootstrap example code.* Bootstrap GitHub. https://github.com/twbs/examples/blob/main/starter/index.html

12. *Examples.* Bootstrap. https://getbootstrap.com/docs/5.2/examples/

13. *Built-in template tags and filters.* Django. https://docs.djangoproject.com/en/4.1/ref/templates/builtins/

14. *How do I set user field in form to the currently logged in user?* Stack Overflow. https://stackoverflow.com/questions/18246326/how-do-i-set-user-field-in-form-to-the-currently-logged-in-user

15. *How to populate user field with current user in Django models via forms?* Stack Overflow. https://stackoverflow.com/questions/72034201/how-to-populate-user-field-with-current-user-in-django-models-via-forms

16. Bilesanmi, A. (2020). *How to drop and create a database in Docker—part 1.* Medium. https://bilesanmiahmad.medium.com/how-to-drop-and-create-a-database-in-docker-part-1-b705ca41a28e

17. Dekanovsky, V. (2020). *Python's geocoding—convert a list of addresses into a map.* Towards Data Science. https://towardsdatascience.com/pythons-geocoding-convert-a-list-of-addresses-into-a-map-f522ef513fd6

18. Grimaldi, E. (2018). *How to create interactive map plots with Plotly.* Towards Data Science. https://towardsdatascience.com/how-to-create-interactive-map-plots-with-plotly-7b57e889239a

19. Lamothe, T. (2020). *How to create outstanding animated scatter maps with Plotly and Dash.* Towards Data Science. https://towardsdatascience.com/how-to-create-animated-scatter-maps-with-plotly-and-dash-f10bb82d357a

20. Hwang, J.P. (2020). *Interactive maps with Python, pandas and Plotly: following bloggers through Sydney.* VisualNoise Data. https://www.visualnoise.io/interactive-maps-with-python-pandas-and-plotly/

21. *Mapbox map layers in Python.* Plotly. https://plotly.com/python/mapbox-layers/

22. *Bubble maps in Python.* Plotly. https://plotly.com/python/bubble-maps/

23. N.N.K. (2022). *Pandas convert column to float in dataframe.* Spark by Examples. https://sparkbyexamples.com/pandas/pandas-convert-string-to-float-type-dataframe/

24. *Dropdown menus in Python.* Plotly. https://plotly.com/python/dropdowns/

25. *plotly.express.scatter_mapbox.* Plotly. https://plotly.com/python-api-reference/generated/plotly.express.scatter_mapbox

26. *How to avoid double saving after defining the 'Save' method on the model?* Stack Overflow. https://stackoverflow.com/questions/69409731/how-to-avoid-double-saving-after-defining-the-save-method-on-the-model

27. Torimiro, S. (2021). *Supporting multiple languages in Django.* TestDriven.io. https://testdriven.io/blog/multiple-languages-in-django/

28. *MDN web docs: Django tutorial part 10: testing a Django web application.* Mozilla. https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Testing

29. Irwin, A. (2018). *No PhDs needed: how citizen science is transforming research.* Nature. https://www.nature.com/articles/d41586-018-07106-5

30. *The Django admin documentation generator.* Django. https://docs.djangoproject.com/en/4.1/ref/contrib/admin/admindocs/

31. *Web application development: 5 relevant types and examples.* Yeeply. https://en.yeeply.com/blog/5-different-kinds-web-app-development/

32. Peterson, R. (2022). *Entity relationship (ER) diagram model with DBMS example.* Guru99. https://www.guru99.com/er-diagram-tutorial-dbms.html

33. *Maps and location for developers*. Mapbox. https://www.mapbox.com/

34. *Local branching on the cheap.* Git. https://git-scm.com/

35. *The home for all developers – including you.* GitHub. https://github.com/

36. *Django ModelForm Boolean field giving none when true.* Stack Overflow. https://stackoverflow.com/questions/16991944/django-modelform-boolean-field-giving-none-when-true

37. Vincent, W.S. (2018). *Django for beginners.* Lean Publishing.

38. Vincent, W.S. (2018). *Django for professionals.* Lean Publishing.

39. Shaw, B.S., Badhwar, S., Bird, A., Chandra KS, B., Guest, C. (2021). Packt Publishing.

40. Mele, A. (2020). *Django 3 by example* (3rd ed.). Packt Publishing.

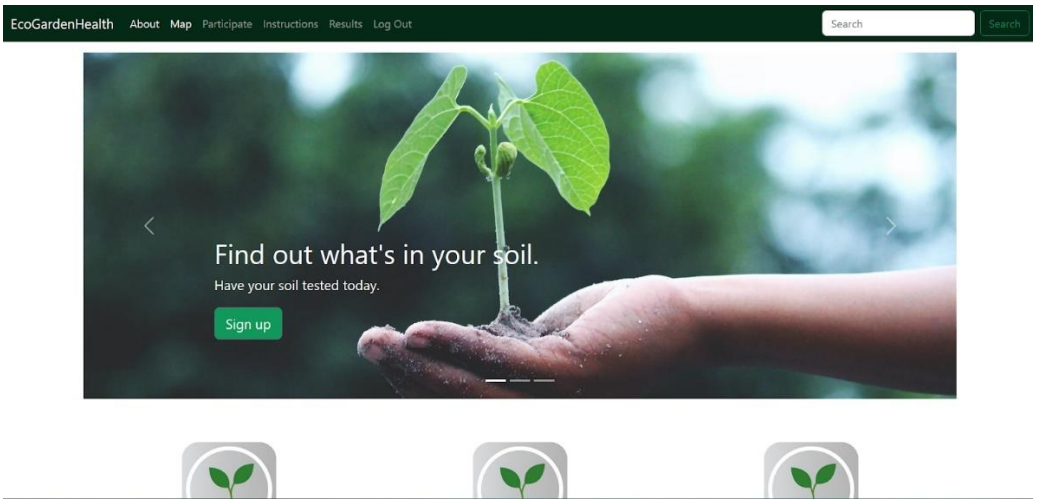
41. Bendoraitis, A., Kronika, J. (2020). *Django 3 web development cookbook* (4th ed.). Packt Publishing.

