

Final Report

Major Project Presented in Fulfilment of the
Internet Computing and Systems Administration Bachelor's
Degree

David Stumbra (dec21)

Department of Computer Science
Undergraduate Dissertation
Under the supervision of Edel Sherratt
Aberystwyth University
2020

I would like to dedicate this to my Mum.

She supported me throughout my life and encouraged me to go to University.

Crustacean Database Project

Table of Contents

1. Abstract.....	4
2. Acknowledgements.....	5
3. Background	6
3.1. Project Description.....	6
3.1.1. Introduction	6
3.1.2. Functionality	6
3.1.3 Related websites	7
4. Approach to project	9
4.1. Technology	9
4.1.1. Proposed	9
4.1.2. Complexity of the website	9
4.2. Project plan	9
5. Analysis	10
5.1. Business Analysis.....	10
5.1.1. Introduction	10
5.1.2. Stakeholders	11
5.1.3. Users and use cases	12
5.1.4. Typical users.....	13
6. UI and Design Evaluation	14
6.1. User Interface (UI).....	14
6.1.1. Introduction	14
6.1.2. User and technological needs.....	14
6.1.3. Website structure	15
6.1.4. Evaluation	25
6.2. Database design.....	26
6.2.1. Tables	26
<i>system:</i>	26
<i>users:</i>	26
<i>pass_resets:</i>	27
<i>records:</i>	27
<i>images:</i>	28
6.2.2. Sample data	29
6.3. Addressing security concerns	31

7. Implementation	32
7.1. Planning.....	32
7.1.1. Proposed technology overview	32
7.1.2. Schedule and development flow	33
7.2. Changes.....	34
8. Achievements.....	35
9. Testing and Evaluation.....	40
9.1. Self-testing	40
9.2. User testing.....	41
9.3. Testing summary.....	43
9.2.3. Ethics.....	43
9. Critical analysis.....	45
10. Appendices.....	46
10.1. Appendix A – Data Dictionary	46
10.2 References	48

1. Abstract

The website is being requested by Bernie Tiddeman and the research team he is involved in. The research team is comprised of researchers from Aberystwyth University and Bangor University. It is also sponsored by the Welsh Government, who have an input in the project.

At first, I had a few meetings with Bernie, and another researcher, where we discussed key functions of the project. The initial summary had a lot of requirements, which were not feasible in the amount of time I had. We then discussed which systems are primary and which are secondary (i.e. which are required, and which I could implement if I had time left at the end). Afterwards, I started investigating potential libraries and systems which I could use to implement the required functionality.

In the business analysis, I looked at the requirements set out for the project, as well as different “stakeholders” i.e. people who will be using the system a lot. I came up with different use cases for those people, looking at how they would individually use the system.

Following that, I started the UI (User Interface) design and evaluation, where I came up with a design, I then had testers evaluate that design, and provide feedback.

During the technical design stage, I looked at creating a database, and evaluating the prototype. This gave me information on how I should approach my implementation.

During the implementation, I discussed stages of the implementation, and came up with a brief timeline and a work schedule. I also wrote about an overview of the technologies used for the project.

I mentioned the different changes that I made during the development and explained why I made those decisions. Afterwards there is a section about the biggest achievements I made during the development.

Finally, there are testing and critical analysis stages, where I wrote about the different tests that have been conducted. During the critical analysis I reflected on what went well, and what I could have done better.

2. Acknowledgements

I wish to thank my friend of 10 years, Marius, who provided a lot of support with CSS and the general design during the development of the project.

I would like to thank all the lecturers at the Department of Computer Science, who have provided guidance and support throughout the three years of my life I have spent at Aberystwyth University. Lots of different lecturers have supported and helped me, but Angharad Shaw [1] was the one I felt like I could always talk to.

I would also like to thank the client, for allowing me to create a website for them and for providing sample data to test with.

A big thank you to all my friends, who supported me, and helped me during these tough 3 years at the University.

3. Background

The project was created for the client, Bernie Tiddeman. He is working on a fisheries research project at Aberystwyth University, alongside Bangor University and the Welsh Government. The project aims to capture and generate large quantities of high-quality fisheries data using video capture technology in a cost and time effective manner.

The hardware would be mounted on fishing vessels, and the data captured would be stored in a database, which would be accessible to select people.

3.1. Project Description

3.1.1. Introduction

For my project, I will be making a website, aiming to create a suitable system to display the captured data. The final website will be designed to look as professional, and to cover most of the functional requirements specified by the client.

The client has requested a website which will show the data produced by their hardware. This data will be stored in a backend database and will be accessible through the visual frontend website. In other words, the website will act as a graphic way of accessing and interpreting the data without having to navigate the technicalities of learning how to use the database directly.

The website will be accessible by select people because the data can include *sensitive* information, like coordinates which some fishermen would like to keep private.

3.1.2. Functionality

The website will contain various features and functions. The main two functions will be login, and data display. The users will be able to login and see the catch data presented visually. Logged in users will also be able to edit their account details, such as name, email address and password, as well as see any announcements written by the administrators. For example, if there is planned maintenance, it is easier to display a notification on the user's homepage, than emailing every single user.

There will also be a contact form, which will allow everyone to contact the designated person for various reasons. This could be done to report a fault, or for people wanting to access the database, to ask for an account. As well as a multitude of other reasons.

3.1.3 Related websites

The website I will produce will be quite niche, and not many other websites like this exist, or at least they are not accessible to the public. At its core, the website is an image gallery, which provides extra information about the images.

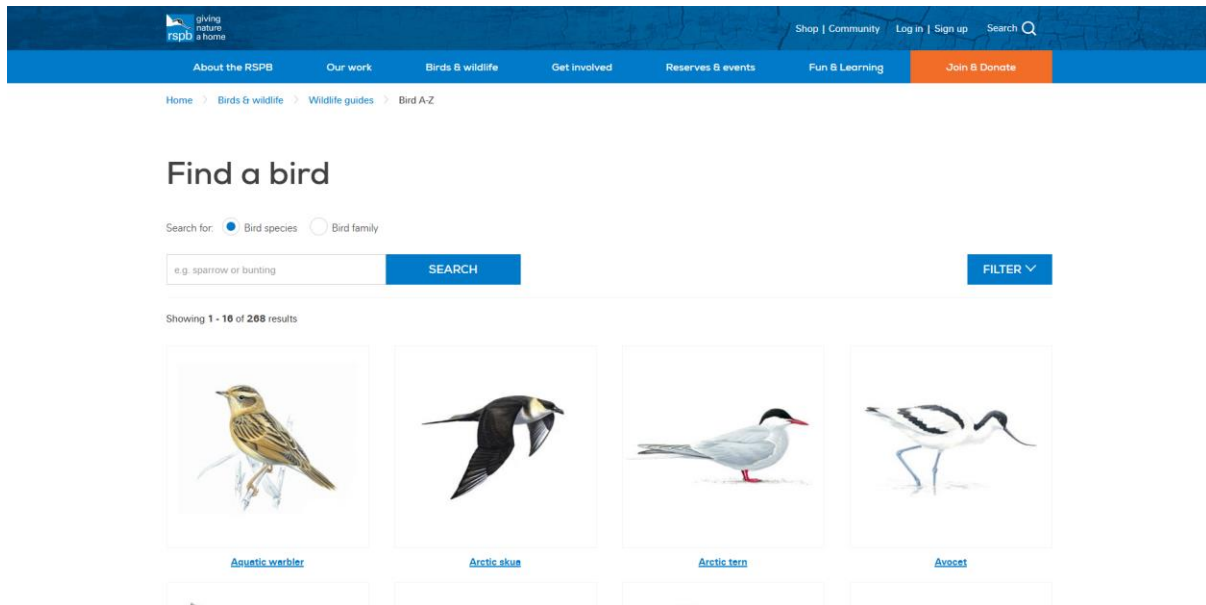


Figure 1 RSPB Website – Find a bird

Based on that, there is a site called RSPB (Royal Society for the Protection of Birds) [2], who have a database of birds.

The website itself is quite complex, contains lots of pages and information on the society and what they do. The website looks very clean and modern, with two dominating colours being white and blue. White being the background colour, and blue used for links, buttons etc.

The page is fairly intuitive and has lots of information. Everything is visible on the page, because the page follows a natural flow (i.e. the content is laid out in an easy to read manner).

Their bird database page is very similar to what I imagine my final page to look like layout-wise. The page contains a grid layout of “thumbnail” images of birds, alongside their name. Clicking on them, will take the user to their individual pages.

Unfortunately, some pages seem to be overloaded with information, making finding what a specific piece of information rather difficult. Not everything appears to be accessible through the menu, meaning the user must navigate through many pages to reach a specific resource. However, the website employs the use of “breadcrumbs”, which allows the user to see how they got to the page where they are now, and back track if need be.

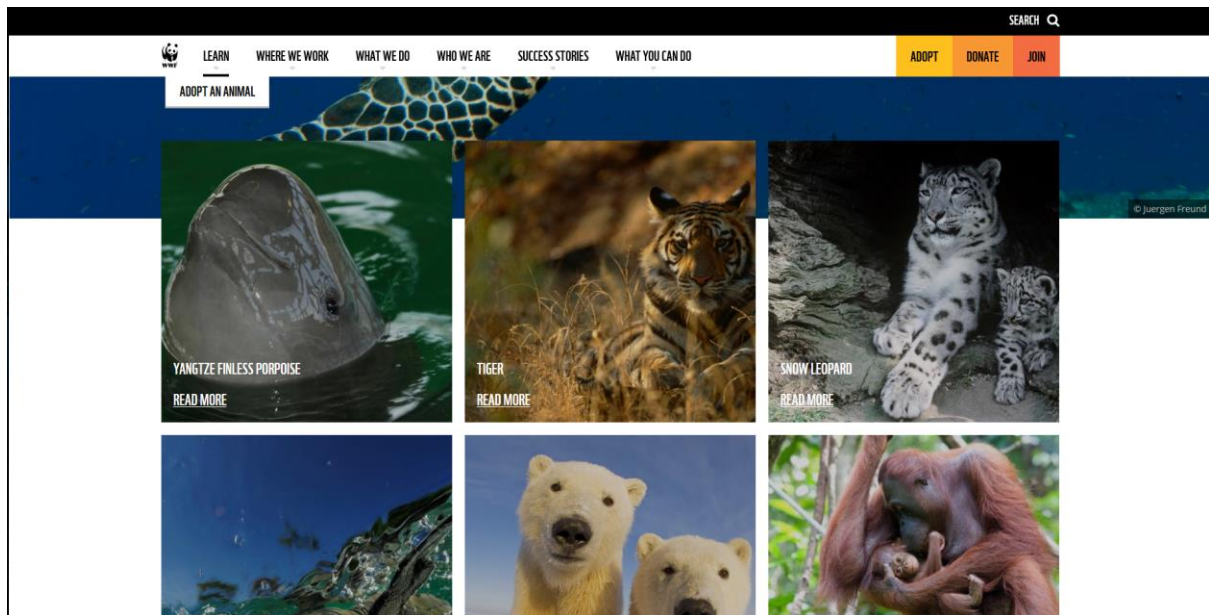


Figure 2 WWF Wildlife

The page above is that of the WWF UK (Worldwide Fund for Nature) [3]. The website has a list of animals a person can “adopt”.

Like the RSPB, this website is also quite complex. Its colour scheme is generally black and white, but their pages contain colourful pictures of animals and nature. The pages also contain a lot of information, which might be somewhat difficult to navigate. I believe their rather minimalistic and neutral colour scheme allows the user to focus on the content more.

