Joe Lloyd, Joshua Walker, Luke Gassman, Nicolas E.V., Sophie Jones

Assignment 2 - Agile

Project Team: Fr-Agile

Devloped Platform: Water Capitol

Contents Page

Word Count (Including Personal Statements): *12,590*

Introduction

Water bowsers are essential products that help maintain and protect the security and wellbeing of the public during any tragedy requiring freshwater aid. To handle these future disastrous events, the government will require a management platform to govern bowsers, alongside allowing members of the public to be informed of any news surrounding the deployment of new bowsers. The development of the water bowser management system has been documented in stages, like the present and future status of development processes. This documentation also shows how Agile was used while constructing the platform, testing of the finished product, front and backend development discussions and other key developmental aspects that concluded in an effective product. Additionally, the use of a product vision and other goals for the project were documented to improve the development and understanding. Business functions and needs are also covered and with assistance of sprint backlogs, reviews and retrospectives.

Login details

All passwords: watercapitollogin

Administrator Accounts:
- Sophie@a
- Nicolas@a
- luke@a
- josh@a
- joe@a

Council Accounts:
- Sophe@c
- Nicolas@c
- luke@c
- josh@c
- joe@c

Maintenance Accounts:
- Sophie@m
- Nicolas@m
- luke@m
- josh@m
- joe@m

Project Aspects

The Project

During a flood, clean water is hard to acquire, with water bowsers the public can have access to fresh clean water. Constituencies and local councils that are victims to a recent or devastating flood will require a management format to place, manage and maintain the water bowsers effectively. In addition, to this the public needs to be informed about where bowsers are and how to access them. Water Capitol was designed and developed to achieve these objectives, for public and council ease of use and peace of mind. This is so that this platform can aid those in need until the lack of fresh water may be alleviated.

Throughout this project, the use of agile and distributed scrum was used. Fortunately, the adoption of agile allowed us to respond quickly to unexpected national circumstances that helped manage the change for the team's development and communication. Scrum is a useful processing framework that allows the team to use sprints and other events (like sprint reviews) that help manage the product components. By acquiring this framework, the production strain of the website is alleviated through distributed work and the use of sprint reviews, to help the team evolve and unify. (Srivastava, Bhardwaj and Saraswat, 2017)

The website contains:

- A designated public section that includes:
    - The basic information about the current effects in the area through a Twitter API
    - The bowsers near the current user with additional information on the status of bowsers
    - Reporting any repairs or problems about bowsers near a user
    - A frequently asked questions section alongside a contact us page
- A council and maintenance section containing:
    - A maintenance platform with access to the tasks page and the discussion page for ease of use when using mobiles.
    - A secure login page
    - Main bowser information, statistics and a live feed of the bowsers
    - Bowser deployment and management
    - Report and inquiry page for public issues
    - Task creation with details and the possibility of acquiring tasks
    - A discussions page for connectivity and communication between employees

- An administrator section containing:
    - A statistical view of all work being done for management
    - An employee accounts overview page for moderation, updating and deletion if necessary
    - A constituencies page for the overview of all designated constituencies
    - A discussion board to contact the other employees if necessary

Page Designs

The initial implemented designs were developed from sketched wire frames, which gave a dependable visual representation of the connecting pages and their design patterns. Using this, the development was eased, and the production of the official pages could commence. Going from the wire frames in diagrams PA1-PA3, the development of the web pages could begin whilst still keeping a concise and similar look in each section by referring back to the initial drawings. Maintenance and council employees are the main users of the website. Therefore, more functionality was needed for maintenance and council opposed to the public sections. The additional functions for adding bowsers, getting requests and departing maintenance workers to the bowsers, are all required for the management of the current crisis and require the most attention when constructing the application. Many of the page designs have not altered vastly from the implemented end product, given the clear thought process used before the construction of the pages themselves.

Council and maintenance

The council and maintenance page's wireframe visible in diagram PA1 are some of the most important pages on the website, given its management and control of the water bowsers, as well as the responsibility they have over people's lives during a crisis.