42. Ravindran, A. (2018). *Django design patterns and best practices* (2nd ed.). Packt Publishing.

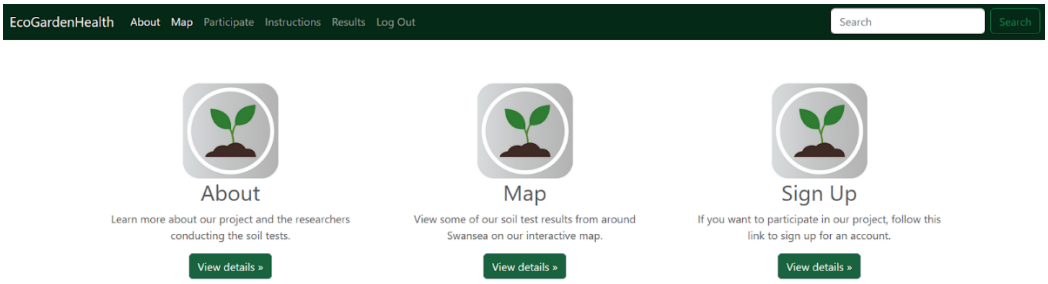
APPENDICES

Appendix 1: EcoGardenHealth Application Screenshots

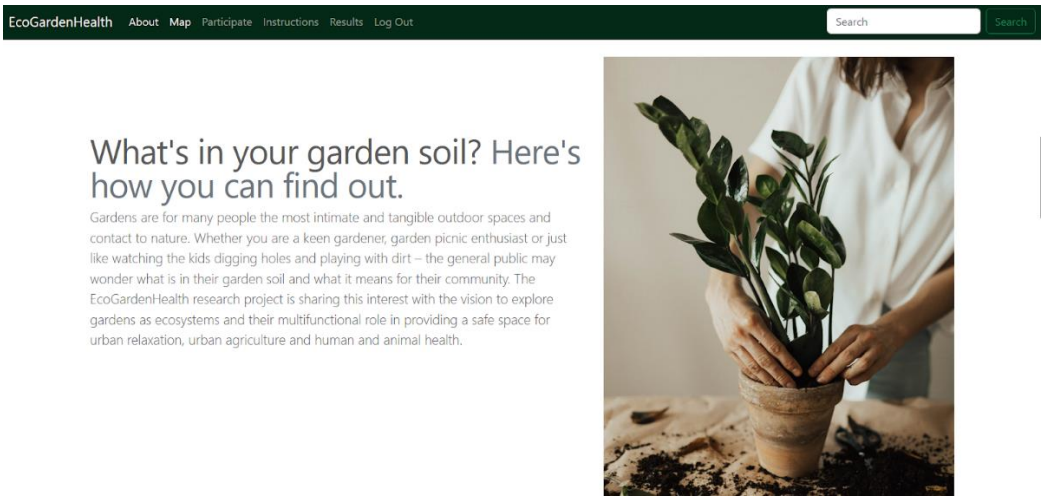
Home Page



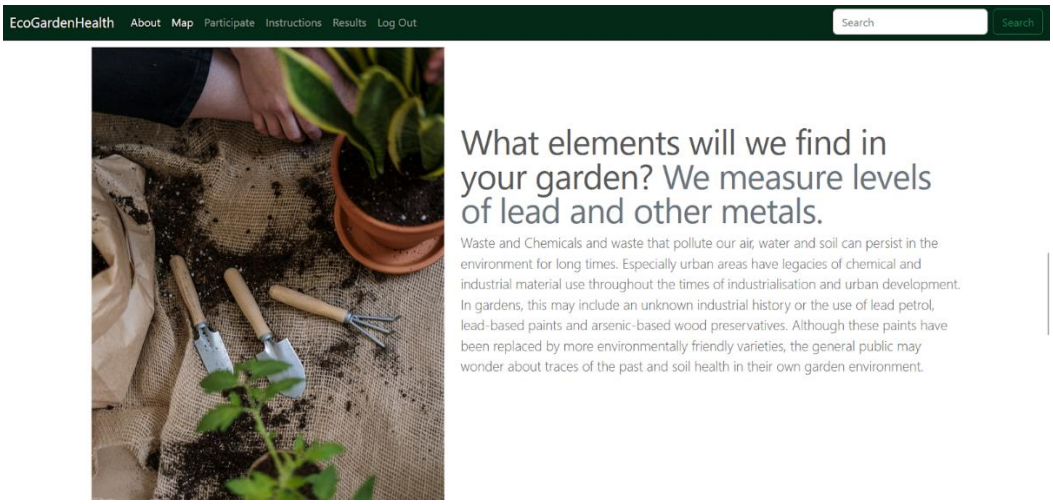
Home Page



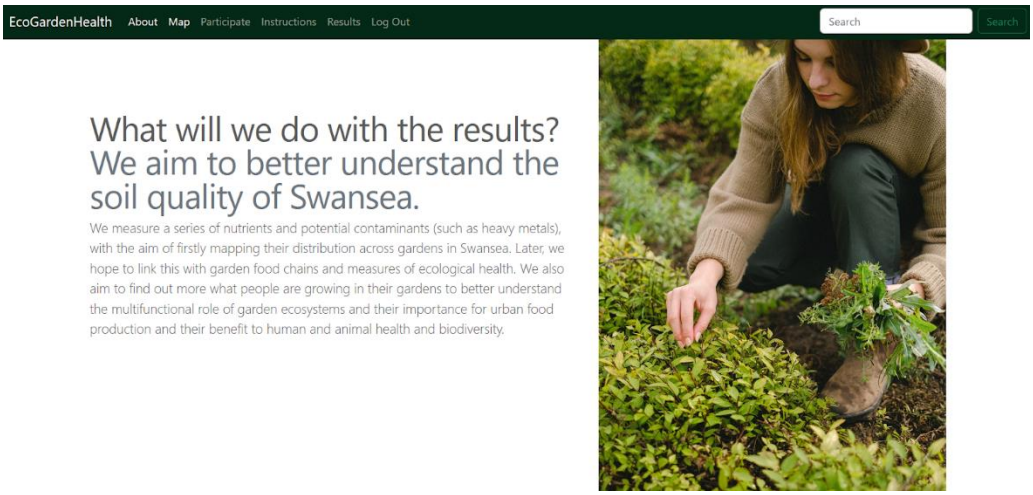
Home Page



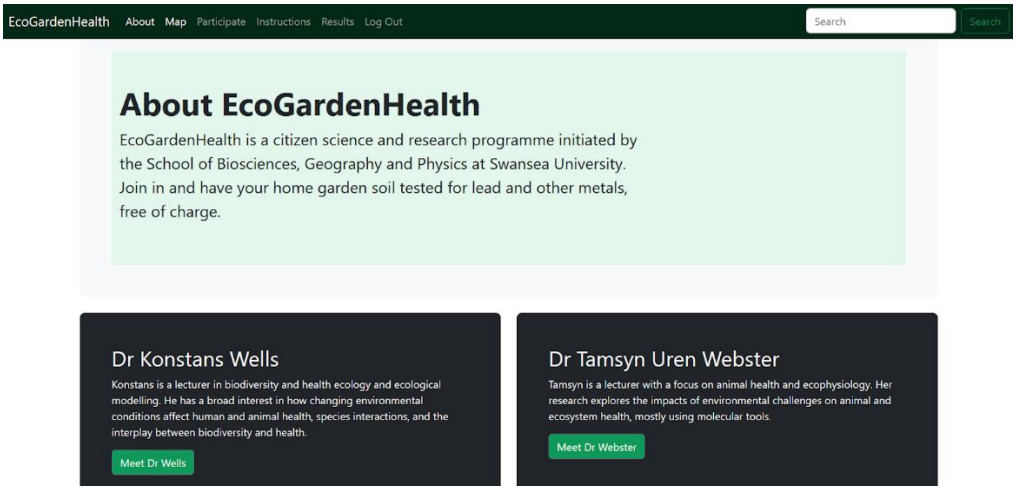
Home Page



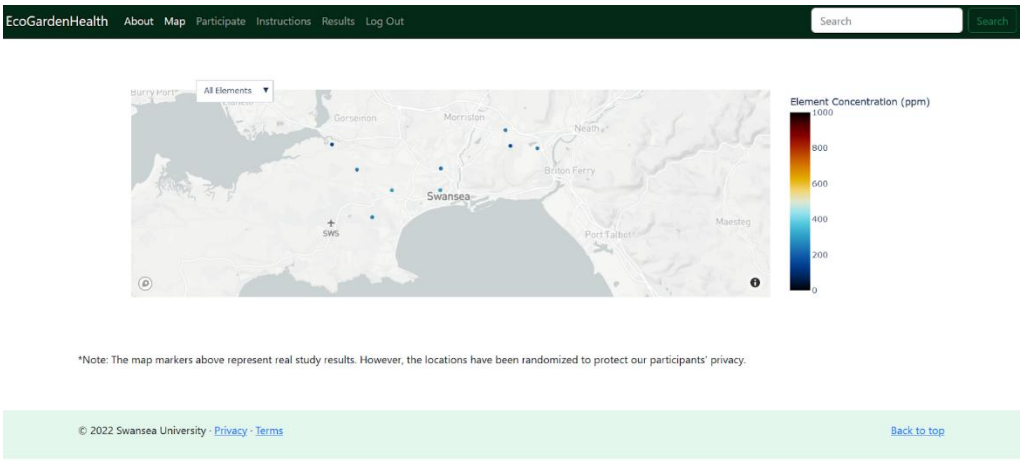
Home Page



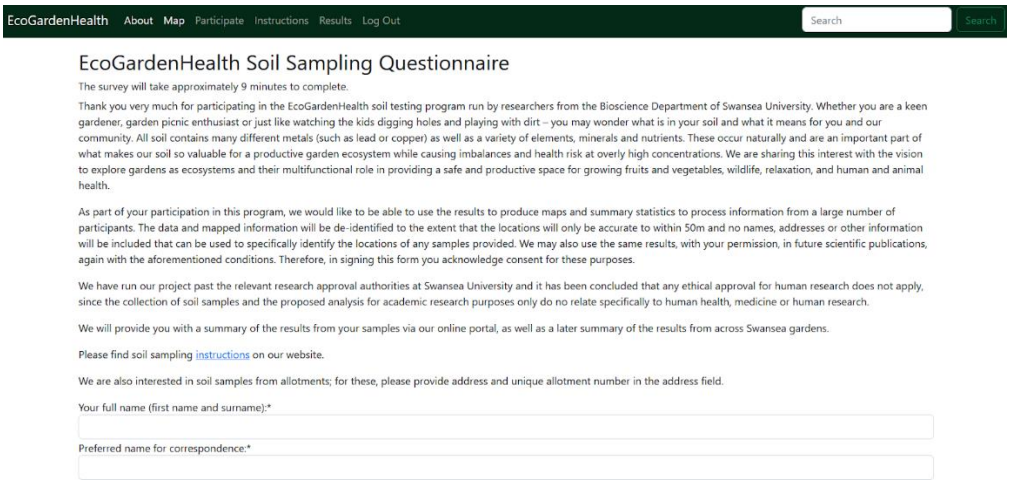
About Page



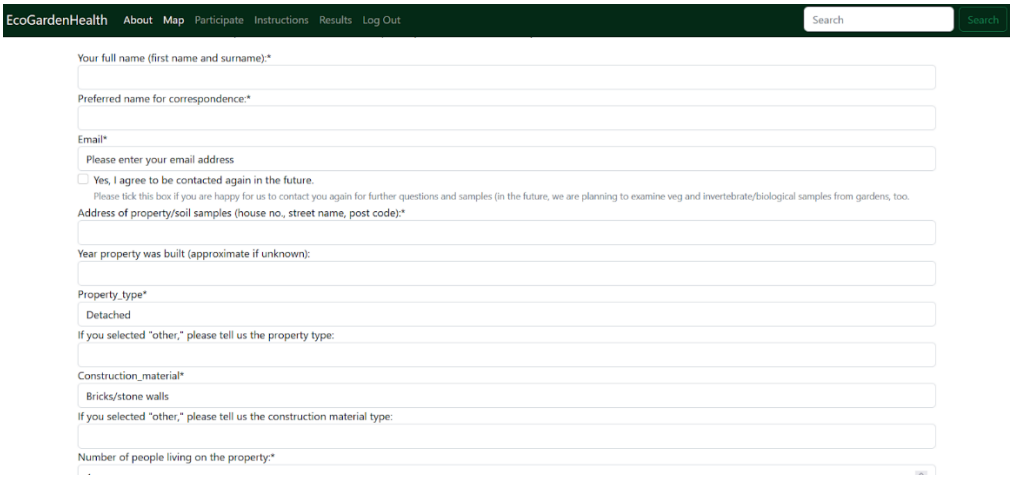
Map Page



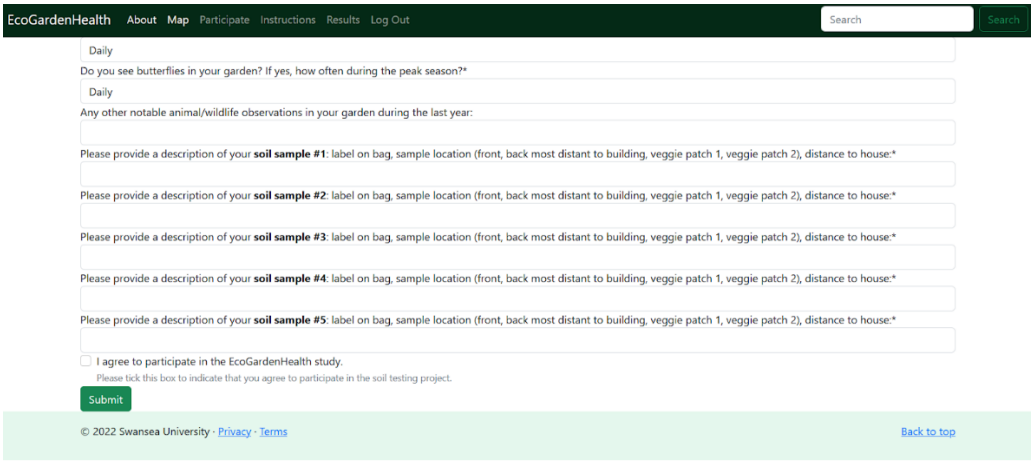
Participation Form Page



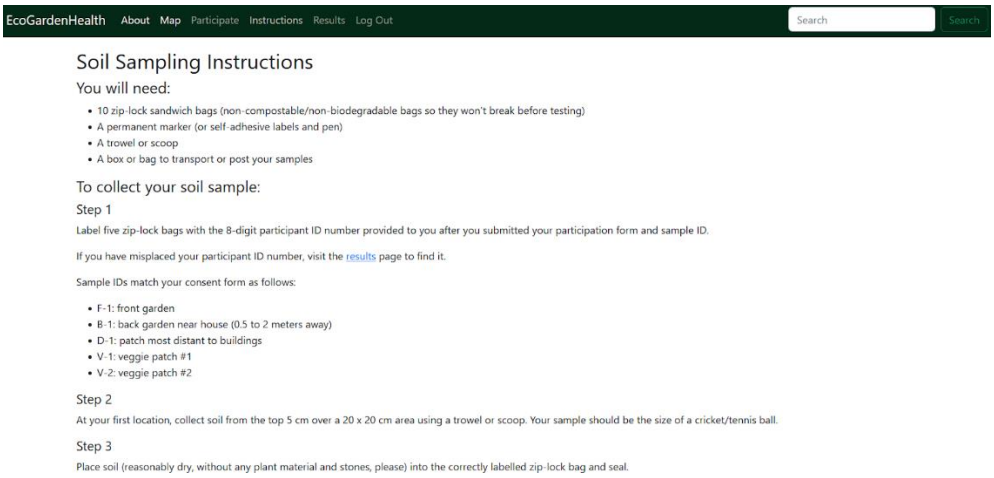
Participation Form Page



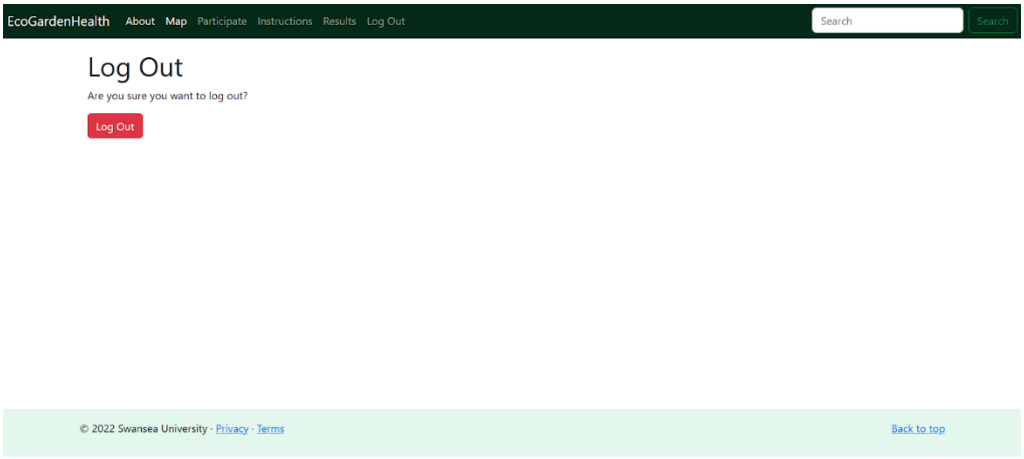
Participation Form Page



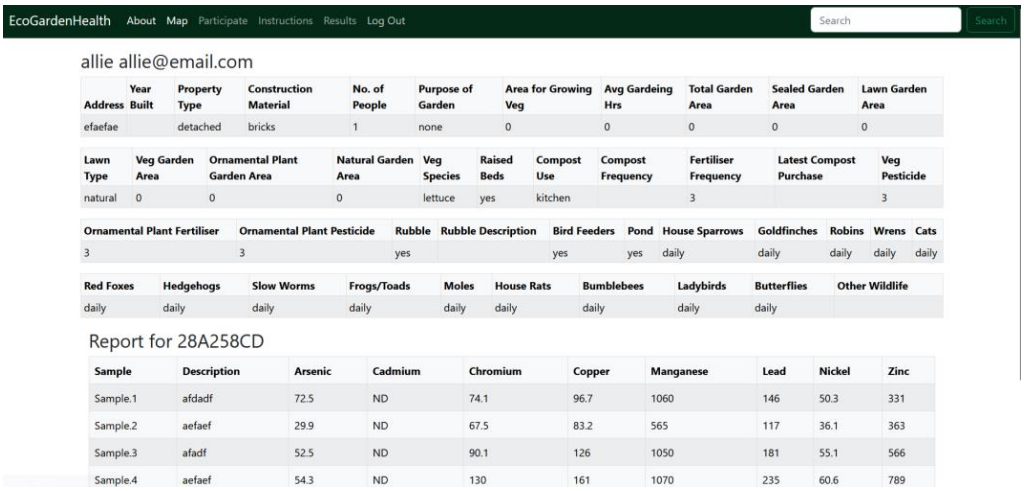
Instructions Page



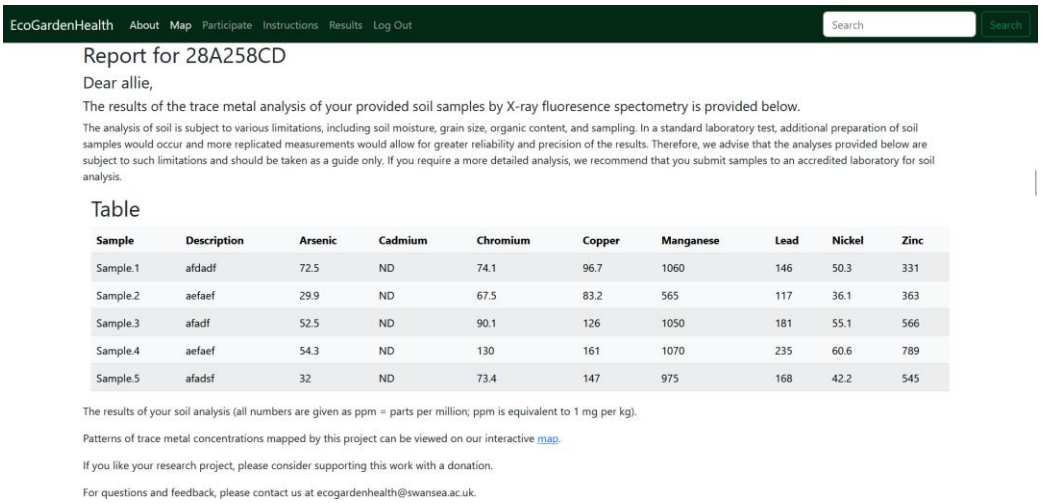
Log Out Page



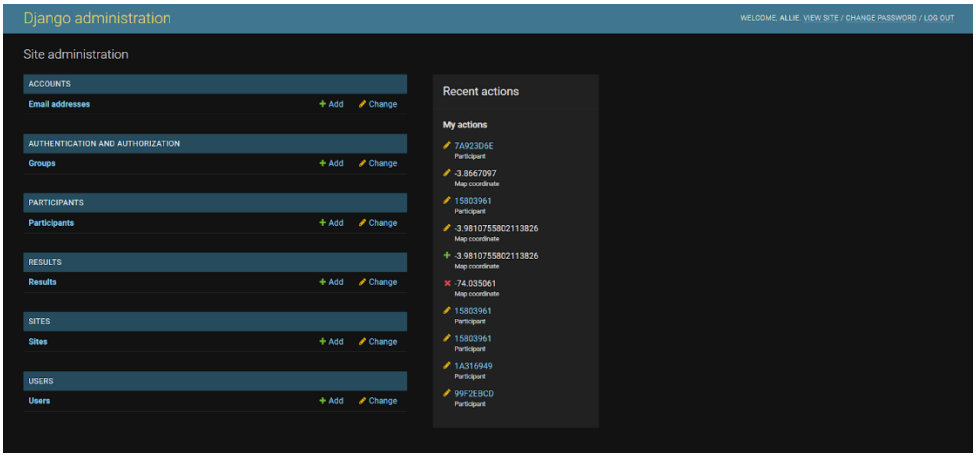
Admin Search Result Page



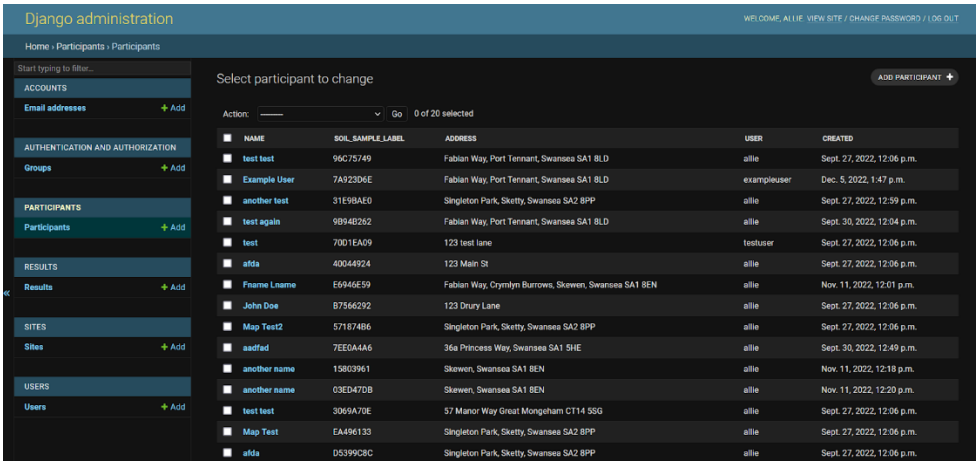
Participant Result Page



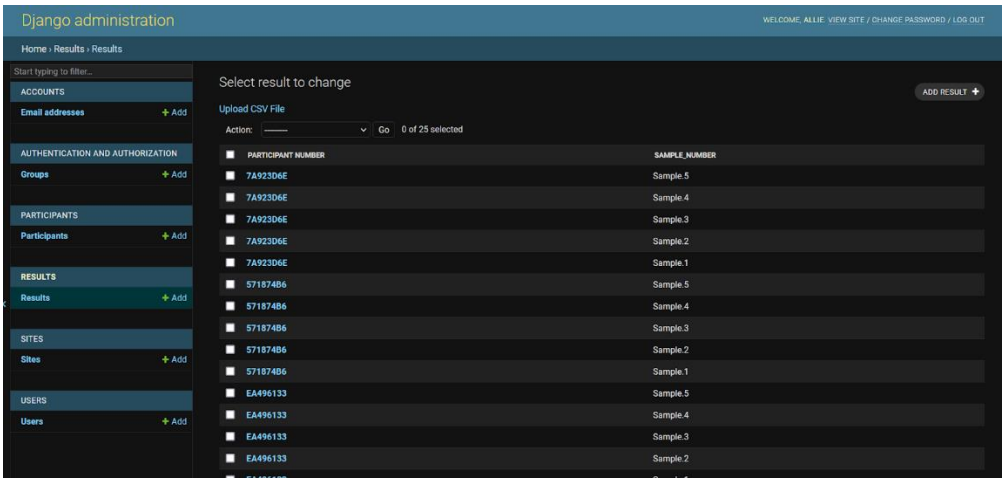
Admin Home Page



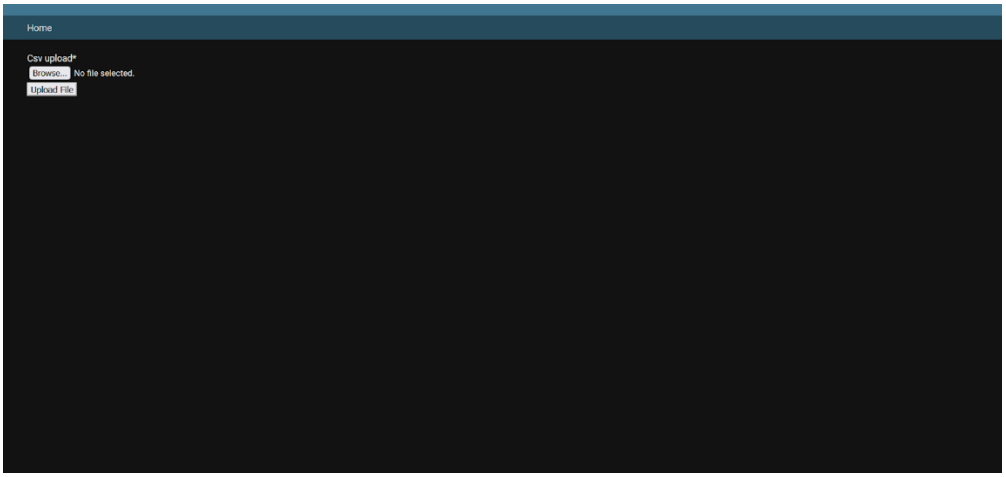
Admin Participant Model Page



Admin Result Model Page



Admin Result Upload Page



Appendix 2: Use Cases

Use Case 1

- Learn about the project

Actors

- Potential participants, researchers from other projects, Swansea residents

Basic flow

- A potential participant hears about the project from a neighbor, an email, or some other marketing technique. She searches for the project, finds the website, and learns about the project on the home and about pages. She decides to sign up to participate.
- A researcher for a separate soil testing project wants to find out about other similar projects in other areas of the world. He searches for soil testing projects and finds the website for EcoGardenHealth. He reads the about page and contacts the Swansea researchers with a collaboration idea.
- A Swansea resident is interested in finding out about the soil quality in her area. She searches the internet for information and finds the website. She is able to learn about the soil contaminant levels in the Swansea area.

Use Case 2

- Sign up for the project

Actors

- Participants

Basic flow

- A Swansea resident has learned about the project and wants to contribute to the data collection efforts. He visits the website and fills out the participation form. He then views instructions informing him how to submit his soil sample as well as an automatically generated soil sample label.

Use Case 3

- Check soil sample label number

Actors

- Participants

Basic flow

- A participant has signed up for the EcoGardenHealth project and is preparing to submit her soil sample. However, she has lost the soil sample label generated by the web portal and is unsure what to write on her sample bags. She logs into the web portal and is presented with her soil sample label, which she can then write on her sample bags for submission.

Use Case 4

- Check the status of a participant’s results

Actors

- Participants

Basic flow

- A participant has submitted a soil sample and wants to check the status of his results. He visits the web portal, signs into his account, and views his previously submitted samples and their test results.

Use Case 5

- View visualized results of the project

Actors

- Participants, Swansea residents, other researchers

Basic flow

- A participant, Swansea resident, or researcher involved in a similar project is interested to see the results of the EcoGardenHealth project. He navigates to the website and views the map page. The map displays soil sample test results in an interactive and pleasant visualization. The result locations are randomized to protect the living locations of the participants.