The wildlife page in *figure 2* follows a grid layout, which makes it easy to understand and read. It is presented in a form of an “interactive gallery”, which contains thumbnails. This means that the user can click the images to go to the animal’s individual page and see more information about them. The page contains information about the animal, threats they are facing, and a map of their habitat.

Unlike the RSPB, the WWF do not use breadcrumbs to help the user navigate their website.

4. Approach to project

4.1. Technology

4.1.1. Proposed

I had many ideas when it came to think of ways to achieve the goal. My initial idea was to use a server-side framework, such as CakePHP or CodeIgniter. Despite having previous experience in using CakePHP, I have quickly come to realise that I did not know enough. I also found that it took me a long time to figure out the basics, where I could have done the same and more in plain PHP in the same amount of time. I am also considering using a content management system (CMS) if there are enough plugins to suit such a specific purpose, but this requires slightly more research.

In the end, I decided to go with just making the website from scratch. So, my decision was to use PHP, HTML, CSS, JavaScript, and a few libraries. I decided to do this because I have experience in making websites from scratch. My second-year project was written from scratch, as well as my own personal website.

The front-end will be structured using HTML and styled using CSS (Cascading Style Sheets). Forms will be handled using jQuery [4] (JavaScript), which will then pass data to the back end. That data will then be handled using PHP on the server-side, which will communicate with the MySQL (MariaDB) database, using PHP's PDO implementation and SQL for the database queries.

4.1.2. Complexity of the website

The website will be reasonably complex. There will be a contact form which will allow the users to contact the website administrators, or whoever else's email will be specified in the system configuration. The website will feature a login form, as well as password reset system, which will allow users to change their password, if the user has forgotten it. This will rely on the password reset link being emailed to the user by the system.

I am also intending on making a page for administrators to change various settings of the page. But it will primarily be used for creating, editing, and deleting user accounts.

I think the most difficult part will be creating the statistics and graphs dynamically. This is likely to involve a lot of mathematics, even if using third-party libraries. My mathematics skills are not great, and the libraries are most likely going to be very complex, which will present a big challenge. If I am unable to program this, I might end up doing a simpler implementation, and simply providing the raw data for the user to interpret.

Some features requested by the client are also difficult to implement. For example, the client mentioned that a freehand-drawn circle map search would be a good feature, however, I was unable to find any decent code examples of this being used, therefore it might not be a feasible feature. At least not in the timeframe I have, especially with the number of other features required.

4.2. Project plan

During the initial planning and review stages, a lot of business research will be done. This will include research into similar websites, and technologies used, as well as the technologies I can use during my production. It will also include drawing up a schedule and a timeline of the project.

During the design stage, both front-end and back-end will be designed. The designs will take into consideration different requirements and wishes from the client, as well as my own thoughts about making a good-looking and usable website. Those designs will then be shown to the client, and if required, changes will be made, based on the feedback provided.

During the implementation stage, I will constantly test the system, and ask other people to test features as I implement them, to make sure I do not create a large backlog of bugs. After the main implementation is finished, I will thoroughly test the system, trying to purposefully break it, to make sure it does not break during live use, and that people do not encounter some obscure bugs, when using it. Of course, it is very hard to avoid bugs completely, and there will be bugs left in the system.

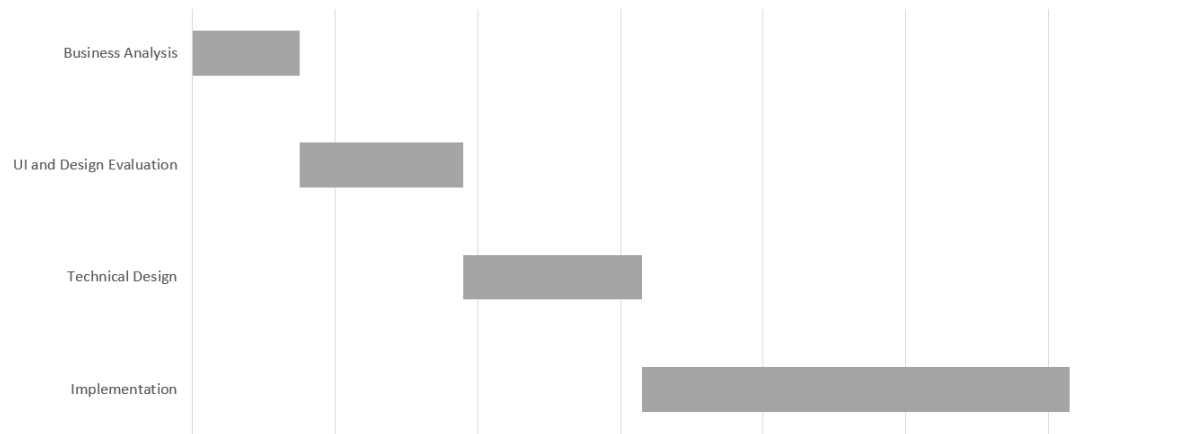


Figure 3 Gantt Timeline

5. Analysis

5.1. Business Analysis

5.1.1. Introduction

The website is being requested by Bernie Tiddeman and the research team he is involved in. The research team is comprised of researchers from Aberystwyth University, Bangor University, and is sponsored by the Welsh Government, who have an input in the project.

The website will be used by the fishermen, and the research team to upload and manipulate data. The research team are also considering whether to open part of the website to the public.

Data presented will be involve various information related to crustaceans such as crabs, their catch locations, and others. The website will then allow the data to be collated and presented as graphs and potentially other type of information.

The business analysis is required in order to understand the market better and understand what the users expect from such a system. I will also be producing diagrams to show the expected use cases and stakeholders.

5.1.2. Stakeholders

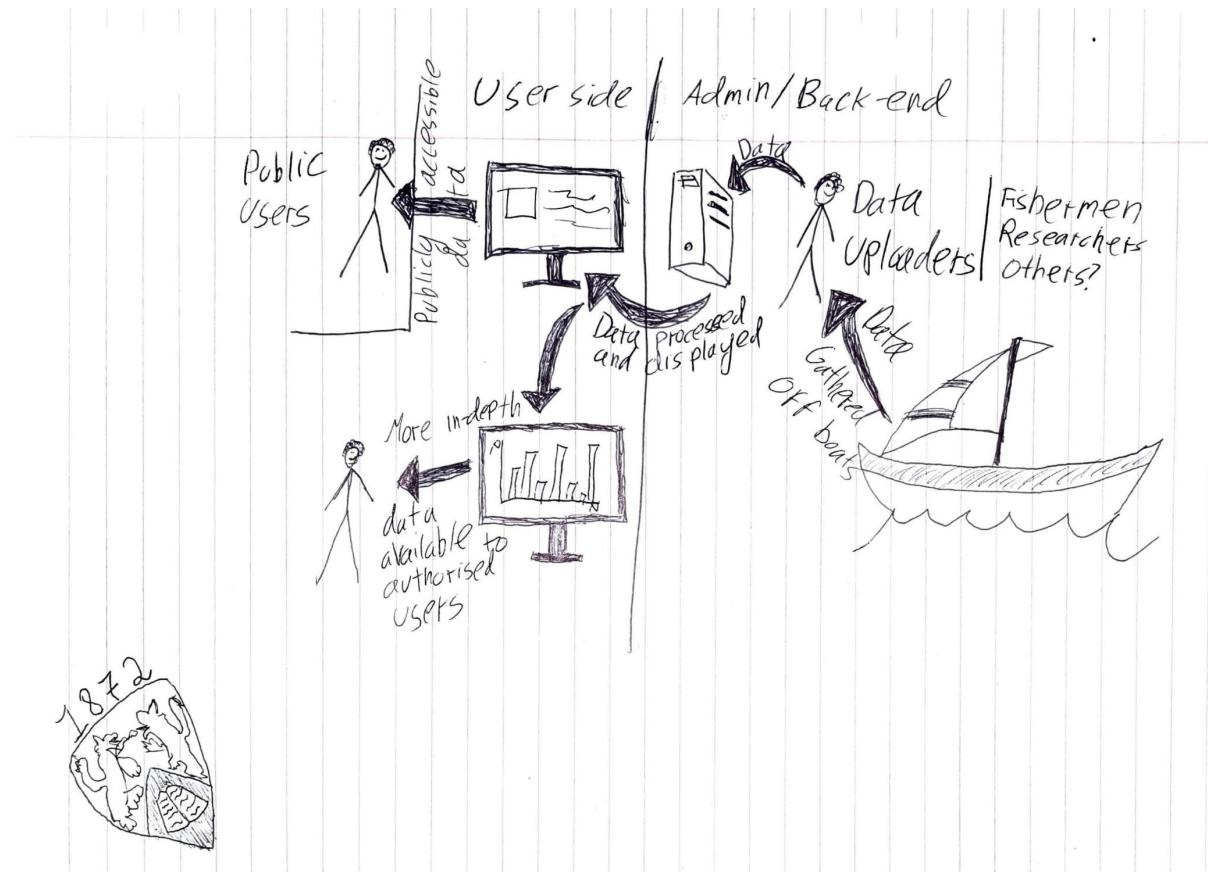


Figure 4 Rich picture

The rich picture above details the key “stakeholders” – people who will be accessing the system.

The most important people will be researchers (e.g. client) and fishermen, who will be the primary users of the system. The secondary users will be the people who need access for research purposes, who are not directly involved in the project.

The arrows in the diagram show the data flow in the system. The line in the middle denotes the separation between the users and the administrators and the people producing the data. It shows the data being generated by the boats, which is then uploaded to the database by the researchers manually or automatically. That data is then processed by the system I will produce, and displayed to the end user, as either raw or formatted data.

5.1.3. Users and use cases

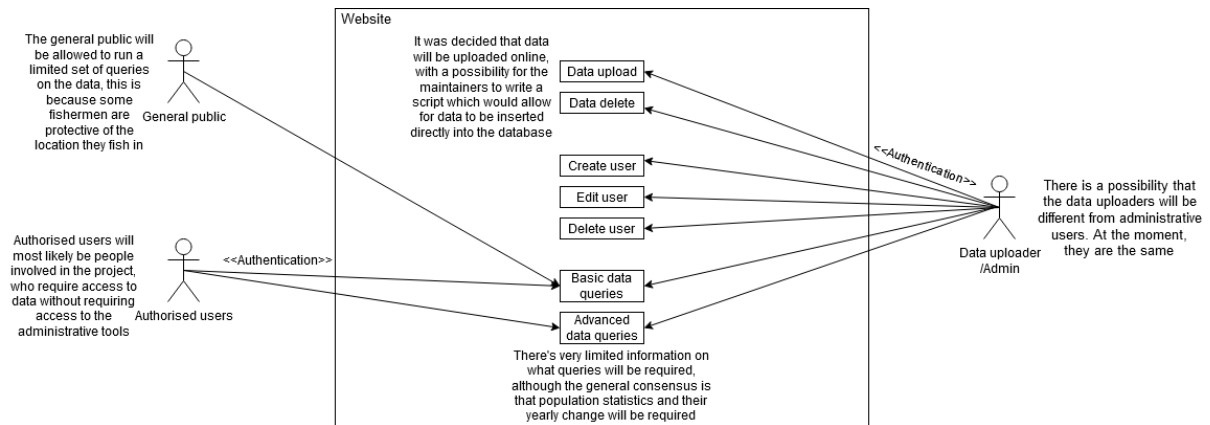


Figure 5 Use case diagram

Above is the initial proposal use case diagram, which represents the main users and website functions. As noted above, there is a possibility that administrative users and users with upload permissions will be separate. However, as of now there is no distinction between the two, therefore they are represented by a single actor. Administrators will have access to all data.