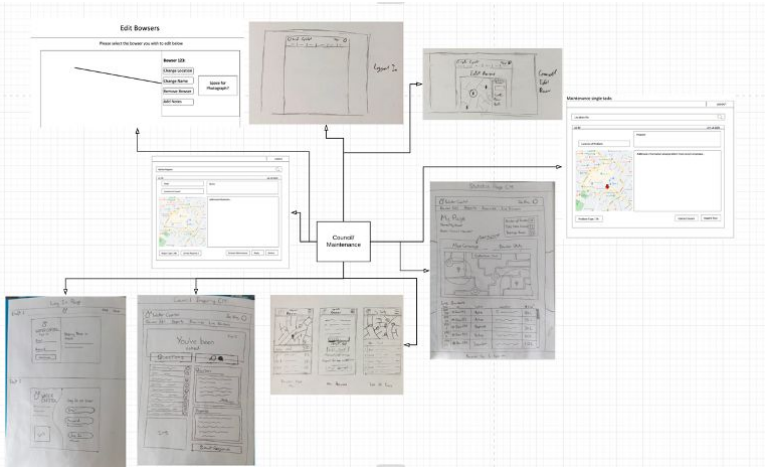


Diagram PA1

Admin

Compared to the council and maintenance side of the application, the administrator has access to the accounts and the statistical view of the work being completed by the council and maintenance employees. Their main purpose is to oversee the running of the platform and to ensure that each employee is working effectively to help transition through the crisis.

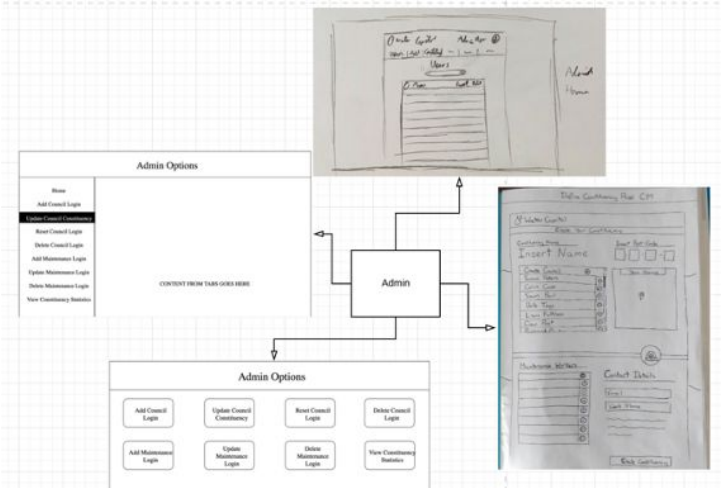


Diagram PA2

Public pages

The public also required their own pages, so that they could visualise the events occurring in their location. Reassuring the public was a central goal during development so the ability to ask and answer key questions was implemented into new page designs to help put the public at ease. Further stated, the public has access to the reporting of any problems, so that council members can manage and set tasks based on the importance of the inquiry.

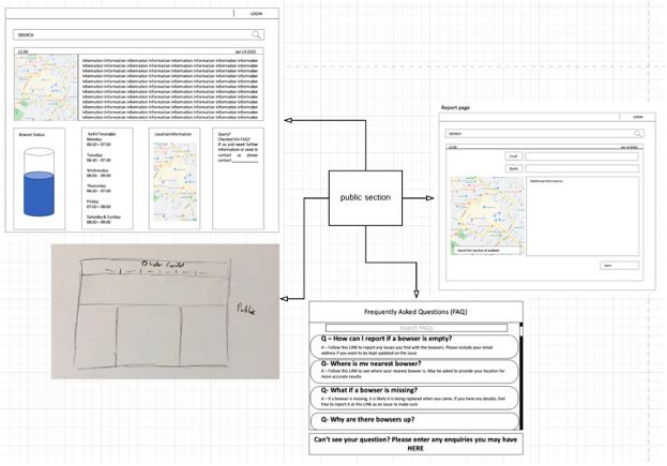


Diagram PA3

Project vision

Team:

Team Name: Fr-Agile

Team Members:	Luke Gassmann
	Joe Lloyd
	Nicolas Euliarte Veliez
	Sophie Jones
	Josh Walker

Which degrees are you each studying towards?