Use Case 6

- Collect and view participant information

Actors

- EcoGardenHealth researchers

Basic flow

- Researchers need to keep track of the participants involved in the study and collect information relevant to the project. They can visit the admin page of the web portal to view questionnaire answers provided by participants when they complete the participant form. They can also visit the search page in the main web portal to search participant information and related test results by participant soil sample label number.

Use Case 7

- Disseminate soil sample test results to participants

Actors

- EcoGardenHealth researchers

Basic flow

- Researchers complete soil sample tests and need to provide participants with their results. The soil testing machine outputs a CSV file with results broken down by element. The researcher visits the admin page of the web portal and uploads this file, with the participant soil sample label included, and the results are automatically parsed into the relational database. The information is then automatically populated into the relevant participant's web portal under their specific login information.

Use Case 8

- Provide information about the project to interested parties

Actors

- EcoGardenHealth researchers

Basic flow

- The researchers want to provide information about their project to potential participants, Swansea residents, researchers involved in related projects, and other stakeholders. They add relevant information about their project to the home and about pages on the website, which are available for the general public to see.

Use Case 9

- Visualize project result data

Actors

- EcoGardenHealth researchers

Basic flow

- The researchers want to show the data collected in their project in an attractive map-based visualization without revealing sensitive location information about their participants. An algorithm automatically randomizes the location of the soil sample before displaying the information on an interactive map. The map displays full results as well as single-element results.

Use Case 10

- Edit participant questionnaire responses

Actors

- EcoGardenHealth researchers

Basic flow

- The researchers have been informed that a participant has accidentally provided incorrect information in their questionnaire. They log into the admin version of the website, find the correct participant, and update the collected data.

Use Case 11

- Store and organize project data in a secure database

Actors

- EcoGardenHealth researchers

Basic flow

- The researchers want a secure and convenient way to store and organize data collected in their project. Participant data is collected and stored in a relational database through a questionnaire on the web portal. Soil sample result data is stored in the same relational database when CSV soil test results are uploaded by the researchers.

Use Case 12

- Translate information on the web application to Welsh or another non-English language

Actors

- Participants and other website visitors

Basic flow

- A potential participant visits the website and wants to learn more about the project. She is comfortable with English but would have a better understanding if she could read the information in Welsh. She switches to a Welsh version of the website and has an easier time reading the information

Use Case 13

- The web application is responsive and automatically adapts to screens of every size

Actors

- Participants, researchers, and other website visitors

Basic flow

- A project participant wants to fill out the participation form but is away from his desktop computer. He visits the web application on his smartphone's browser and is able to access the form and fill it out without the need for excessive zooming or scrolling.