Authorised users would include financial sponsors, such as the Welsh Government, and possibly the EU Fisheries Fund, although the latter is debatable due to the current political situation in the country. Those agencies might use the data gathered for statistics, and to prevent overfishing.

Other authorised users would be various researchers and fishermen themselves, as it would help them monitor various population statistics.

Authorised users will be able to access most data, however, there was a mention of a permissions system, where certain users would be able to access only certain pieces of data – this needs further discussion and assessment.

Data upload: This will be done by administrators. Data will be uploaded either through a form on the page, or directly into the database through automated means. The database will likely be very large, containing many entries.

Data delete: This will allow the administrators to delete records, if they contain invalid data, or if the record is no longer needed in the system. There is no way to edit an entry, as it would make more sense to upload the entry anew, instead of editing it, as that can create mistakes and inconsistencies.

Create user: The system will allow for new users to be created and maintained; this is so proper access control can be maintained. The client wants to be able to control who has access to the data, therefore implementing a user system is ideal.

Edit user: Editing users is an important feature. This will allow the administrators to change the users' details like their name, surname, email address and password. It will also be possible to disable the user, which will prevent them from logging in without having to delete their account.

Delete user: Deleting a user is useful, especially if the user no longer needs access to the system, or if their account is no longer being used.

Basic and advanced queries: If a decision is made to make some data public, the public will have access to some basic data, whereas users with accounts will have access to more advanced data, and their queries. This is another layer of access control.

Overall, the stakeholders' needs in the use case diagram are reflected very well.

5.1.4. Typical users

Typical users of the website will generally be from a single demographic. The users will be researchers, fishermen or those generally interested in the data. They will usually be associated with the government or some academic institution.

Some typical user biographies:

Dewi is a 32-year old researcher at the Bangor University. He is researching crab populations and uses the system to gather data on crustaceans and their catch location.

Anna is a 25-year old Welsh Government employee gathering data to help the Government prepare marine conservation plans.

Rhys is a 45-year old civilian researcher, who requested access to the system, because he is interested in recording the change in numbers of crustaceans over time.

The most common use will be to check different crustaceans' records, especially their information, such as their size, gender etc.

6. UI and Design Evaluation

6.1. User Interface (UI)

6.1.1. Introduction

A good UI (User Interface) design is important. When a person uses an interactive item, whether it would be a website, a mobile phone, or even keylock; it is important for the user to be able to use the interface with ease, without having to spend time figuring out how to do a specific action.

For my project, I have to take the users into account, as well as look at current trends, and make sure that the interface is easy to navigate and does not confuse the user. Confusion can lead to mistakes, which can lead to misunderstandings, and potential loss of data.

I will attempt to create a user-friendly, and easy to use design, which will make the user's visit to the website pleasant. I will look at how other websites deal with common design issues and will implement similar measures. I intend to ask a few people to perform a set of tasks on the website and record their feedback. This will give me a better insight into what works well, and what needs some changes. I have submitted a research ethics form, which was approved at departmental level. This requires me to inform all participants that their data will be stored confidentially, and that they can withdraw their consent at any point in time.

However, I am not a designer. I cannot see what colours compliment each other, and which ones do not. I am also red-green colour blind. Because of that, I will aim to create a simple but functional design which meets the previously mentioned requirements.

6.1.2. User and technological needs

It is important to identify what the users need from the system. It is also equally as important to identify how the system must function.

I should consider the platforms on which the website is likely to be used, and make sure the website development is targeted to those platforms. This includes consideration for screen sizes, as well as inputs. Ideally, the design should be responsive and work well on the targeted platforms. The target platforms are laptops and desktop computers. Because the data is supposed to be accessed by researchers, fishermen and other similar users, it is unlikely that they will need mobile access. This means that mobile will not be a target platform.

We also need to take data into consideration. If the backend database contains a lot of data, the website needs to deal with that accordingly, and the amount of data should not cause unnecessary slowdowns. The website will use a combination of PHP and JavaScript (and libraries) to deal with data. The system itself is not expected to be very large, but it is supposed to handle relatively large datasets.

I intend to ask a few individuals their opinion on the design, and take their comments into account, to make sure the website is easy to use, and that the users understand how to use the website, without having to spend a long time trying to figure out how it works. I will prepare a list of simple tests for them to perform, which will give me a better idea on how the website performs when used by average users.

6.1.3. Website structure

Firstly, I need to come up with a basic layout, especially for the header (top part of the page), and footer (bottom part of the page). These will stay more or less the same throughout the page, with some differences to reflect if the user is logged in, and whether or not they are an administrative user.

I will produce a sample layout which will act as a guide when producing the actual implementation of the website. The layout will be based on pages I have explored previously. I aim to make the layout as simple and straightforward as possible; I also expect it to follow a “natural flow” and make it easily understandable.

There will be 3 access tiers; Tier 1 will be available to people who are not signed in and will only have access to the most basic search and data querying tools. Tier 2 will be available to people who have accounts and are signed into the system, they will have access to all data and querying tools. Tier 3 is only available to a few users who are designated as system administrators.

Below are navigation tree diagrams showing the 3 access tiers discussed previously.

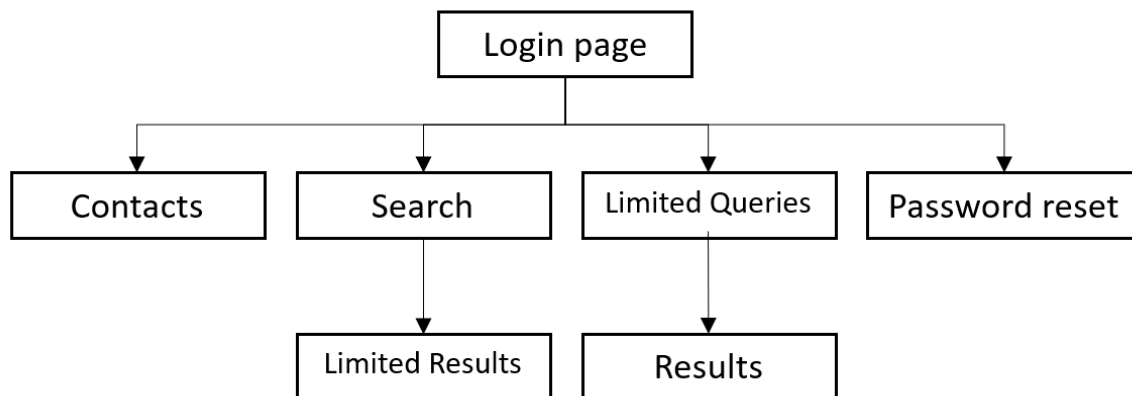


Figure 6 The navigation options available to general public (i.e. users who are not signed in).

Fig. 6 Shows a navigation diagram, which displays the options available to users who are not logged in to the system. Although the diagram includes access to queries and data to the public, it is unclear whether the data will be accessible publicly. Even if I will not implement public access to the data, the structure of my code, will most certainly allow for easy implementation of the feature by the future maintainers of the code.

The diagram shows the login page as the root element, which will be the homepage for users who are not signed in. I decided to make it the default page, because most users accessing the system will have accounts, therefore it makes sense to have it as the main page.

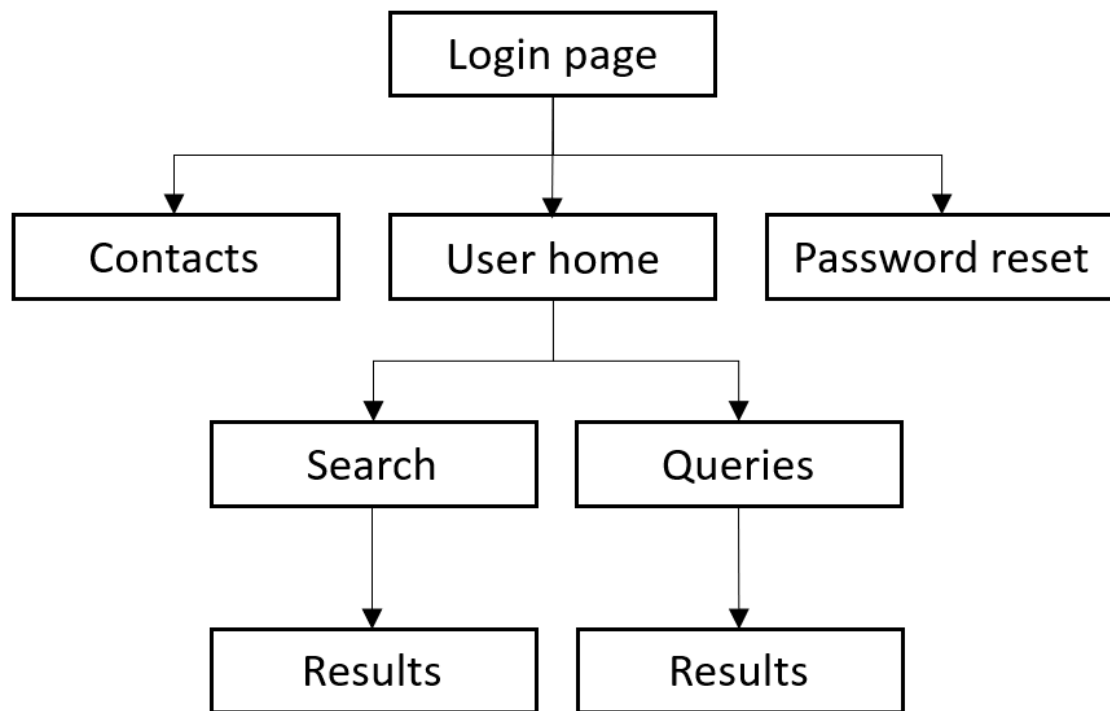


Figure 7 The navigation options available to authorised non-administrative users.

Fig. 7 Shows the options available to logged in non-administrative users. The only difference between logged in users, and the general public is that the logged in users have access to more data.

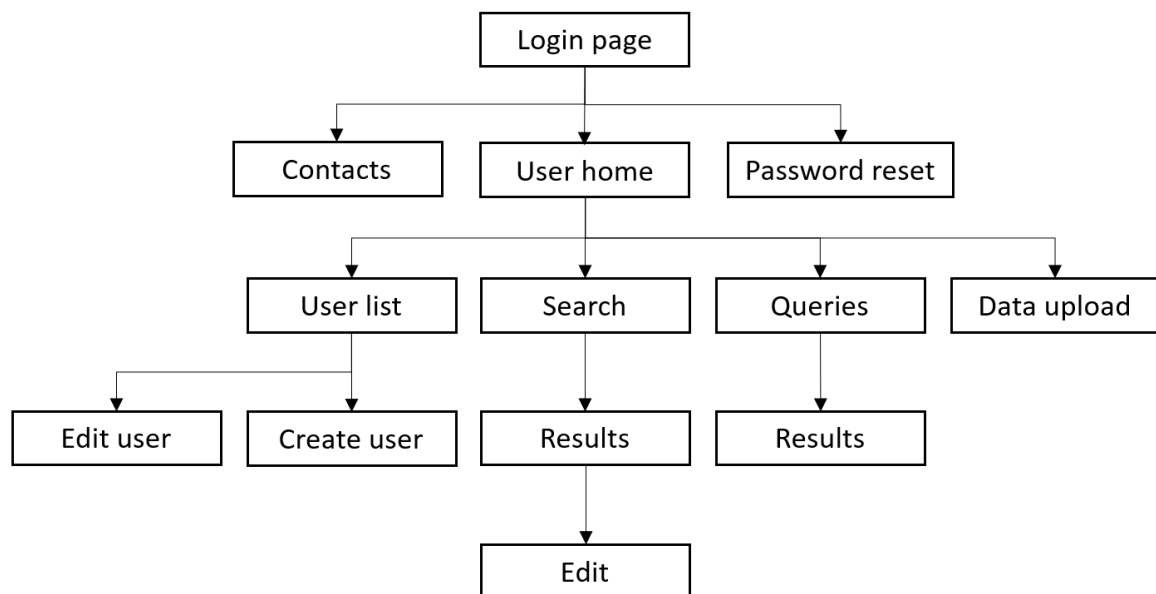


Figure 8 The navigation options available to administrative users.