Luke Gassmann:	Computing
Joe Lloyd:	Computing
Nicolas Euliarte Veliez:	Computing
Sophie Jones:	Business Computing

Josh Walker: Computing

Aspirations:

What do each of you want to get out of the project and CT5038 apart from a pass? (Each member writes a sentence)

Luke:

I want to appreciate and work with a dynamic and adaptive team that works together effectively, so that I can apply what I have learnt to the real world in my future ambitions.

Joe:

I am aiming for a first but more importantly I want to work in a team which can support each other's weaknesses in order to become more effective

Josh:

I want to work as part of a cooperative agile team that will make a comprehensive and easy to use piece of software with the aim of a first in this module

Nicolas:

I want to learn professional implementations of product development to further my understanding of industry standards.

Sophie:

I want to understand how this helps an organisation from a project aspect but also in general in the running of a business day to day.

We believe that we want to pursue our highest possible grade and develop team building by using high quality Agile principles like communication and a focus on the product.

Commitments:

How much time is each of you prepared to commit to this project each week?

Time boxing has shown us that we can offer 7 hours to this project a week in consideration to our other commitments. We believe that allows for the team to have enough face to face time as possible to actively make good progress through the development of the project.

Which day and at what time will this team meet to work together?

(Recommendation is that you are together for at least 3 hours each week)

We will meet Tuesdays and Thursdays due to our personal schedules and commitments. This will allow for the team to have an estimated 7 hours of face to face contact and possible online video meetings on other days when possible.

Organisation:

Where will you keep your Product & Sprint backlogs and any other documentation you are working on so that it is available to all team members?

We are storing project information on Jira and Google Drive so that all team members can access the information as and when needed to produce their allocated sections of the application.

How will you allocate Sprint backlog tasks to team members?

Sprint Backlog tasks will be assigned based on the competence needed in the area requested. Additionally, peer programming may be used in areas when needed to further develop team member skills overall.

We will review progress at each team meeting as well as through a WhatsApp group-chat when asking for prerequisites for example. Completed tasks will be marked as done on the Sprint backlog on Jira.

How and when will you prepare for the Sprint review meeting?

(At a Sprint review you demonstrate the functionality in your application to the client using your test data. There will be 2 Sprint review meetings during the project.)

Preparation will involve ensuring the most up-to-date version of the application is available, rehearsing a demonstration as well as bringing a check list of any specific features we wish to ask about or change. We will also be making notes on any improvements and positives the client has. We will also be showing test data to show how well functions are working and we will provide a list of known issues which may be experienced but will be fixed as soon as possible. We will also allow the client to use the website themselves to encourage their feedback.

Product Vision Box:

Please attach additional sheets of paper that show your Product Vision Box and Product Vision Statement.

<table>
<tr><td colspan="4">Vision Statement:
This product is made to help counteract the effects made by several natural disasters within the United Kingdom by providing an application to manage water shortages and inform the public.</td></tr>
<tr><td>Target Group:
Governments and constituencies</td><td>Needs:
Services need to be in place to help those who lack the basic living conditions necessary to support both themselves and loved ones around them in times of need. Thus, we argue that a future with security and protection during disastrous circumstances is better than no protection at all.</td><td>Product: Water Capitol
A website to incorporate methods of support and accountability for council members. Focused on water shortages, our system will combine both effective water shortage managerial tools, with modern front-end technology to allow both the public and council confidence in responding to unpredicted circumstances.</td><td>Business goals:
To help others in times of need, as we believe that every life is important.</td></tr>
</table>