Use Case 14

- The application is optimized for screen readers

Actors

- All website visitors

Basic flow

- An interested party wants to learn more about the project but she has a disability that forces her to rely on a screen reader to have content read aloud to her. She visits the EcoGardenHealth web application and there are no issues with her third-party screen reader. She is able to have the content read aloud to her.

Use Case 15

- The application securely stores users' data to protect it from malicious actors

Actors

- Participants

Basic flow

- A potential participant wants to submit the participation form but is worried about submitting his personal data. He submits the form anyway. A malicious actor tries to steal the data from the web application but is prevented from doing so by web security measures.

Use Case 16

- The user wants to use third-party login to register

Actors

- Participants

Basic flow

- A new participant wants to sign up for the project, but she does not want to have to remember a new password. She uses third-party authorization to use her social media login credentials to create a user account on the web application.

Appendix 3: Functional Software Requirements

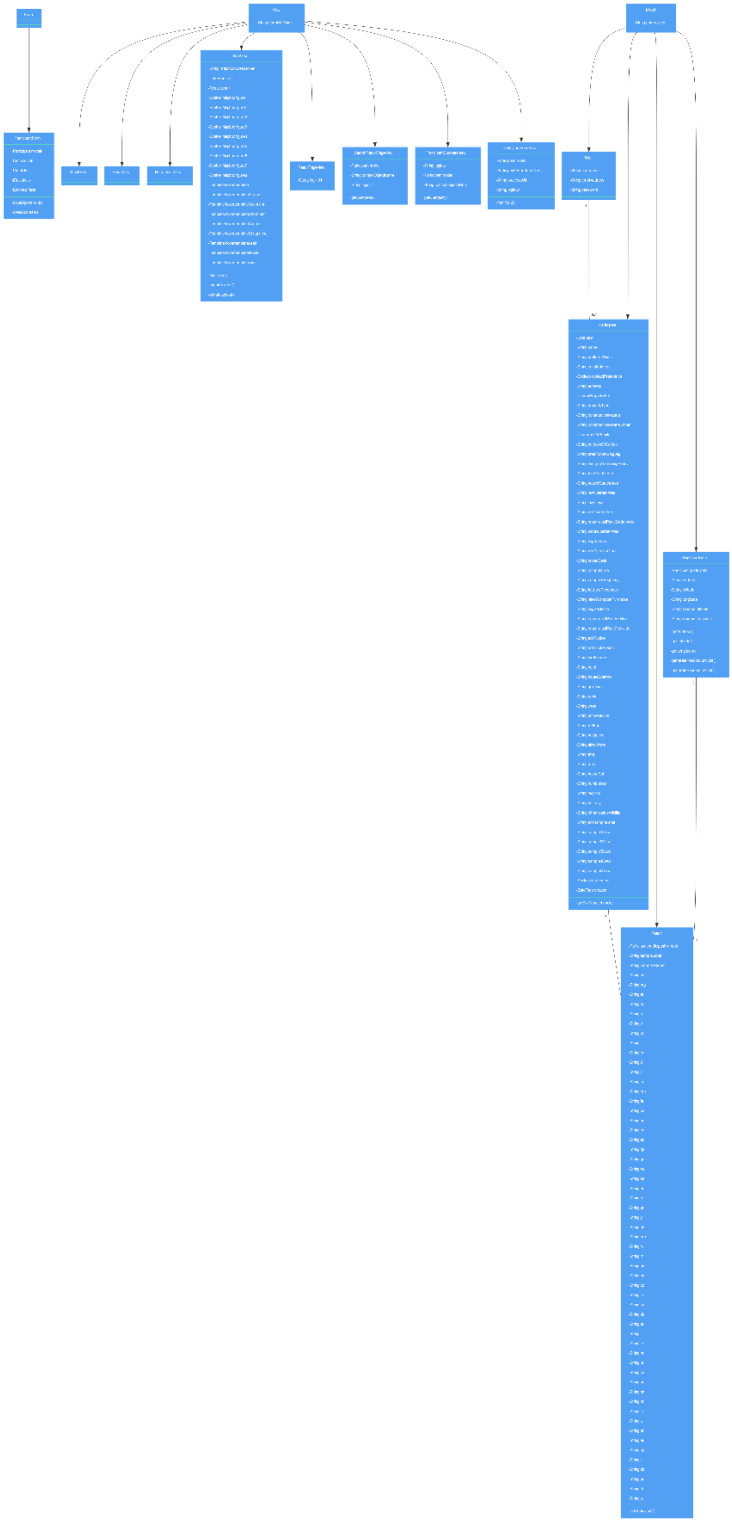
- The website has home and about pages that provide information about the project and are visible to all website visitors.
- The website has a registration/sign in page that is visible to all visitors.
- The registration page allows new users to register a user account using an email address and password.
- The login page allows registered users to log into their accounts.
- Users can register and log in using optional third-party authentication.
- The built-in authentication service includes password reset functionality.
- The website has a participation form that is only available to authenticated users.
- The participation form contains a list of questions created by the research team that can be filled out by authenticated users.
- The participation form can only be submitted when all required fields are filled out and the agreement box has been checked.
- The participation form tracks which user is logged in when it is filled out and stores the information in the database.
- The data in the submitted participation form is stored in the relational database.
- Upon form submission, the website automatically generates a random and unique soil sample label number and displays it to the user.
- The website has a results page that is only available to authenticated users.
- The results page shows soil sample label numbers and pending and completed results for the currently logged in user.
- The results page displays relevant results in a table format that is retrieved from researcher-uploaded data in the relational database.
- The website includes a search feature that is only available to authenticated and authorized administrative users.
- The search feature allows administrative users to search for user participation form data and soil testing results via the unique soil sample label number.
- Administrative users have access to an admin version of the web application, which allows them to view and edit all participant data.
- Administrative users can upload test result data in the form of a CSV file via the admin version of the website.
- The CSV file is automatically parsed and stored in the relational database when it is uploaded.
- Soil sample test results are displayed on an interactive map page that is visible to all website visitors.
- Participant sample location information is automatically converted into latitude and longitude, randomized within the Swansea area, and displayed on an interactive map.