Fig. 8 Shows the ability to upload data, however this is not the only way to upload data. Another way to upload data to the system will be by creating a record in the database manually or through automated means, and by uploading the relevant picture to the file store.

The contacts page will be used to contact the people running the database, for example to request an account, or to report an issue.

Crustacean Database			
Home	Search	Queries	Contacts

Welcome to the crustacean database! This database is run by Aberystwyth University researchers. The database contains records of various crustaceans from around Wales. At the moment, we have 1,108 records!
If you have an account, please login below.

Username:

mrbean

Password:

.....

Login

Forgotten password

Copyright © Name 2020

Figure 9 Login form

This is the main (home) page of the website, as majority of the users will have accounts, the default page will display the login form. It will display a short message at the top, alongside some statistics of how many records there are – it will be configurable.

Crustacean Database			
Home	Search	Queries	Contacts

If you forgot your password, please use the form below to reset it

E-mail:

ratkinson@example.com

Last name:

atkinson

Submit

Copyright © Name 2020

Figure 10 Password reset request form

If the user clicks “Forgotten password”, they will be taken to this page, which will ask for their email and last name. This will be done so people cannot try random email addresses and spam account

owners with password reset notifications. Once their details are entered, they will receive an email with a link which will allow them to enter a new password. Another option would be to include some form of anti-spam protection, such as a captcha. The benefit of doing that would be that the users with account would not have to enter their name, especially if in a public location. It would also reduce the amount of spam passed through the system.

Crustacean Database

Logout	Search	Queries	Contacts
--------	--------	---------	----------

Welcome, Rowan Atkinson
User type: User
Session started: 30/06/2020 11:22am

Relevant information:
There is nothing to display at the moment!

Copyright © Name 2020

Figure 11 User homepage

This is the page a non-administrative user will see when logged in. It will display some information about the user, and any announcements (“relevant information”) that may have been posted by the administrators. Because the system is not very big, this page might be quite empty, which might look a bit strange to some users.

Crustacean Database

Logout	Search	Queries	Contacts
--------	--------	---------	----------

Welcome, Rowan Atkinson
User type: Administrator
Session started: 30/06/2020 11:22am

Users

Upload

Relevant information:
There is nothing to display at the moment!

Copyright © Name 2020

Figure 12 Administrator homepage

Administrators will have a slightly different interface, they will have additional two buttons called “users” and “upload”, which will allow them to access the list of users and to upload data. The button location might be moved around slightly during development.

Crustacean Database

Logout	Search	Queries	Contacts
--------	--------	---------	----------

Search:

Welcome, Rowan Atkinson!

Rowan Atkinson – Administrator – ratkinson@example.com	Edit
Adam Smith – User – asmith@example.com	Delete Edit

Copyright © Name 2020

Figure 13 User list

The user list will contain every user registered on the system, alongside a simple search bar. Administrators will be able to edit every user and delete everyone except themselves. They will also have an option to create new users. There will also likely be an option to disable users, so the user account would still exist, but it would be unusable. The administrators would not be able to disable their own account or remove their administrative privileges.

Crustacean Database

Logout	Search	Queries	Contacts
--------	--------	---------	----------

Welcome, Rowan Atkinson!

Name:

Type: User ☒ Administrator ☐

Email:

New password:

Repeat new password:

Note: Leave the password fields blank if not changing the user's password!

Copyright © Name 2020

Figure 14 Administrator user edit page

Above is the user editing screen, which allows the administrator to change all details. With the exception of the ones noted in the previous section.

Crustacean Database

Logout	Search	Queries	Contacts
--------	--------	---------	----------

Welcome, Rowan Atkinson!

Name:

Type: User ☒ Administrator ☐

Email:

New password:

Repeat new password:

Copyright © Name 2020

Figure 15 New user page

User creation page is almost identical to the edit page, with a few differences being the confirmation button is instead called “Create”, and entering a password is compulsory.

Crustacean Database

Logout	Search	Queries	Contacts
--------	--------	---------	----------

Welcome, Rowan Atkinson!

Image name:

Species:

Sex: Male ☒ Female ☐

Carapace width:

Abdomen length:

Abdomen width:

Boat name:

Latitude:

Longitude:

Image:

Copyright © Name 2020

Figure 16 New record page

Clicking “upload” from the administrator’s user page will bring them to the page above, which will allow the administrator to create a new record. Not applicable fields can be left blank. This is a speculative page, it might or might not exist, depending on the database structure.

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

You can use the search form below to find records of crustaceans stored within our database!

Image name:	<input type="text" value="Image1"/>	Abdomen length:	<input type="text" value="109"/>
Species:	<input type="text" value="Lobster"/>	Abdomen width:	<input type="text" value="55"/>
Sex:	Male <input checked="" type="radio"/> Female <input type="radio"/>	Boat name:	<input type="text" value="Swift"/>
Carapace width:	<input type="text"/>		
<input type="button" value="Submit"/>		<input type="button" value="Clear"/>	

Copyright © Name 2020

Figure 17 Public record search page

Once the data has been uploaded, everyone will be able to search for it. However, not all data will be available publicly. Although there will not be an option to set individual visibility of different pieces of data, the code structure aims to make it relatively simple to implement later on.

Crustacean Database

Logout	Search	Queries	Contacts
--------	--------	---------	----------

You can use the search form below to find records of crustaceans stored within our database!

Welcome, Rowan Atkinson!

Image name:	<input type="text" value="Image1"/>	Abdomen length:	<input type="text" value="109"/>
Species:	<input type="text" value="Lobster"/>	Abdomen width:	<input type="text" value="55"/>
Sex:	Male <input checked="" type="radio"/> Female <input type="radio"/>	Boat name:	<input type="text" value="Swift"/>
Carapace width:	<input type="text"/>	Latitude:	<input type="text" value="52.38"/>
		Longitude:	<input type="text" value="4.08"/>
<input type="button" value="Submit"/>		<input type="button" value="Clear"/>	

Copyright © Name 2020

Figure 18 Logged in user record search page

A logged in user will have slightly more search options, latitude and longitude included above are used for illustration purposes only and might change during development.

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

1 record found!

Image1	Lobster	Male	View
--------	---------	------	------

Copyright © Name 2020

Figure 19 Search result page

Searching for a record without being logged in will return a page showing a list of all matches. The users will then be able to view their chosen record.

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

Image name: Image1
Species: Lobster
Sex: Male
Abdomen length: 109
Abdomen width: 55
Boat name: Swift
Latitude: 52'38 N
Longitude: 4'08 W

Back



Copyright © Name 2020

Figure 20 Individual record page

This is what a record screen will look like to the user. It will contain metadata about the catch, and the image (if applicable).

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

Welcome, Rowan Atkinson!

Image name: Image1
Species: Lobster
Sex: Male
Abdomen length: 109
Abdomen width: 55
Boat name: Swift
Latitude: 52'38 N
Longitude: 4'08 W

Back

Delete

Edit



Copyright © Name 2020

Figure 21 Administrator record page

Administrative users will also have the option to edit or delete the record. Deleting will return the user back to the search screen.

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

Welcome, Rowan Atkinson!

Image name:
Species:
Sex: Male ☒ Female ☐
Abdomen length:
Abdomen width:
Boat name:
Latitude:
Longitude:

Cancel

Save



Copyright © Name 2020

Figure 22 Record edit page

The edit screen will allow the administrators to change all the details about the record, except for the image. If they wanted to change the image, they should delete the record, and upload a new

one. This is done to prevent mistakes, and inconsistencies in the database.

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

You can use the below queries to extract specific data about our catches

Query:

Population density ▼

Location selection:

Coordinates ▼

Number of

Lobster ▼

 within

5

 miles of: Latitude

52.38

Longitude

4.08

Submit

Clear

Copyright © Name 2020

Figure 23 Statistics query page

Users will also be able to run queries on the data. Logged in users will have more filters and tools than users who are not logged in.

Demonstrated above is the “population density” filter, which allows users to find the number of matches within a square area of a specific point, as well as the population density within that area.

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

There are 2 catches of Lobster within 5 square miles of 52.38N, 4.08W!
Population density is 0.4/sq. mi.

Image1	Lobster	Male	View
Image49	Lobster	Female	View

Copyright © Name 2020

Figure 24 Query result page

This is what running the query would return.

Crustacean Database

Home	Search	Queries	Contacts
------	--------	---------	----------

Please use the contact form to contact us!

First name:

rowan

Last name:

atkinson

E-mail:

ratkinson@example.com

Reason:

Select a reason... ▼

Tell us about your issue:

Submit

Clear

Copyright © Name 2020

Figure 25 Website contact page

Finally, there is a possibility that some problems might occur, therefore the contact form allows users to contact the maintainers of the website directly. The form could be used to report problems, or to request access to the system, as it will be available publicly. This page will most likely have some sort of anti-spam protection, in order to reduce the number of robots abusing the form.

6.1.4. Evaluation

Before conducting the tests, I told users that their answers, and all other data they provide will be held anonymously and will be stored in a password-protected archive.

The testing stage was brief and involved asking 3 people some questions of how they would perform certain actions on the prototype. I provided them the prototype in PDF format, which has internal links, to simulate the behaviour on a real website.

2 of them noted that it was difficult to figure out how to use queries, when they had a simple form with fields to fill (similar to what the edit page looks like), this led to me redesign the query form to make it more readable.

Following the search test, everyone noted that the initial search results did not contain enough information to uniquely identify a result, therefore I included the image name, species, and gender. I considered including a thumbnail for every result, however, I decided against it. Large searches would have taken a long time to complete, and the page would take a long time to load if it had to load a lot of thumbnails.

No other significant changes were made, other than some minor stylistic layout changes to make the design look more visually appealing. However, given the minimalistic look of the prototype, the design might change during development.

I think that the initial design was well done, especially because the UI is not particularly complicated, and does not include a lot of pages or options, thus getting it almost right the first time is not very surprising. All pages follow almost the same layout, therefore I did not expect the users to find navigation difficult, as they would get accustomed to seeing the layout from the first few moments.

However, given the small number of testers available, there might be some other things we missed during our testing.

6.2. Database design

The database will contain 5 different tables.

6.2.1. Tables

system:

This table will only contain 3 columns: *id (PK)*, *motd* and *user_announcement*. It will be used for storing system configuration and information.

Id – This will be the entry’s identification number, it is a primary key, therefore it must be unique in the table.

Motd – This stands for “*message of the day*”, but it simply refers to a message which will be displayed to the users.