Figure PV1

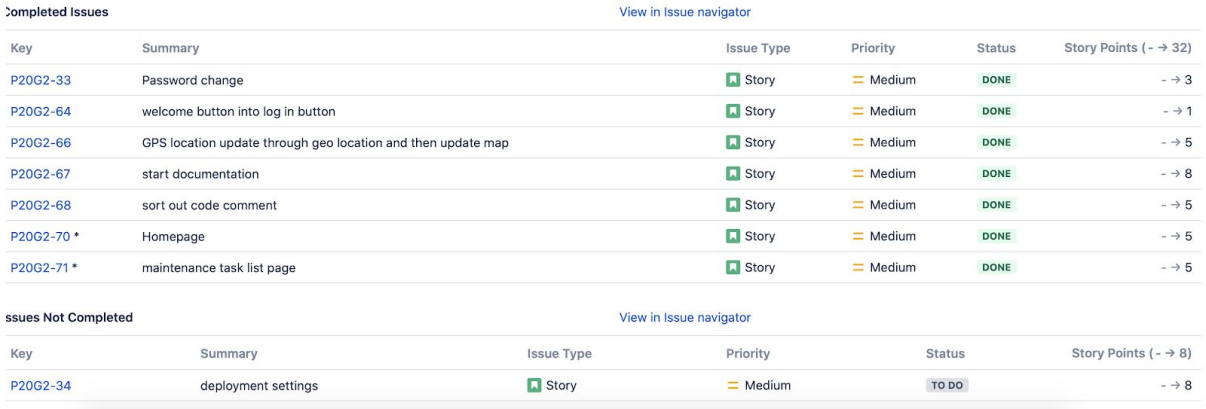
Business Functions vs Needs

The main requirement that businesses need to recognise is the importance of a fully functioning website over the business extensive necessities. The business needs a separation of functionality and unnecessary priorities to the finished product. Referring to this, one issue that agile has to overcome, are stakeholder's putting in place additional unnecessary requirements. These unnecessary needs do not benefit the organisation or the product's main purpose even though stakeholders may want the unnecessary additions.

Business needs are implemented to help overcome the problems the project is constructed to overcome. The need is something that will not change the result, an example, having a different font text over the one that was originally used. This wouldn't take long to change but this is not a requirement it is a need. One example of this is when the team needed to complete the project and include mobile support. The reason for this business need was due to maintenance employees not carrying larger devices when in remote locations, however sorting out jobs and repairs are a main requirement. (Business Needs vs. Requirements | moelgendy blog, 2020).

Sprint backlogs

Sprint backlogs were used to monitor the workflow of the team. The team decided at each sprint meeting what the next steps were to get the project functioning and working. The team used software called Jira to plan and manage the sprints and user stories. Jira was advantageous because creating the backlogs was useful since the team could plan and prepare.



:ompleted Issues — View in Issue navigator

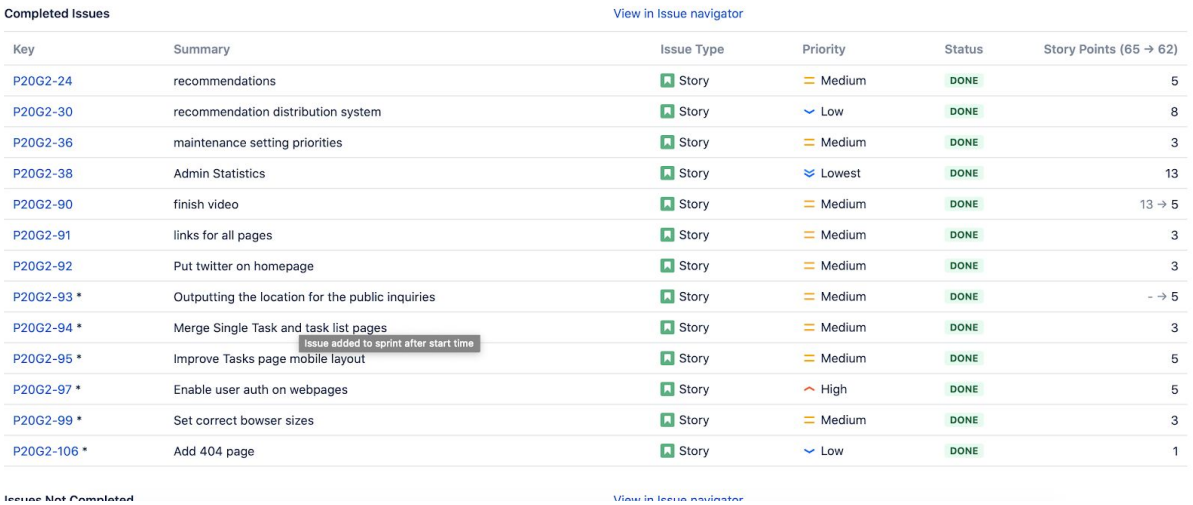
Key	Summary	Issue Type	Priority	Status	Story Points (- → 32)
P20G2-33	Password change	Story	Medium	DONE	- → 3
P20G2-64	welcome button into log in button	Story	Medium	DONE	- → 1
P20G2-66	GPS location update through geo location and then update map	Story	Medium	DONE	- → 5
P20G2-67	start documentation	Story	Medium	DONE	- → 8
P20G2-68	sort out code comment	Story	Medium	DONE	- → 5
P20G2-70 *	Homepage	Story	Medium	DONE	- → 5
P20G2-71 *	maintenance task list page	Story	Medium	DONE	- → 5

ssues Not Completed — View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (- → 8)
P20G2-34	deployment settings	Story	Medium	TO DO	- → 8