- The map displays data as overall results and as results broken down by specific elements.
- The map is scrollable and zoomable.
- The web application can be translated into multiple languages.
- The web application has a Welsh translation option.
- The web application is optimized for screen readers.
- The web application is responsive and can be used on devices with screens of varying sizes.
- The web application passes all written tests.

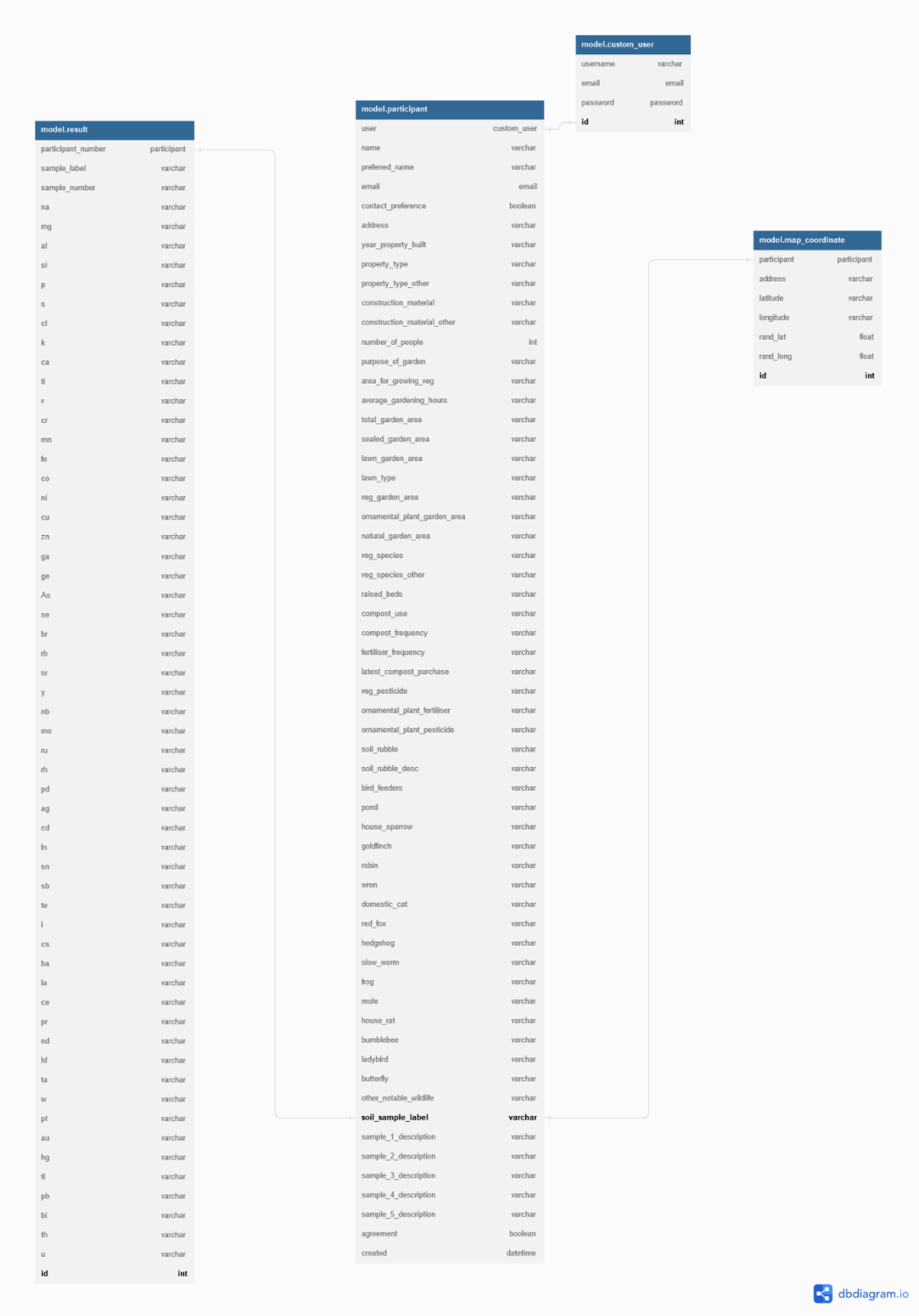
Appendix 4: Nonfunctional Software Requirements

- The web application user interface is visually appealing and easy to use.
- The design of the web application is as simple as possible to allow all users to navigate it easily.
- The web application is accessible to users of all abilities.
- The map feature is easy to use and understand.
- The web application is scalable and easy to maintain, making use of object-oriented code.
- The database design is secure, scalable, and flexible.
- The web application is secure and deployment-ready before it is put online.
- The web application adheres to local regulatory and data privacy laws.

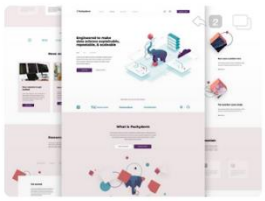
Appendix 5: UML Diagram

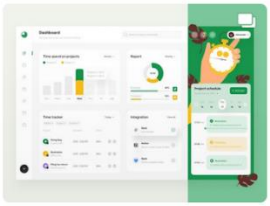


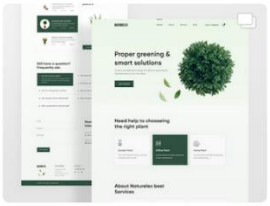
Appendix 6: Entity Relationship Diagram

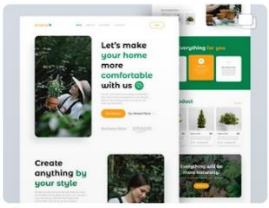


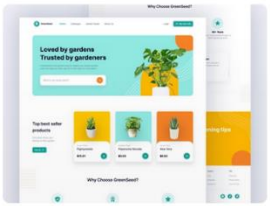
Appendix 7: Dribbble User Interface Design Inspiration