User_announcement – This will be displayed to logged in users. It is a useful feature, as it can be used to tell the user about upcoming maintenance or downtime.

users:

This table will contain 7 columns: *email (PK)*, *first_name*, *last_name*, *admin*, *disabled*, *password* and *api_key (U)*. It will be used for storing user account details.

Email – This will be the user’s login, which is also a primary key, meaning it must be unique across the system. Because of this, a single email account can only have a single email address.

First_name – This is the first name recorded on the account, it will be used to greet users.

Last_name – This is the last name recorded on the account, it will be used to greet users.

Admin – This is a Boolean value. It will record whether or not the account has administrative privileges. The database in use, records Boolean values as 1 and 0.

Disabled – This is a Boolean value. It will record whether or not the account is disabled. If the account is disabled, the user will not be able to log in, or use the API, but they will still be able to reset their password.

Password – This will store the user’s password in an encrypted format. The encryption algorithm is specified to be “BCRYPT_BLOWFISH”, which is the PHP’s standard encryption algorithm for the PHP version the website will be developed on. [5]

Api_key – This will contain a randomly generated string of 30 characters. This will allow users to access the pseudo-API I will develop. This field must be unique, as having duplicate keys in the system would cause errors and would be a security problem.

pass_resets:

This table will contain 5 columns: *email*, *date*, *time*, *reset_key* (PK) and *valid*. It will be used to store user password reset request information.

Email – This is the requesting user’s email address. There can be many of the same ones in the table, as the user can request password resets many times over. However, it is an index.

Date – This is the date of when the password reset was requested. This will be used to remove old password resets after a certain amount of time.

Time – This is the time of when the password reset was requested. This, alongside the date, will be used to remove old password resets.

Old active password resets will also use both *time* and *date* fields, to decide when it is time to invalidate them. This is because having forever-active password reset keys is a security problem.

Reset_key – This will be a 30 character long, cryptographically secure string. It will be emailed to the user as part of a link, to enable them to reset their password. This is also a primary key, meaning it must be unique.

Valid – This is a Boolean value. It stores whether or not the password reset request is active. When a new request is created, it is automatically set to active. When the user resets their password, it is invalidated, and can no longer be used. If a user requests a password reset, while there is an active request, the previous key gets invalidated, and the new one becomes active instead.

records:

This table will be by far the largest and will contain 9 columns: *img_name* (PK), *species*, *sex*, *carapace_width*, *abdomen_length*, *abdomen_width*, *coords_lat*, *coords_long* and *boat*. It will be used to keep all the database entries on crustaceans.

Img_name – This is the record name, it is also a primary key, as it has to uniquely identify the specific record.

Species – This will contain the animal’s species.

Sex – This is a Boolean value. It will store male as *true/1* and female as *false/0*.

Carapace_width – This contains the animal’s carapace width. Not all records will contain a carapace width value, as it is not applicable to all animals.

Abdomen_length – This contains the animal’s abdominal length. Not all records will contain an abdominal length value, as it is not applicable to all animals.

Abdomen_width – This contains the animal’s abdominal width. Not all records will contain an abdominal width value, as it is not applicable to all animals.

Coords_lat – This represents the latitude of where the animal was captured. It is represented as a floating-point number (i.e. decimal).

Coords_long – This represents the longitude of where the animal was captured. It is represented as a floating-point number (i.e. decimal).

Boat – This will contain the name of the boat which capture the specific animal.

images:

This table will only contain 3 columns: *img_name* (PK), *image* and *img_type*. It will be used to store image information.

Img_name – This is a primary key and contains the image’s name. It is directly related to “img_name” in the *records* table. For example, if a record called *image1* does not exist in “records”, it also cannot exist in “images”.

Image – This will store the actual image. These columns will be much larger than any other column in the database, because they will contain BLOBs (Binary Large Objects). The images will then be parsed from the database, and decoded as base64-string objects of immutable, raw-data [6]. They will be stored as type *MEDIUMBLOB*. This type can contain up to 16,777,215 bytes of data [7], or 16.77 megabytes, which is plenty enough for a single image. The largest image in the sample dataset I was provided is 106 kilobytes, meaning the images are compressed.

Image information will be stored separately from its parent entry for 2 reasons:

1. So that queries will not take a long time to complete by having to run over large BLOBs during operations. Especially if more than one query is run on the same page.
2. So that the database is easier to maintain, and that images can be created and deleted from the database manually, as per client’s request.

There are some constraints implemented, which should prevent data from being changed in case the system encounters an error.

Columns *img_name* in tables “images” and “records” are related. This means that *img_name* in “images” must match a record in “records”, and any changes to the image name will be reflected in both tables. Deleting the record will also delete the image from the “images” table. An equivalent constraint is also implemented between “users” and “pass_resets”.

6.2.2. Sample data

Some sample data was provided by the researchers for use in productions and testing. Because the client does not have any strict requirements when it comes to data formats, it was converted data to an easier to implement format.

```
{
  "image1": {
    "species": "lobster",
    "sex": "1",
    "abdomen_length": "109",
    "abdomen_width": "55",
    "coords_lat": "52.38",
    "coords_long": "4.8",
    "boat": "swift"
  },
  "image5": {
    "species": "crab",
    "sex": "0",
    "carapace_width": "68",
    "coords_lat": "52.48",
    "coords_long": "4.11",
    "boat": "swift"
  }
}
```

Here are 2 records in JSON format, this is the format which will be used for performing data queries. In the example above, we have a male lobster, and a female crab. As mentioned in the previous section, sex is represented as a Boolean, - male being 1 (i.e. true), and female being 0 (i.e. false). This is also how it will be represented in the database.

Below are the images associated with the above data, which will be stored in the “images” table as BLOBs (Binary Large Objects).



Figure 26 Male lobster



Figure 27 Female crab

The tables and records will be created using phpMyAdmin [8], which is a web-based open-source application used for managing databases. As I have quite a lot of experience using it, I believe it will be a great asset, allowing me to manage the database quicker and more efficiently. Because we will be storing images in the database, phpMyAdmin also offers a profiling tool, which will allow me to see how long it takes for queries to execute, as well as different components of those queries.

email	first_name	last_name	admin	disabled	password	api_key
jsmith@example.com	John	Smith	0	1	\$2y\$10\$xRiY TjBlgLl2atXg DdK...	NULL
ratkinson@example.com	Rowan	Atkinson	1	0	\$2y\$10\$4Pu Zy9kO7vrQc3 iK2...	fed13fdbb20c 2ca62...

The table above shows two user accounts and their details. A user called Rowan Atkinson is an administrator with an email ratkinson@example.com. Whereas, John Smith is a regular user with an email jsmith@example.com.

John Smith's account is disabled, meaning he still has an account on the system, but cannot access it. The passwords will be stored in an encrypted form, using a secure and modern hashing algorithm with salting to reduce the likelihood of the user credentials being hacked.

The last fields stores the user's API key, which is a 30-character long unique string of pseudo-random characters, generated using a cryptographically-secure function [9].

email	date	time	reset_key	valid
ratkinson@example.com	2020-03-08	19:24:06	fjgutdlcjguthrckqz7g4d1cds597	1

Above is a sample password reset request, submitted for ratkinson@example.com on the 8 March 2020, at 19:24. The key was randomly generated by the system, and still has not been used, therefore the field *valid* is true. Once the password is reset, it will become invalid, and *valid* value will be set to false.

id	motd	user_announcement
1	Hello and welcome to our system!	Thank you for using the user area!

The table above shows a sample entry in the “system” table. At the moment, the table only records “message of the day” type messages, and user area announcements. The system will only display the latest entry, this might later be improved to allow for multiple messages to be displayed.

img_name	image	img_type
image1	0xffd8fffe000e4c61766335342e34...	image/jpeg

Above table shows a sample image record. Value *image1* would have to match a record in the “records” table, field “image” keeps a BLOB which is usually upwards of 100 KiB in size. The last field records the MIME (Multipurpose Internet Mail Extensions) type of the image [10], which will then be read from the database and sent by the server as a header in order for the browser to correctly recognise and display the image. This will be used when displaying images as part of the pseudo-API.

6.3. Addressing security concerns

Security issues are a big problem in today’s society, especially when it comes to the Internet. Large companies hire people with the sole purpose of auditing their security measures and making sure they are safe.

The website safety is very important for me, therefore I will be implementing various security measures, to try and make the website as secure as I possibly can.

There can be many security problems, especially when designing and implementing a website from scratch. Problems like SQL injections [11], XSS (Cross-Site Scripting) [12], and CSRF (Cross-Site Request Forgery) [13] are very common. Fortunately, they can be avoided with careful planning, and making sure they code I write is not susceptible to those attacks.

In order to mitigate those problems, I will be using prepared statements to protect against SQL injections, this will greatly reduce the risk of data and integrity loss. I will also be implementing a token-based system, which will reduce the likelihood of CSRF attacks.

The biggest challenge will be making a secure user system. Securing the login form, and the user system overall will be quite difficult. But it is important to protect the sensitive parts of the system.

7. Implementation

7.1. Planning

7.1.1. Proposed technology overview

In this section, I will cover the technical and design aspects of the implementation. I will be hosting the website on a VM (Virtual Machine) provided to me by the Department. The website will only be accessible internally, within the University network. The URL is: <https://mmp-dec21.dcs.aber.ac.uk/project/> [14].

I have previous experience in building websites. My second-year group project was also created from scratch, and I was the lead developer. I was responsible for writing the backend code. I have used different languages and technologies in the past. Namely: HTML, PHP, JavaScript (including jQuery [4]), CSS and SQL for database queries. My main IDE is NetBeans [15], I have been using it for at least 5 years.

One of the main obstacles of the system is data. Large datasets can mean that the page will load slowly. Another factor is inefficient code. If the software is programmed inefficiently, it will run inefficiently, thus increasing load time.

For example, let us assume that the software is very well optimised. We have a dataset of 278.8 kibibytes. I ran 5 queries and told the database to give me the cached results every time. It took an average of 123 μ s for me to receive the data. This is less than 0.2ms!

In a perfect system, the data transfer speeds would increase at the same rate as dataset size. So, if we multiply our dataset size by 5, giving us 1.36 mebibytes, we will see our transfer speeds multiply by 5 as well.

Over 5 attempts, we would see an average transfer rate of 615 μ s. This is still less than 0.7ms.