Figure SB1

Figure SB1 shows the sprint backlog for when the sprint needed to be done. What went into the sprint backlog was what the team believed were the next steps forward for the project. With any project, foundations need to be laid to ensure that the rest of the project can be developed. Additionally, by documenting the tasks the development team was assisted, as tasks can easily be checked, and developers can keep track of their assigned tasks.



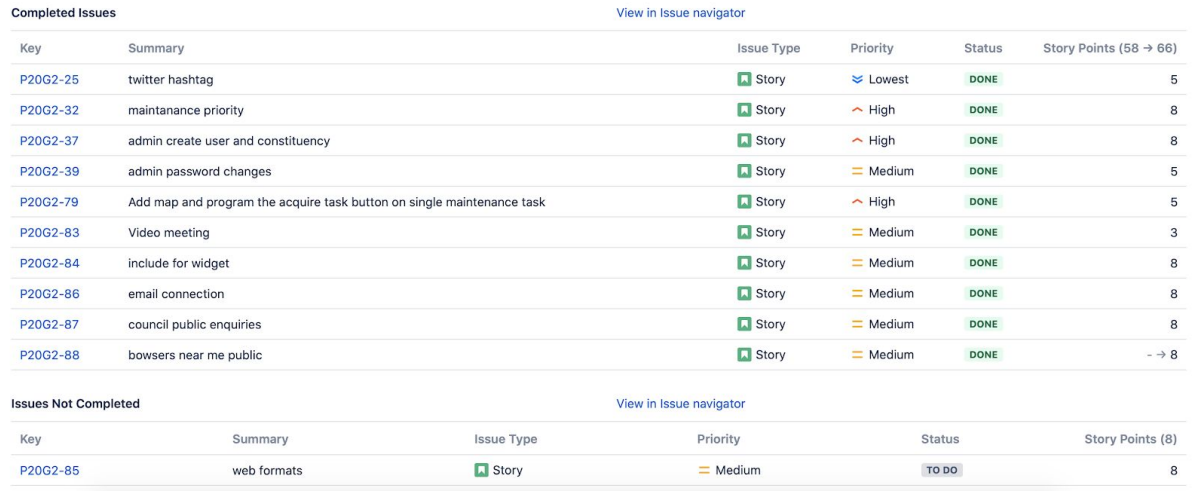
Completed Issues View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (65 → 62)
P20G2-24	recommendations	Story	Medium	DONE	5
P20G2-30	recommendation distribution system	Story	Low	DONE	8
P20G2-36	maintenance setting priorities	Story	Medium	DONE	3
P20G2-38	Admin Statistics	Story	Lowest	DONE	13
P20G2-90	finish video	Story	Medium	DONE	13 → 5
P20G2-91	links for all pages	Story	Medium	DONE	3
P20G2-92	Put twitter on homepage	Story	Medium	DONE	3
P20G2-93 *	Outputting the location for the public inquiries	Story	Medium	DONE	- → 5
P20G2-94 *	Merge Single Task and task list pages	Story	Medium	DONE	3
P20G2-95 *	Improve Tasks page mobile layout	Story	Medium	DONE	5
P20G2-97 *	Enable user auth on webpages	Story	High	DONE	5
P20G2-99 *	Set correct bowser sizes	Story	Medium	DONE	3
P20G2-106 *	Add 404 page	Story	Low	DONE	1

Issue added to sprint after start time

Figure SB2

With each sprint, the team would add what they agreed would be the next steps forward in getting the project completed. As the team progresses they learn what their velocity is, so there is trial and error in determining the number of effort-points for each sprint. From Figure SB2-SB3, it is evident between the sprint backlogs that trial and error was used to get the correct velocity.