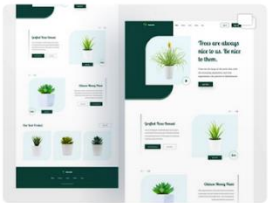


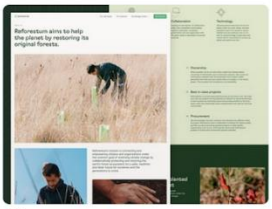


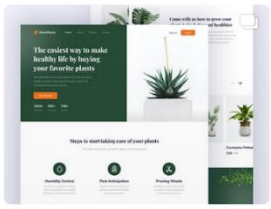


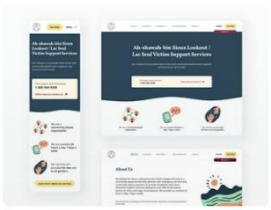


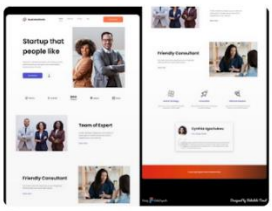


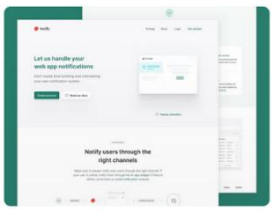










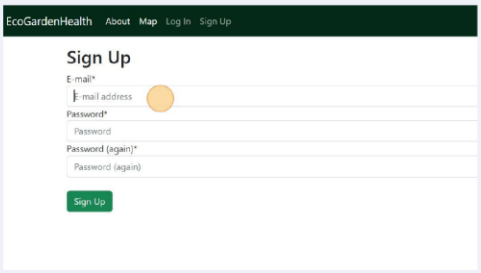


Appendix 8: Application Handbooks

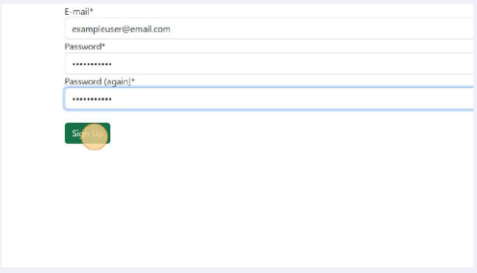
User Handbook: Participant

1 Navigate to http://127.0.0.1:8000/accounts/signup/

2 Click the "E-mail*" field.

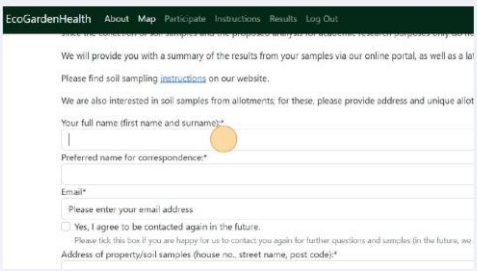


3 Click "Sign Up"

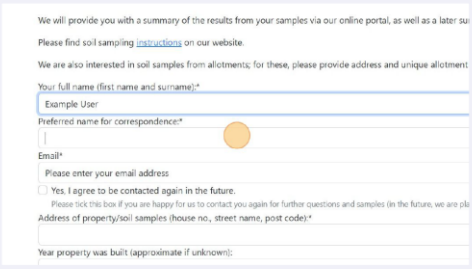


4 Click "Participate"

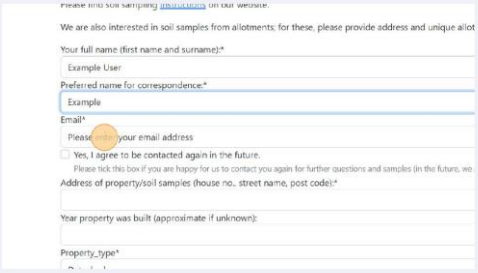
5 Click the "Your full name (first name and surname):*" field.



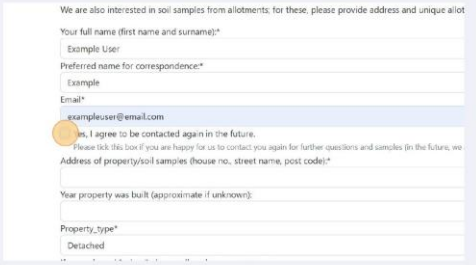
6 Click the "Preferred name for correspondence:*" field.



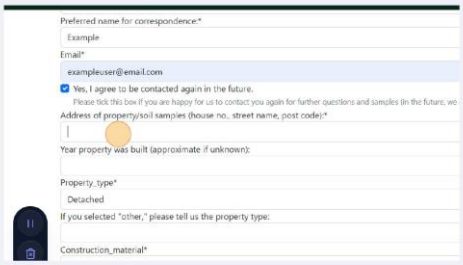
7 Double-click the "Email*" field.



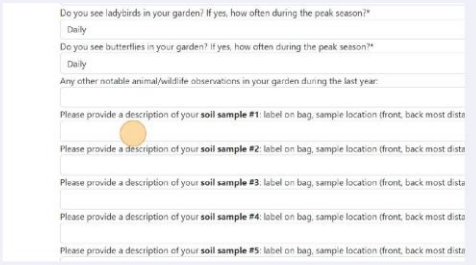
8 Click the "Yes, I agree to be contacted again in the future." field.



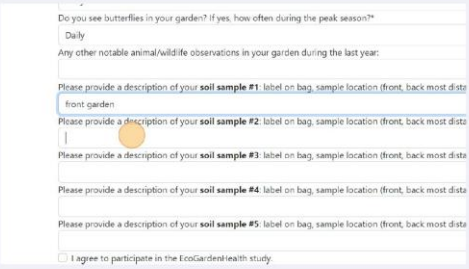
9 Click the "Address of property/soil samples (house no., street name, post code):*" field.



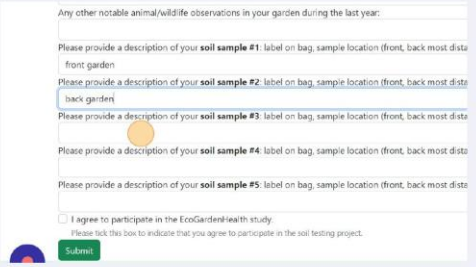
10 Click the "Please provide a description of your soil sample #1: label on bag, sample location (front, back most distant to building, veggie patch 1, veggie patch 2), distance to house:*" field.



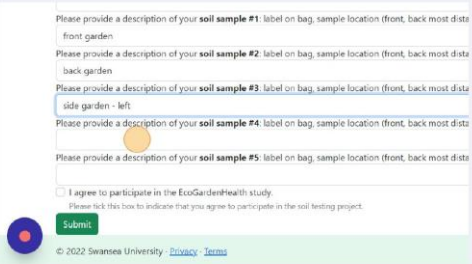
11 Click the "Please provide a description of your soil sample #2: label on bag, sample location (front, back most distant to building, veggie patch 1, veggie patch 2), distance to house:*" field.



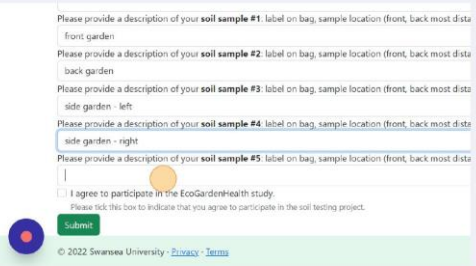
12 Click the "Please provide a description of your soil sample #3: label on bag, sample location (front, back most distant to building, veggie patch 1, veggie patch 2), distance to house:*" field.



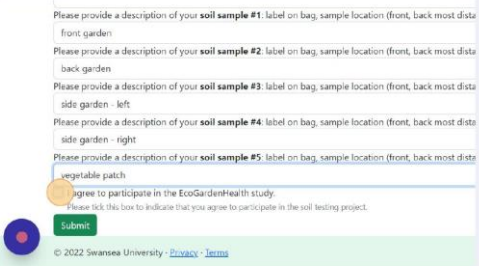
13 Click the "Please provide a description of your soil sample #4: label on bag, sample location (front, back most distant to building, veggie patch 1, veggie patch 2), distance to house:*" field.



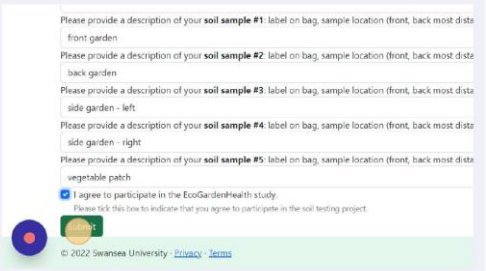
14 Click the "Please provide a description of your soil sample #5: label on bag, sample location (front, back most distant to building, veggie patch 1, veggie patch 2), distance to house:*" field.



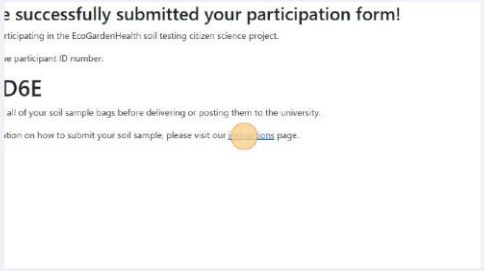
15 Click the "I agree to participate in the EcoGardenHealth study." field.



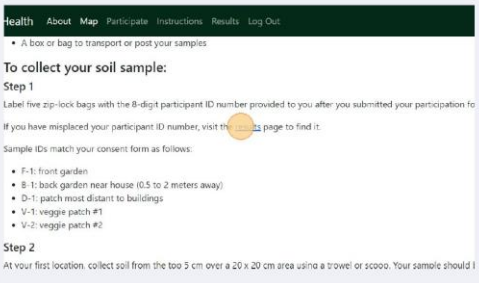
16 Click "Submit"



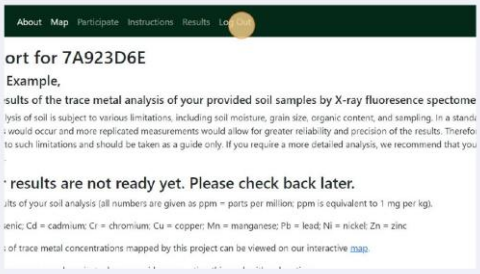
17 Click "instructions"



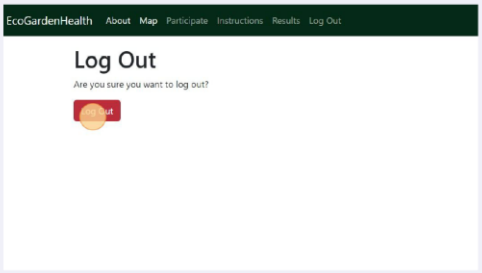
18 Click "results"



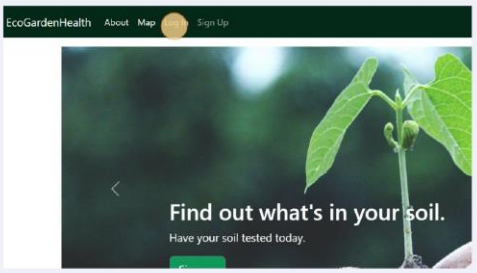
19 Click "Log Out"



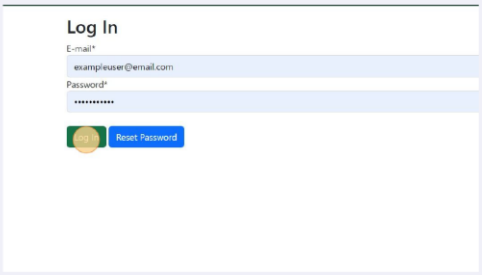
20 Click "Log Out"



21 Click "Log In"



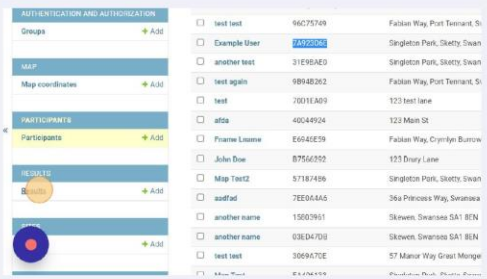
22 Click "Log In"