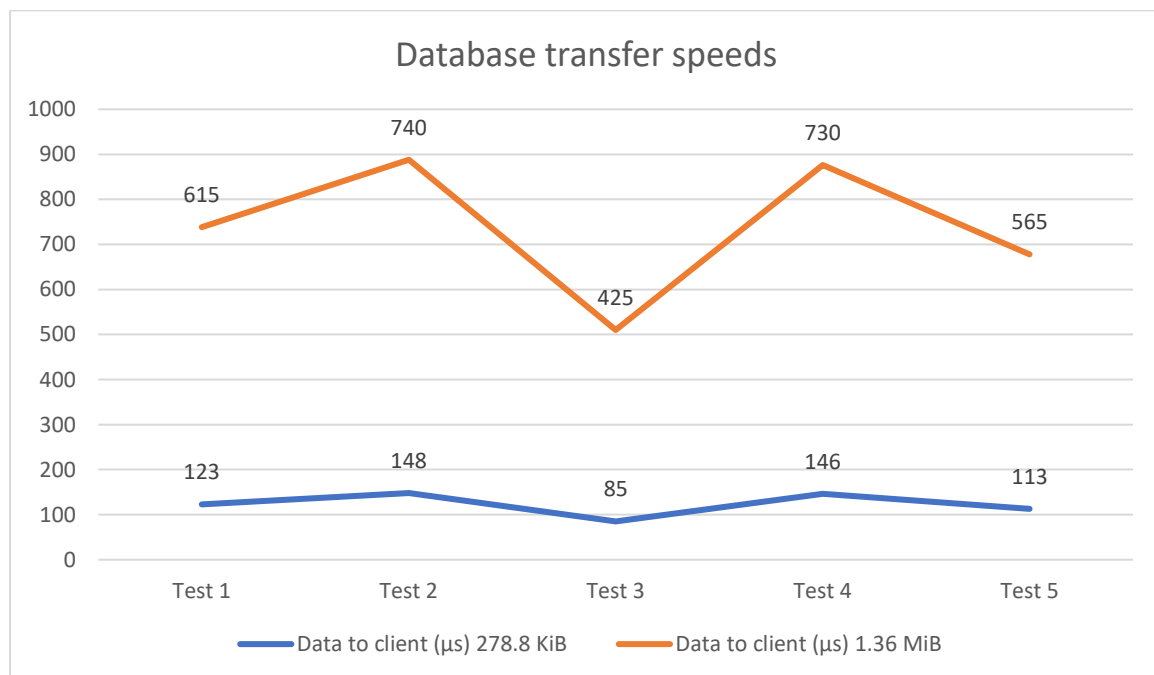


Figure 28 Server to client transfer speeds

This data is not perfect. It is simply a visual way of saying that the transfer speeds will not be a major problem if the implementation code is handled well enough.

I will use PHP and JavaScript for main data handling. PHP is a server-side language; therefore, it will do all the “heavy lifting” when it comes to processing data and communicating with the database server. It will also create dynamic pages, which is a very important aspect when it comes to creating websites which have dynamic (non-static) data. I will also use it to transfer data between the client and the server using POST method. This is more secure than using GET, as it does not save any variables in the URL. Some low-security parts of the website will use GET, like linking to a record. This will allow users to share links directly to a specific crustacean record.

JavaScript is a client-side language and runs on the user’s browser. I will use it to make the website more interactive and improve the overall experience for the user. I will also use it display and format data on-screen.

I will also configure the server to use SSL, by creating a self-signed certificate. Naturally, this will generate a security warning, as the browser will not recognise the certificate, but it will work well enough to test SSL features on the website.

The website will also send password recovery and contact emails to users. In order for this to work, I will have to configure a mail server (e.g. postfix), and make sure that the PHP’s internal mail() function works, before using it in production.

7.1.2. Schedule and development flow

I have been developing the system for a while; however, I will now make a schedule, which will enable me to follow a development plan better.

A generic overview of my plan:

Features	Core	Secondary	Bug fixes
Time allocation	85%	10%	5%
Examples	Login, user management, animal listings	Queries, API	

Table 1 Schedule overview

I aim to spend 75% of my total development time on developing core features. I and the client discussed which features are the most necessary, and in what order – from most to least.

10% of my time will be spent on secondary features. These features are not core but will greatly improve the website.

5% of my time at the end, will be spent on bug fixes. This is not a lot of time but given that I will be testing the system as I go, I expect this will be enough time to test the system, and fix majority of the bugs encountered.

7.2. Changes

There have been quite a lot of changes during the main development. Some were made because I have come up with better ways of doing something. Others, because I have spent too long trying to get a feature working and were falling behind on implementing other features.

1. A major change was a switch between using different pages to show data (i.e. sending a POST request from one page to another, to represent state change) to using AJAX for nearly all forms. I believe this was a positive change. However, using AJAX comes with its own caveats, like having to protect the forms. For this, I implemented a token-based CSRF (Cross-Site Request Forgery) system, as mentioned earlier. I got this idea from the Netsparker's web security blog [16]. They mentioned many ways of implementing anti-CSRF protection, but their token-based solution seemed the most appropriate.
2. I have decided not to implement an edit form for individual records. This was because editing a record can create inconsistencies in data, which could lead to erroneous data products if someone decided to use that data. Therefore, the administrators will have to delete a record, and upload a new one, if they wanted to make some changes. This would likely reduce data inconsistencies at the cost of time.
3. I initially wanted to use Bootstrap [17] for a lot my page design. However, once I started implementing parts of my own design, I noticed that parts of Bootstrap were clashing with my CSS. I did not want to redesign my website; therefore, I abandoned the idea of using Bootstrap.
4. I was going to use Google Maps for mapping and showing the catch locations. However, it turns out Google expects a payment method to be added to the account, prior to providing access to their API. I was not comfortable with the notion; therefore, I chose a different solution – Open Layers. They do not require a payment method, nor an API key to use their maps. And since the catch locations will be in water, accuracy of street mapping is not important.
5. I initially wanted to design the website to only be available in the language it was programmed in (i.e. English). However, as part of accessibility, I structured my code in such a way, to allow for "language templating". This means that the website has a language template – a file which contains definitions of every bit of text. Translating the file to a different language, would allow the administrators to change the language by simply changing a single line of code in the configuration file.

```
/*
 * Email field name
 */
$email_field = "Email";

/*
 * First name field
 */
$first_name_field = "First name";

/*
 * Last name field
 */
$last_name_field = "Last name";
```

Figure 29 Language template

8. Achievements

The development overall was a “bumpy ride”. Some parts were easy, as I have done them before, others were difficult. There are of course, a lot of parts I am proud of. There are also a few I think I could have done better, if I would have managed my time better.

The main part I am quite proud of is the user management system. I believe my system is efficient and works well. It is also the part of the website I have spent a considerable amount of time on.

I am also quite happy with my implementation of anti-CSRF protection. It is a very simple system, which did not require a lot of code. I am happy that such a simple system, can provide a solution to such a widespread problem.

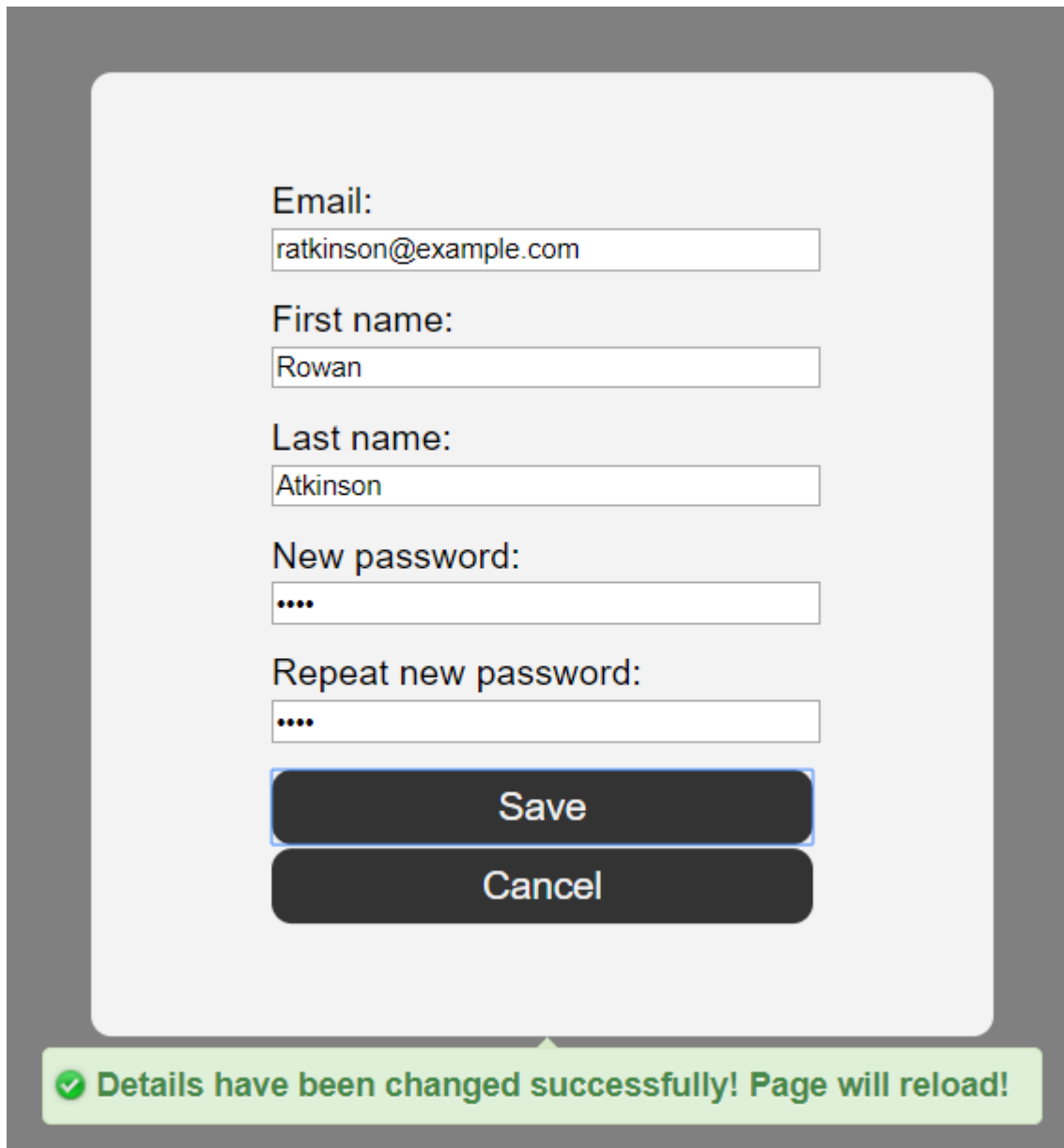
```
<div id="login">Login</div>
Email:
<br>
<input id="email" class="field" type="email" name="email" placeholder="name@example.com" required="">
<br>
Password:
<br>
<input id="password" class="field" type="password" name="password" placeholder="....." required="">
<br>
<input id="token" type="hidden" value="d8a11d5bbb267e3e8d6501734ff1c79bda43fcc9599fcb6d">
<input id="login_button" type="submit" name="login" value="Submit">
</form>
<br>
<button id="p-reset">Forgotten password?</button>
</div>
<div id="overlay"></div>
<div id="reset-form"></div>
</body>
</html>
```

Figure 30 Hidden token input

Fig. 30 shows a hidden input called “token”. This is the backbone of the anti-CSRF system. When a user visits a website, the server creates a session for that user. They are then assigned an ID, which is stored in a *PHPSESSID* cookie. The user is assigned a random token value, which is then submitted alongside the form, and is validated before any action takes place. This makes it hard for anyone to impersonate the user.

I have previously had limited experience with PDO, because I preferred the PHP’s procedural MySQLi implementation [18], due to my limited understanding of PHP’s objects. However, a guide by [phpdelusions.net](#) [19] has helped me out a lot, it provided a lot of code examples, and “good practices” to follow when using PDO.

The client wanted a website which would allow the users to browse a list of crustaceans. I saw this as the main requirement, and I implemented it. I believe it works quite well. Another request was to implement different queries. Unfortunately, I did not achieve this. There were a couple of reasons for this. One being that the client did not know what they wanted; therefore, I could not predict what queries would be useful to someone working with crustaceans. Another reason is the current pandemic, which caused me a lot of stress and anxiety, due to which I had trouble concentrating. I will write more on this in the final part.