Completed Issues View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (58 → 66)
P20G2-25	twitter hashtag	Story	Lowest	DONE	5
P20G2-32	maintanance priority	Story	High	DONE	8
P20G2-37	admin create user and constituency	Story	High	DONE	8
P20G2-39	admin password changes	Story	Medium	DONE	5
P20G2-79	Add map and program the acquire task button on single maintenance task	Story	High	DONE	5
P20G2-83	Video meeting	Story	Medium	DONE	3
P20G2-84	include for widget	Story	Medium	DONE	8
P20G2-86	email connection	Story	Medium	DONE	8
P20G2-87	council public enquiries	Story	Medium	DONE	8
P20G2-88	bowsers near me public	Story	Medium	DONE	- → 8

Issues Not Completed View in Issue navigator

Key	Summary	Issue Type	Priority	Status	Story Points (8)
P20G2-85	web formats	Story	Medium	TO DO	8

Figure SB3

Agile’s Assistance to the Project

Team evolution

The group started by sitting down and discussing which roles the team felt comfortable in development, in the end, there were 4 programmers and scrum master. When they first met they discussed the objectives that they wanted as a group, this was a timely process as they also discussed how they would communicate because they would not see each other daily and from that communication was done through WhatsApp.

Setting out the goals was a key benefit, as team members knew what was expected from one another. If a team member was not contributing and was considered a burden, the team would apply countermeasures, however, this did not arise. Another issue they had to overcome was trust, since the team had not previously worked together, fears and objections needed to be overcome. Over time, they learned to trust each other, and this improved over the weeks as they learned each other's strengths, and which could assist in allocating tasks.

The next meeting, they started the first sprint. Each sprint meeting included a sprint retrospective, to measure progress on the minimal viable product and to measure the following weeks' effort points which improved with trial and error. This was to ensure that they were working effectively, but also to not misjudge the workload. Meeting every week helped the team's understanding of strengths and weaknesses.

Using WhatsApp was also a great decision by the team, since not meeting daily, if a team member had any queries, issues or wanted to keep up to date, they could discuss it through WhatsApp. This maintained communication and helped the team grow together.

Due to national obstacles the team adapted to distributed development. Since the team was not allowed to meet due to these circumstances, they adapted to using skype for the weekly meetings. The team relied on these forms of communication to keep the group informed of any progression. The main priority when using distributed scrum was that the team still met to discuss development. The key was working efficiently to the best of the team’s ability. This was done through Skype and Microsoft teams for sprint reviews. This took some time to get used to, but once comfortable they carried on working as normal.

Distributed development was initially difficult, as the team was used to seeing everyone once a week for sprint meetings. Although this was not the same as meeting in person, the team learned to adapt which is an important skill in agile. With reduced communication, there was lack of time management which made pair programming and meeting initially difficult. Throughout this, WhatsApp was used as the main form of communication.

As a team, there were some benefits. When working together cooperatively, team members assisted one another, without judgment, when needed. When the team needed adapting, they did so as a team. If a team member was indecisive, they could discuss it with the team, changes were made so that the group agreed so everyone was content.

Additional benefits were that team members were liable for their own tasks. This meant if a team member did not do all the work it affected the whole team. Although team members may get frustrated, scrum alleviated this through retrospectives and story points would be placed within the next sprint. Additionally, using agile increased the team's trust and assisted in relying upon one another.

Agile has helped the development of the website, through short development sprints. Additionally, having a team with mixed skills meant that there were developers to program the website and to create and link the database using PHP. Having a person to speak to the stakeholder also ensured that they meet requirements and could ask questions to prevent confusion.

Overcoming drawbacks that prevented the team from working together was a valuable life lesson, which will help in future real-world projects by learning how to adapt in new environments. Alongside working with others, trust issues may occur, since the team may have not worked together beforehand and may have work-ethic expectations since they may not know how a new team works.

Throughout the project, the team learned to work collectively and to make sure it was completed on time ready for each sprint review. When team members were stressed about not completing it on time, they were encouraged by one another to finish the task and approached as a team.

Adopting scrum

As a team they adopted scrum ceremonies, such as daily stand-ups, sprint retrospectives, sprint review meetings and a sprint backlog. These were used so team members knew where the team was during the development phase. Knowing the progress is key because if the team is not aware, then it can lead to complications further down the line. When communicating with stakeholders, if the team doesn’t know the progress it can come across unprofessional.