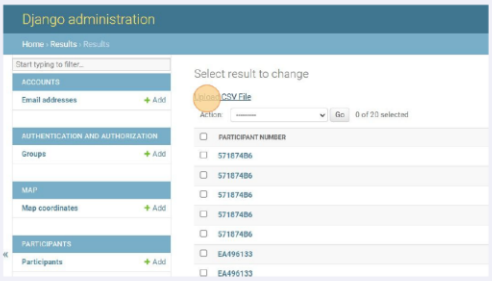
User Handbook: Admin

1 Navigate to http://127.0.0.1:8000/admin/

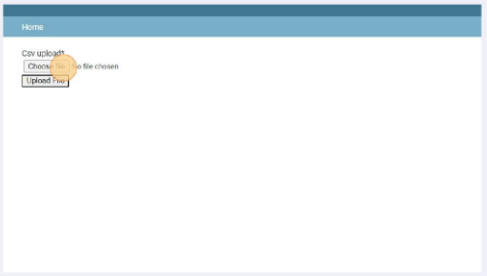
2 Click "Results"



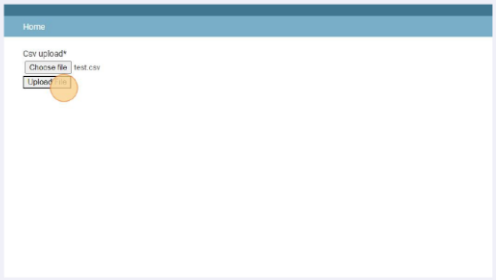
3 Click "Upload CSV File"



4 Click the "Csv upload*" field.

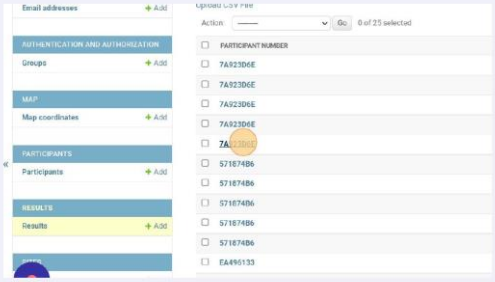


5 Click "Upload File"

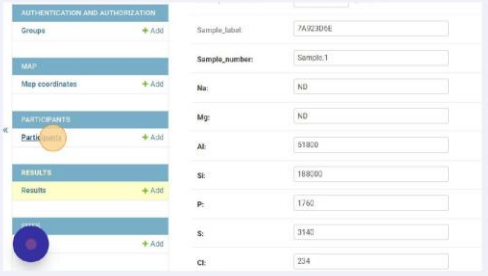


6 Click "Results"

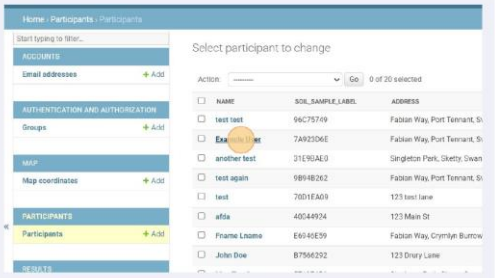
7 Click "7A923D6E"



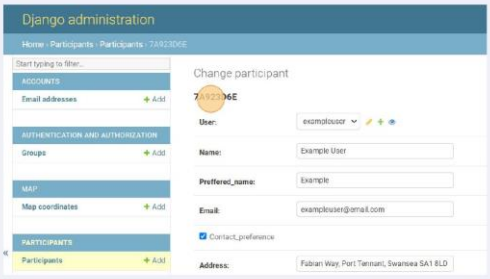
8 Click "Participants"



9 Click "Example User"



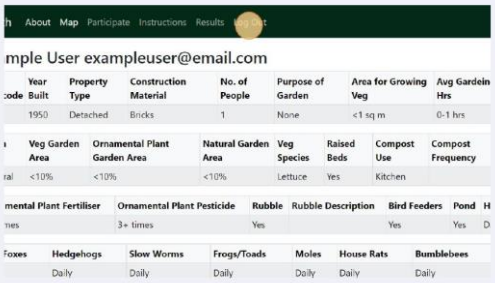
10 Double-click "7A923D6E"



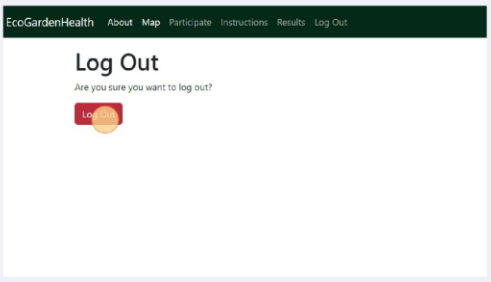
12 Click the "Search" field.

14 Click "Search"

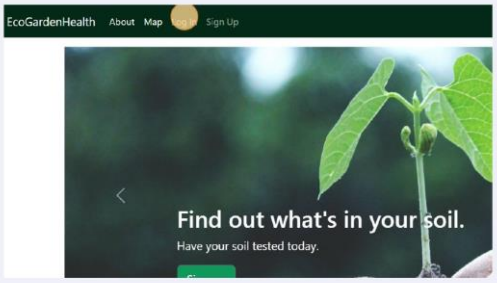
15 Click "Log Out"



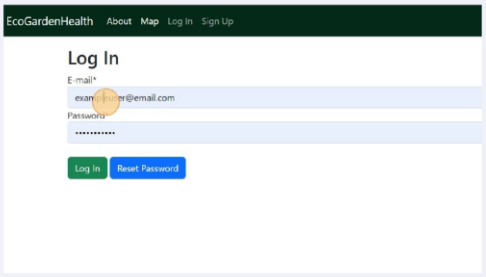
16 Click "Log Out"



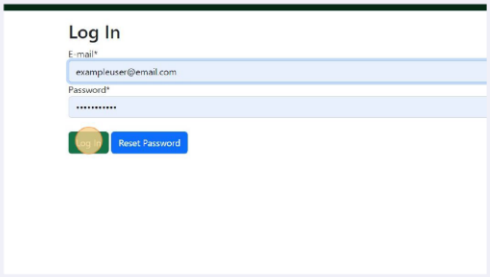
17 Click "Log In"

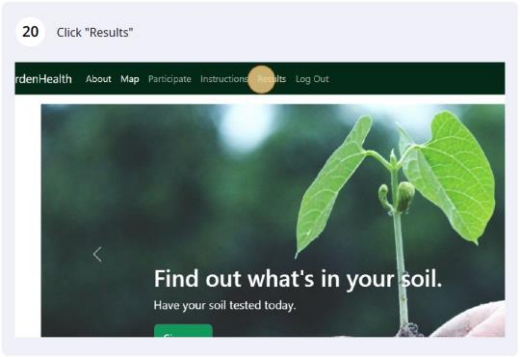


18 Click the "E-mail*" field.

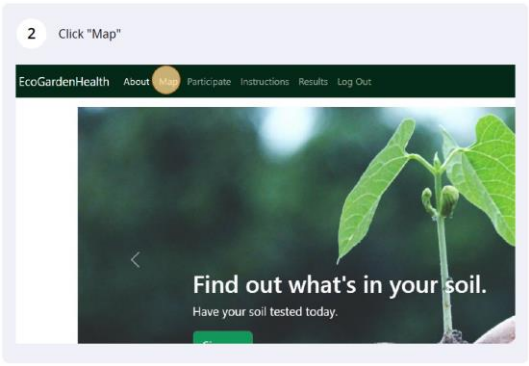


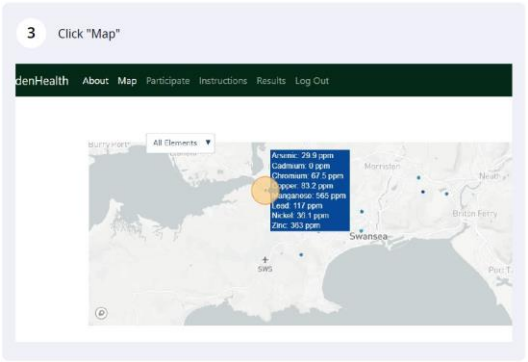
19 Click "Log In"