A screenshot of a user profile update form. The form is light gray with rounded corners and is set against a dark gray background. It contains several input fields: 'Email:' with the value 'ratkinson@example.com', 'First name:' with 'Rowan', 'Last name:' with 'Atkinson', 'New password:' with four dots, and 'Repeat new password:' with four dots. Below the fields are two dark gray buttons with white text: 'Save' and 'Cancel'. At the bottom of the form, a green banner with a white checkmark icon and the text 'Details have been changed successfully! Page will reload!' is displayed.

Email:
ratkinson@example.com

First name:
Rowan

Last name:
Atkinson

New password:
....

Repeat new password:
....

Save

Cancel

✓ Details have been changed successfully! Page will reload!

Figure 31 User detail change success message

Fig. 31 shows a success message after the user has updated their password. These messages were done using a jQuery extension called Notify.js [20]. I decided to use it because the documentation was very clear, and the extension was very simple and easy to use. Because of it, I have made the website very provide a lot of feedback to the user's actions.

The image shows a user registration form with the following fields and options:

- Email:
- First name:
- Last name:
- New password:
- Administrator: ☐
- API Access: ☐
- Buttons: Save, Cancel

At the bottom, a red error message box states: **! User with that email address already exists**

Figure 32 An error notification

Fig. 32 shows a Notify.js generated error message.



Figure 33 A list of crustaceans

Fig. 33 shows a list of crustaceans. This is one of the main parts of the website. It dynamically generates a list of thumbnails, with some information underneath to help the user identify a record. Both images and the text underneath are hyperlinks, leading to the record page, with more

information. The record on the right has a black thumbnail. This is an indication that the record does not contain an image. This is because the client has explicitly told me that not all records will have an image, but all images will have metadata (i.e. a record) associated with them.

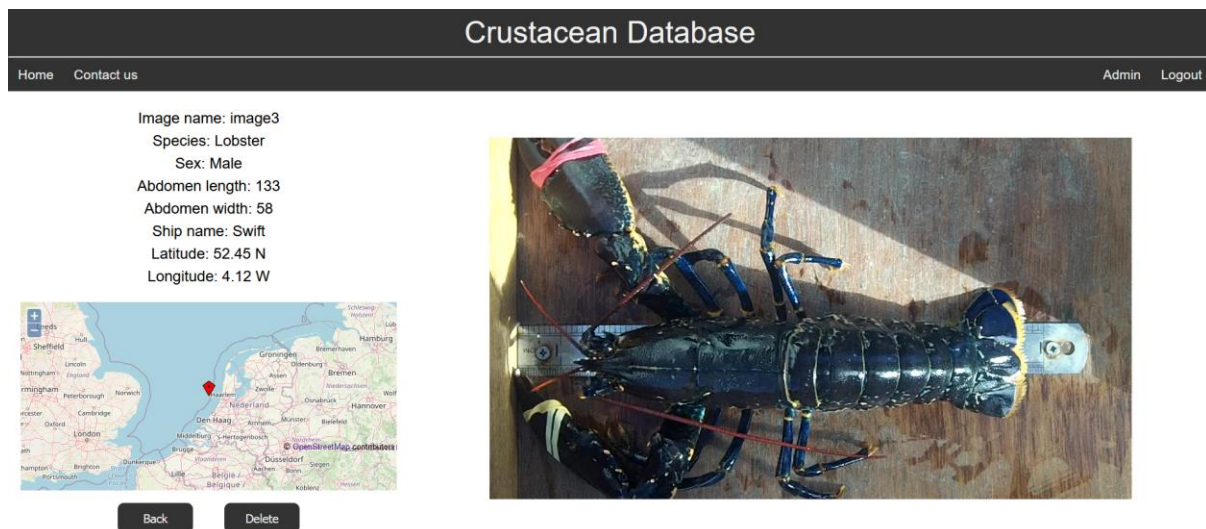


Figure 34 Crustacean listing

Fig. 34 shows a male lobster. This page shows more detailed information about an animal from the list. All the data is generated from the database, including the image – which is a base64-encoded BLOB. The map contains a marker placed at the coordinates of the catch. Only logged in users will have access to the coordinates and the map.



Figure 35 Public view of a crustacean listing

Fig. 35 shows a nearly identical page to fig. 34. The only differences are the lack of coordinates, and the map. This was done to protect the fishing spots, which are considered to be sensitive information by the fishermen.

Crustacean Database

[Home](#)
[List](#)
[Contact us](#)

[Admin](#)
[Logout](#)

User list

New User

Email	First name	Last name	Admin	Disabled	API Key	
dec21@aber.ac.uk	Dafydd	Stumbra	Yes	Yes	None	<div>Edit</div>
dec24@aber.ac.uk	staff	account	No	No	None	<div>Edit</div>
jsmith@example.com	John	Smith	No	No	5aaba688a24198da2143e3a7f2c22c	<div>Edit</div>
ratkinson@example.com	Rowan	Atkinson	Yes	No	6461433ac4748d84e4443d50782b4e	<div>Edit</div>
test2@example.com	test	one	No	No	None	<div>Edit</div>
test@example.com	Test	User	Yes	No	fed13f8b620c2ca62585c2eda9f8f	<div>Edit</div>

Figure 36 User list/admin page

Fig. 36 is that of the admin page. It is quite empty, and only contains the user list. While I am quite proud and happy of the way the user management system turned out, I was also going to add a lot more. Most of the options available in the configuration file, were supposed to be available here. However, I could not implement them due to the lack of time.

Crustacean Database

[Home](#)
[List](#)
[Contact us](#)

[Admin](#)
[Logout](#)

Contact Us

Name:

John Smith


Email:

name@example.com

Message:

Please enter your message!

Submit


protected by reCAPTCHA

Privacy
Terms

Figure 37 Contact page

Fig. 37 contains the simple, yet useful contact form. It is also protected by Google’s reCaptcha v3 [21]. The form is very simple and contains three fields. When the user submits the form, if it passes the captcha validation, an administrator will receive the email. The recipient is specified in the configuration file – this is one of the functions that was supposed to be available in the administration panel. The “protected by reCaptcha” text is visible in the bottom right corner.

9. Testing and Evaluation

9.1. Self-testing

I have performed a lot of tests throughout the entire course of the development. However, I have also performed a set of tests which are meant to test the core functions of the developed product. I did not write any tests for functions I planned to but did not implement.

These tests are similar to the tests which were performed by volunteers, but also contain some additional technical tests, which are meant to test parts not generally seen by the end user.

Test	Expected outcome	Actual outcome	Result
Login to the website	Login is successful	Login successful	Pass
Contact the page administrators	Form succeeds and email is sent	Email sent/received	Pass
Change your email on the account	Email change is successful	Email changed	Pass
Find out information about record "image2"	Data is visible	Information found	Pass
Create a new user	User creation successful	User created successfully	Pass
Create them an "API Key" in the edit user menu	API Key creation successful	API Key created successfully	Pass
Disable the account	Account is disabled	User is disabled	Pass
Remove the user's API key	API Key is removed	API key removed successfully	Pass
Create a user with the same email	User creation fails	User was not created	Pass
Delete the user	Deletion successful	User deleted successfully	Pass
Delete record "image2"	Deletion successful	Record deleted successfully	Pass
Attempt a login by deleting the anti-CSRF token	Login fails	Login failed	Pass
Attempt to submit a contact form by deleting Google's token	Submissions fails	Form failed to submit/No email received	Pass
Attempt to change user details by deleting the anti-CSRF token	Detail change fails	Form returned an error	Pass
Request a password reset	Form succeeded/Email received	Received a reset token	Pass
Reset the password using the reset token	Password reset successful	Password reset	Pass

Figure 38 List of tests performed

As I expected, the system passed all the tests. This is because I continuously tested the system during development, to make sure I did not leave some obvious bugs, which might be difficult to fix later.

Unfortunately, I did not record the tests I did during development. This was because I tested nearly every line of code, rather than testing by function, which would have led to me having to write hundreds of tests.

9.2. User testing

According to Nielsen [22], the ideal number of testers is 5. However, I did not have the opportunity to have 5 testers. I have asked 3 different people to test the website.

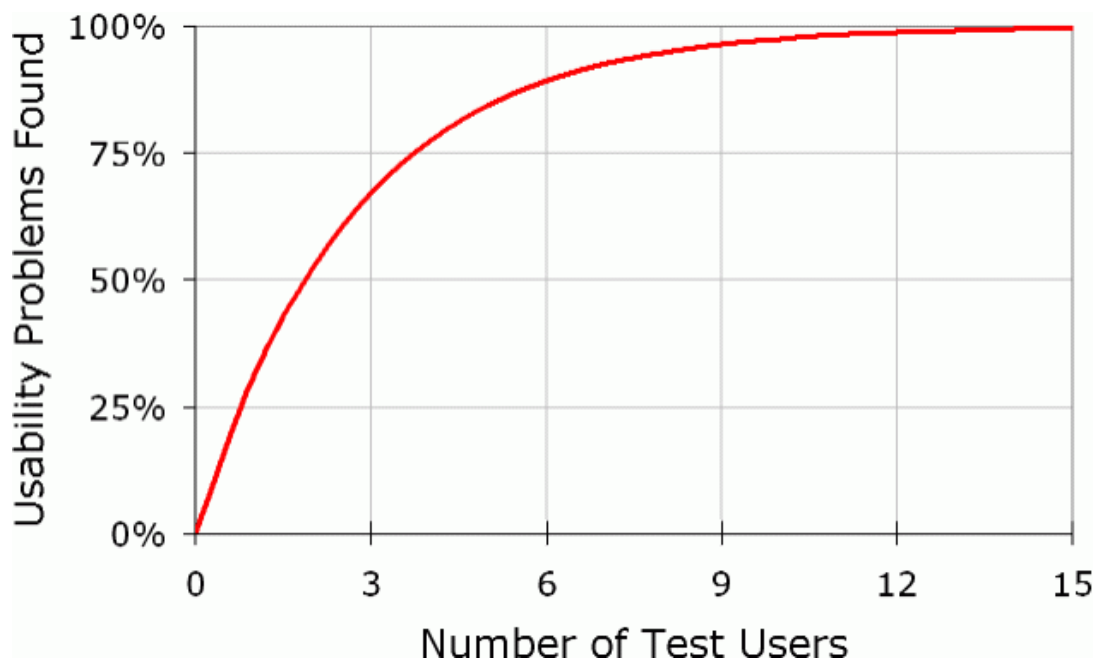


Figure 39 Nielsen's Problems and users' chart

According to Nielsen, even with just 3 testers, we can find around 65% of all existing usability problems.

The website is hosted in a location only accessible to those on campus, or people with access to the University VPN service. Two people I asked even reported not being able to use the VPN because it simply would not connect. Face-to-face testing was not an option because of the current social distancing rules.

Test	Expected outcome	Actual outcome	Result
Login to the website	Login is successful	Log in successfull	Pass
Contact the page administrators	Form succeeds and email is sent	Contact form works	Pass
Change your email on the account	Email change is successful	Successfully	Pass
Find out information about record "image2"	Data is visible	Found information	Pass
Create a new user	User creation successful	User created	Pass
Create them an "API Key" in the edit user menu	API Key creation successful	API key created successfully	Pass
Disable the account	Account is disabled	Account disabled succesfully	Pass
Remove the user's API key	API Key is removed	Succesfully	Pass
Create a user with the same email	User creation fails	User not created	Pass
Delete the user	Deletion successful	Delete successfull	Pass
Delete record "image2"	Deletion successful	Delete sucssessfully	Pass
Overall			Pass

Comments

Fairly easy to use and navigate around.
 Found layout natural.
 Dark colour schemes work well.

Figure 40 Volunteer #1 test results

Fig. 39 shows the tests I asked the user to perform, as well as the outcome, and their comments on the website. The user found the website easy to use and liked the “natural flow” of the content. They also found the darker colour scheme gentler on the eyes (*not mentioned*).

Test	Expected outcome	Actual outcome	Result
Login to the website	Login is successful	Login successful	Pass
Contact the page administrators	Form succeeds and email is sent	Contact form works	Pass
Change your email on the account	Email change is successful	Email changed	Pass
Find out information about record "image2"	Data is visible	Data found without any difficulty	Pass
Create a new user	User creation successful	User created with some difficulty	Pass
Create them an "API Key" in the edit user menu	API Key creation successful	API key created without any difficulty	Pass
Disable the account	Account is disabled	Account disabled without any difficulty	Pass
Remove the user's API key	API Key is removed	API key removed without any difficulty	Pass
Create a user with the same email	User creation fails	User was not created	Pass
Delete the user	Deletion successful	User deleted without any difficulty	Pass
Delete record "image2"	Deletion successful	Record deleted without any difficulty	Pass
Overall			Pass

Comments

Straightforward and easy to use
 Takes a few moments to get used to the layout
 Website layout works well and is easy to understand

Figure 41 Volunteer #2 test results

Fig. 40 Mentions having some difficulty in creating a user account as an administrator. They did not have any trouble doing anything else. They also mentioned that it takes a few moments to get used to the layout. This is normal because seeing a new website will nearly always result in having to take a few moments to look around.

Test	Expected outcome	Actual outcome	Result
Login to the website	Login is successful	Login successful	Pass
Contact the page administrators	Form succeeds and email is sent	Email sent/received	Pass
Change your email on the account	Email change is successful	Email changed	Pass
Find out information about record "image2"	Data is visible	Information found	Pass
Create a new user	User creation successful	User created with difficulty	Pass
Create them an "API Key" in the edit user menu	API Key creation successful	API Key created successfully	Pass
Disable the account	Account is disabled	User is disabled	Pass
Remove the user's API key	API Key is removed	API key removed successfully	Pass
Create a user with the same email	User creation fails	User was not created	Pass
Delete the user	Deletion successful	User deleted successfully, account was made administrator and was disabled	Pass
Delete record "image2"	Deletion successful	Record deleted successfully	Pass
			Overall Pass

Comments

Login page is too empty/bland
 User pages are empty and blank
 Found it very easy to use

Found all the necessary information

Figure 42 Volunteer #3 test results

Fig. 41 shows the volunteer having problems creating a user account as an administrator. This is perhaps an issue with the button placement, where the users do not immediately see where they should go to complete the action. The volunteer also found the pages to be too empty, and not containing enough information. The website does not contain a lot of information, because it is meant to be functional. The website is based on providing information about crustaceans. Perhaps a different layout could have worked better.

Despite that, they found it easy to use, and found all the necessary information.

9.3. Testing summary

Testing went well, and the system has passed all the tests.

During the user testing phase, 2 out of 3 people reported difficulty creating a user. This could be a layout issue.

One volunteer also thought the pages were too empty, which once again, indicated a layout problem. This is especially likely to be the case because we cannot add more information as there is nothing else to add.

The solution could be to develop a few different sample layouts and get them evaluated before implementation. I am not a designer; therefore, it is of no surprise that the layouts were not ideal. I have attempted to create a simple but functional layout.

9.2.3. Ethics

The volunteers have been asked to provide their age and IT background, to better gauge which demographic were performing the tests. They were also provided with a randomly generated 4-digit

ID number, and my email address. If they wanted to withdraw their data, they could email me with their ID number, and I would remove their data from my testing list.

They were also told the purpose of the tests and how their data will be used.

9. Critical analysis

Overall, I think the project went okay. I could have most certainly done a better job if not for the current pandemic. I think this is especially true, because my previous deliverables received good marks, and shows that I could have done much better.

The pandemic has affected me quite a bit. Several of my family members are in high risk groups due to various long-term health conditions. My parents also live in central London, which puts them even more at risk. Because of this, I found it difficult to concentrate, and frequently lost focus due to near-constant anxiety and stress.

My Business analysis was not very long, but it did cover most of the use cases. However, I did not find any good examples of related websites, because I was unsure of what I should be looking for. When I received feedback for the submission, I was given some examples, which gave me a much better idea of what I was after, and really helped me understand similar existing websites, which were covered in this report. The technical report was quite thorough, and covered a lot of the scenarios, and showed every database table, and its fields. I have since added more fields, to reflect changes made during development, but the technical report still provided me a strong foundation for creating and improving the database.

The design phase is where I made many mistakes. I am not good at design, I have asked for help with design from a good friend of mine, who is studying a creative subject. And I do appreciate a lot of his help, but he was also busy a lot of the time, meaning I had to improvise while trying to keep the same style. In the end, that did not work, leading to poor design decisions.

During the implementation phase, I was initially going to use a framework. However, I quickly realised that using a framework will not be ideal, because I did not know enough about the chosen frameworks to carry out a full project. It would have caused me to spend lots of time figuring out the basics, rather than developing the actual system.

I believe I achieved the core needs of the project, and I think the website is functional enough to provide enough information on the required animals. Of course, the client also asked for query tools, but I was unable to implement this, due to the aforementioned issue, and also because they did not know what they wanted.

If I were given an opportunity to continue working on the project, I certainly would. I would likely change the whole design. I would produce a few sample designs, and ask users to evaluate them, then choose the best liked one. I would also continue to work on the core systems and implement the rest of the ones I did not have time for.

Overall, there is certainly room for improvement, but given the circumstances I put a lot of time and effort into this project, even if not all of it was as productive as I had hoped.

I certainly learned quite a lot of new things about PHP, and even different ways of approaching the same problem. I had also found out my strengths and weaknesses, especially when it comes to managing an entire project from start to finish.

Word count: 10,192

10. Appendices

10.1. Appendix A – Data Dictionary

cs39930

images

Column	Type	Null	Default	Links to	Comments	Media (MIME) type
img_name (<i>Primary</i>)	varchar(15)	No		records -> img_name		
image	mediumblob	No				
mime	varchar(25)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	img_name	3	A	No	

pass_resets

Column	Type	Null	Default	Links to	Comments	Media (MIME) type
email	varchar(50)	No		users -> email		
date	date	No				
time	time	No				
reset_key (<i>Primary</i>)	varchar(30)	No				
valid	tinyint(1)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	reset_key	8	A	No	
reset_key	BTREE	Yes	No	reset_key	8	A	No	
email	BTREE	No	No	email	4	A	No	

records

Column	Type	Null	Default	Links to	Comments	Media (MIME) type
img_name (<i>Primary</i>)	varchar(15)	No				
species	varchar(30)	No				
sex	tinyint(1)	No				
carapace_width	int(5)	Yes	NULL			

abdomen_length	int(5)	Yes	NULL			
abdomen_width	int(5)	Yes	NULL			
coords_lat	float	No				
coords_long	float	No				
boat	varchar(20)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	img_name	4	A	No	

system

Column	Type	Null	Default	Links to	Comments	Media (MIME) type
id (<i>Primary</i>)	int(4)	No				
motd	text	Yes	NULL			
user_announcement	text	Yes	NULL			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	1	A	No	

users

Column	Type	Null	Default	Links to	Comments	Media (MIME) type
email (<i>Primary</i>)	varchar(50)	No				
first_name	varchar(20)	No				
last_name	varchar(20)	No				
admin	tinyint(1)	No	0			
disabled	tinyint(1)	No	0			
password	varchar(255)	No				
api_key	varchar(30)	Yes	NULL			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	email	7	A	No	
api_key	BTREE	Yes	No	api_key	7	A	Yes	

10.2 References

- [1] "Aberystwyth University - Staff," [Online]. Available: <https://www.aber.ac.uk/en/contact-us/directory/staff/profile/ais/>. [Accessed 9 May 2020].
- [2] "Birds A-Z | Bird Guides - The RSPB," [Online]. Available: <https://www.rspb.org.uk/birds-and-wildlife/wildlife-guides/bird-a-z/>. [Accessed 9 May 2020].
- [3] "Wildlife - WWF," [Online]. Available: <https://www.wwf.org.uk/learn/wildlife>. [Accessed 9 May 2020].
- [4] "jQuery," [Online]. Available: <https://jquery.com/>. [Accessed 10 May 2020].
- [5] "PHP: password_hash - Manual," [Online]. Available: <https://www.php.net/manual/en/function.password-hash.php>. [Accessed 9 May 2020].
- [6] "Blob - Web APIs | MDN," [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/Blob>. [Accessed 9 May 2020].
- [7] "MEDIUMBLOB - MariaDB Knowledge Base," [Online]. Available: <https://mariadb.com/kb/en/mediumblob/>. [Accessed 9 May 2020].
- [8] "phpMyAdmin," [Online]. Available: <https://www.phpmyadmin.net/>. [Accessed 9 May 2020].
- [9] "PHP: random_bytes - Manual," [Online]. Available: <https://www.php.net/manual/en/function.random-bytes.php>. [Accessed May 9 2020].
- [10] "Media Types - IANA," [Online]. Available: <https://www.iana.org/assignments/media-types/media-types.xhtml#image>. [Accessed 9 May 2020].
- [11] "SQL Injection | OWASP," [Online]. Available: https://owasp.org/www-community/attacks/SQL_Injection. [Accessed 10 May 2020].
- [12] "Cross Site Scripting (XSS) Software Attack | OWASP Foundation," [Online]. Available: <https://owasp.org/www-community/attacks/xss/>. [Accessed 10 May 2020].
- [13] "Cross Site Request Forgery (CSRF) | OWASP Foundation," [Online]. Available: <https://owasp.org/www-community/attacks/csrf>. [Accessed 10 May 2020].
- [14] "Project root," [Online]. Available: <https://mmp-dec21.dcs.aber.ac.uk/project/>. [Accessed 10 May 2020].
- [15] "Welcome to Apache NetBeans," [Online]. Available: <https://netbeans.apache.org/>. [Accessed 10 May 2020].
- [16] "Protect Your Website With Anti-CSRF Tokens | Netsparker," [Online]. Available: <https://www.netsparker.com/blog/web-security/protecting-website-using-anti-csrf-token/>. [Accessed 10 May 2020].
- [17] "Bootstrap · The most popular HTML, CSS, and JS library in the world.," [Online]. Available: <https://getbootstrap.com/>. [Accessed 10 May 2020].

- [18] "PHP: MySQLi - Manual," [Online]. Available:
<https://www.php.net/manual/en/book.mysqli.php>. [Accessed 10 May 2020].
- [19] "(The only proper) PDO tutorial - Treating PHP Delusions," [Online]. Available:
<https://phpdelusions.net/pdo>. [Accessed 10 May 2020].
- [20] "Notify.js," [Online]. Available: <https://notifyjs.jpillora.com/>. [Accessed 10 May 2020].
- [21] "reCAPTCHA: Easy on Humans, Hard on Bots," [Online]. Available:
<https://www.google.com/recaptcha/intro/v3.html>. [Accessed 10 May 2020].
- [22] J. Nielsen, "Why You Only Need to Test with 5 Users," Nielsen Norman Group, 18 March 2000.
[Online]. Available: <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>. [Accessed 10 May 2020].