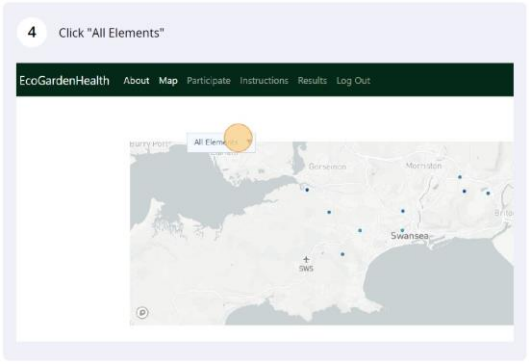


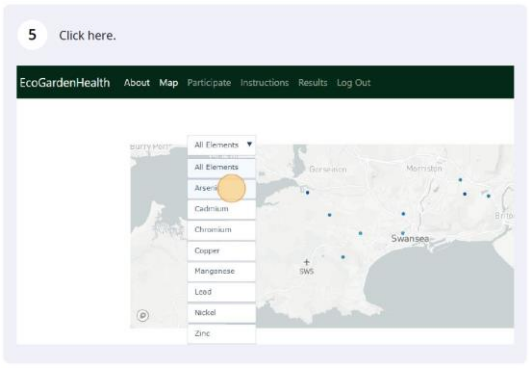


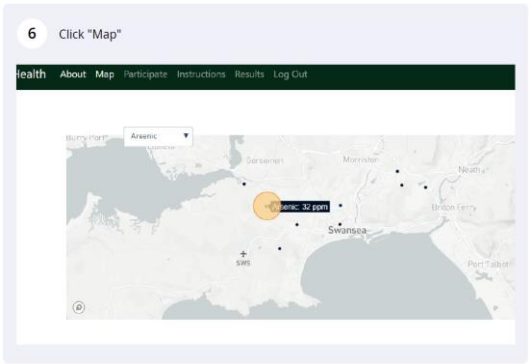
User Handbook: Map

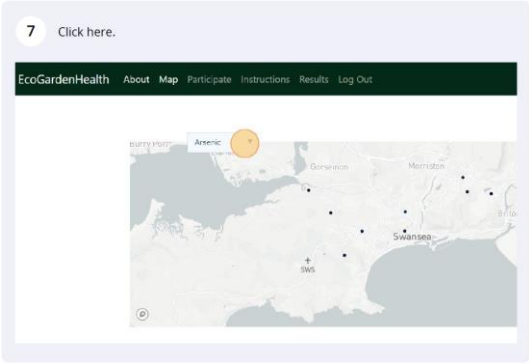




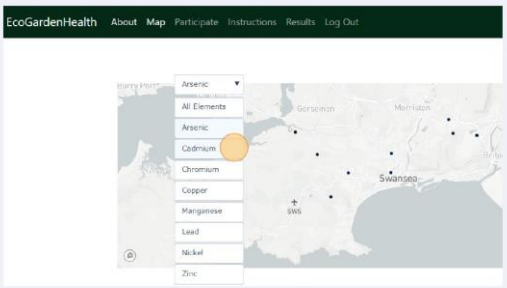




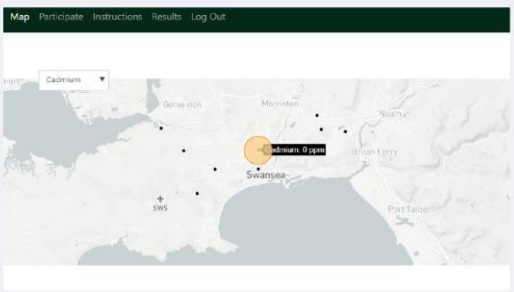




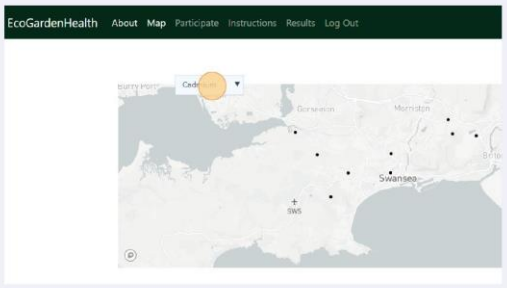
8 Click here.



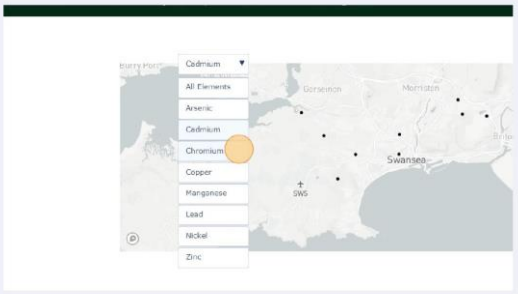
9 Click "Map"



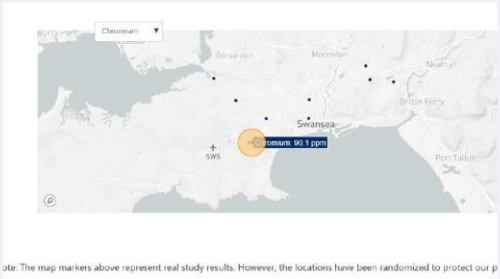
10 Click "Cadmium"



11 Click here.

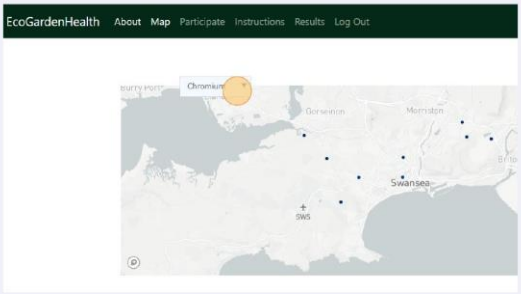


12 Click "Map"

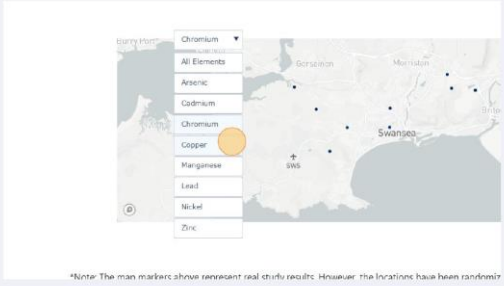


ote. The map markers above represent real study results. However, the locations have been randomized to protect our p

13 Click here.

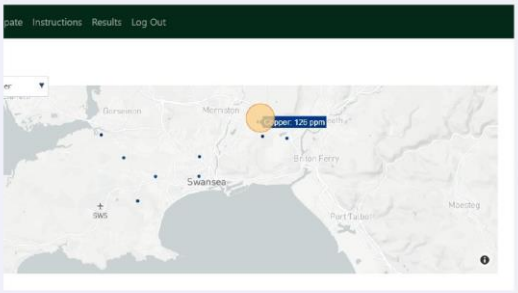


14 Click here.

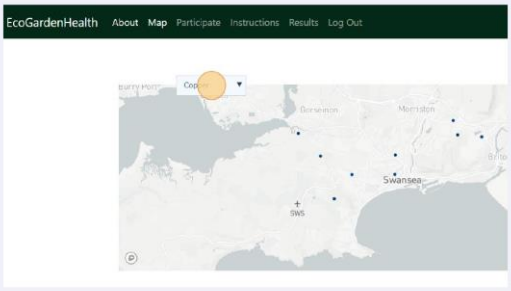


*Note: The map markers above represent real study results. However, the locations have been randomiz

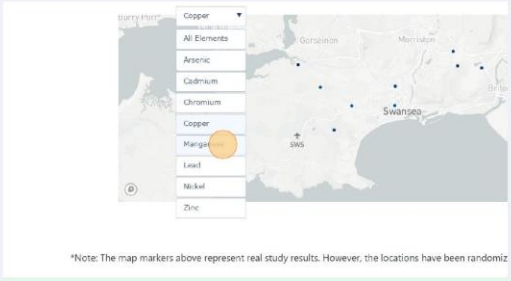
15 Click "Map"



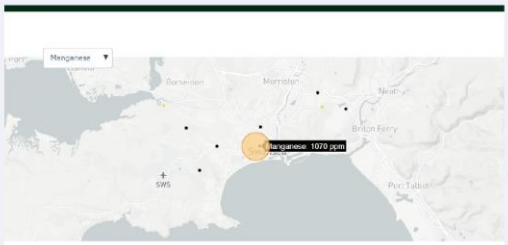
16 Click here.



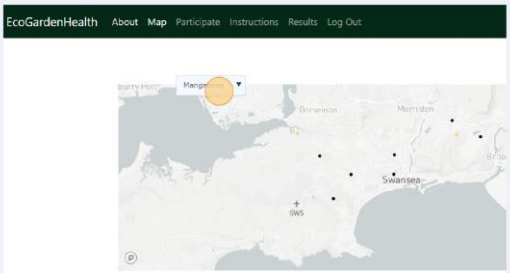
17 Click "Manganese"



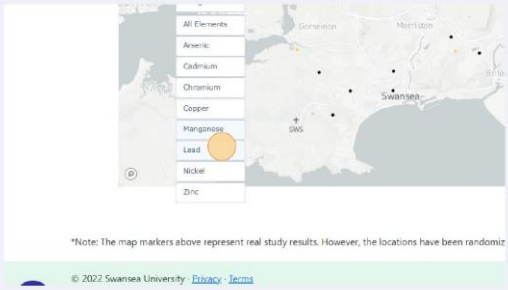
18 Click "Map"



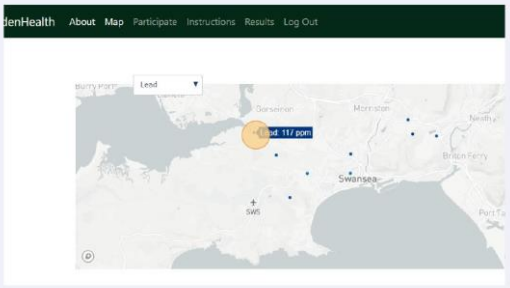
19 Click here.



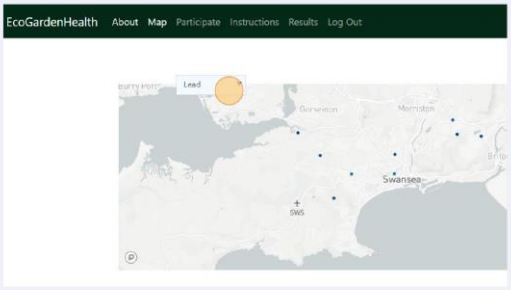
20 Click here.



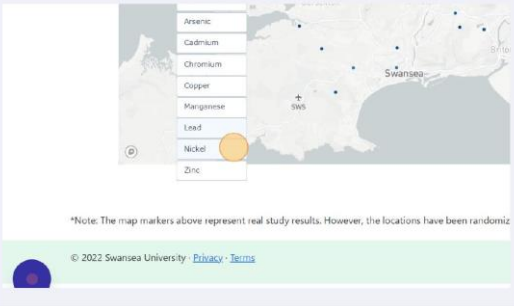
21 Click "Map"



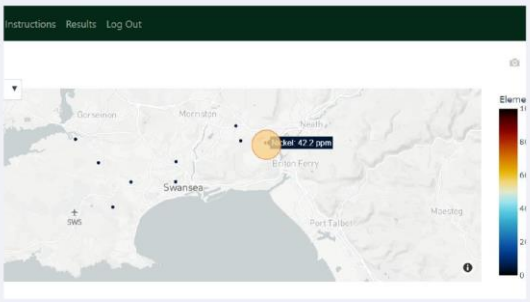
22 Click here.



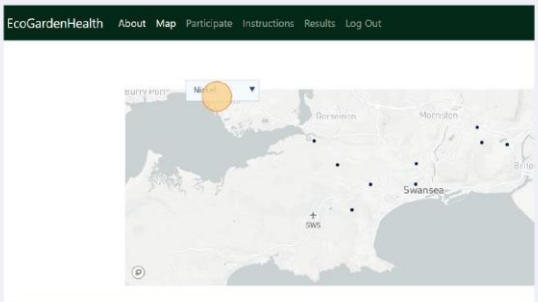
23 Click here.



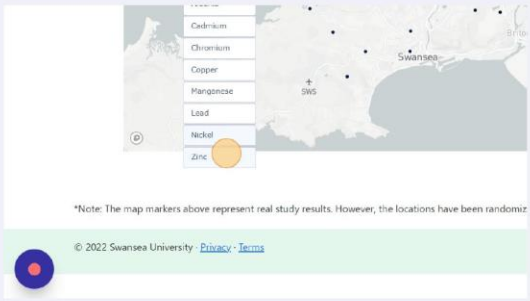
24 Click "Map"



25 Click here.



26 Click here.



27 Click "Map"