When assigning the roles, the team needs to know what they are doing, so that people know their primary roles and what is expected of them. For example, the scrum master’s role facilitates the team, by reducing distraction and writing paperwork. The development team is there to make stakeholder requirements into an actual product.

User stories are useful for the team as they set out requirements from the stakeholders to ensure that the team is on the right track. Without these user stories, the development team knows what to work on. This is so they do not deviate from the project, as

this can also lead to adding items into the project that the stakeholder may possibly not want to lead to redundancy.

Additionally, with the user stories the product owner prioritizes the backlog to ensure the main key factors are met. This is also used to make sure that business needs are met over business wants. By doing so this can ensure that the development can be done to stakeholder requirements. If the stakeholder wants a simple but effective looking website the development team needs to do the background tasks first before reaching the final product.

Sprint retrospectives are a great way to have a meeting with the stakeholders and to get their opinion on how the project is progressing. This is the time where the stakeholders can see a minimal viable product and also request changes on the product, as it progresses so it is not incomplete by on release. The main benefit of this is customer satisfaction, as this means the team is likely to get rehired. An additional benefit is the feedback loop so the team can know the positive and negatives of the development.

Sprint planning meetings come at the end of a sprint or after the sprint retrospectives. This is so the team can decide what they believe needs to be done, instead of having the product owner and the stakeholder deciding. This is because the stakeholder might not have the technical knowledge to enhance the development. The team dynamic is assisted through the backlog as the team members can decide on their tasks based on their strengths and remain accountable.

One of the final ceremonies was product backlog refinement, this is when the team breaks down epics into more manageable tasks. For example, the login page epic initially seemed simple but actually contained complex work such as connecting to the database, resetting passwords and logging in. Backlog refinement assisted the team in knowing what needed to be done.

Sprint Review Minutes

Taking sprint review minutes is very important as the team needs to know what they discussed each week so that if they need a recap on what has been done they can easily do so. Within Appendix 3 are the minutes from all sprint meetings as well as the sprint retrospectives. Since the team was not meeting daily this was handy and it also displays how the team worked every week. The longer the project went on the shorter the meetings went since they were coming to the end of the project. The final weeks were just finalising everything to ensure that it was functional and was ready to be handed over to the stakeholders for a final review.

Unit Testing

Alongside traditional testing methods, such as exploratory, usability and functionality testing, which were done during coding to ensure that each feature was implemented as

intended and with ease of use; unit testing was implemented to ensure that key functionality of the website behaves as intended.

Implementing unit testing for each page was important for three main reasons:

- To ensure that the page behaved as expected without the human error of assuming.
- To ensure that future changes did not cause a regression in the functionality or features.
- To save time when by avoiding manually testing each page when retesting features was required.

To implement unit testing on the server, Laravel's Testing API was used as it provides both DOM testing and PHP unit testing, it also has the advantage over in browser unit testing as it provides a central repository for the unit tests on the server with the code it is testing compared to running them on a local machine.

Due to the importance of security authentication, unit tests were written to check the access rights of each page for each user type by checking if a fake user of each type could access the page or not. Fake users were also used to ensure that the logout and login pages functioned as intended.

```php
class LoginTest extends TestCase {

    public function testGuestAccess()
    {
        $response = $this->call('GET', '/login');

        $response->assertResponseOk();
    }

    public function testCouncilAccess()
    {...}

    public function testAdminAccess()
    {...}

    public function testMaintenanceAccess()
    {...}

    public function testCouncilLogin()
    {
        $password = '1234';
        $user = factory(User::class)->create([
            'user_type' => 'C',
            'password' => bcrypt($password),
        ]);

        $response = $this->from('/login')->post('/login_submit', [
            'email' => $user->email,
            'password' => $password,
        ]);

        $response->assertRedirect('/councilhome');

        $this->assertAuthenticatedAs($user);
    }

    public function testAdminLogin()
    {...}

    public function testMaintenanceLogin()
    {...}

    public function testInvalidLogin()
    {
        $user = factory(User::class)->create([
            'user_type' => 'C',
            'password' => bcrypt('1234'),
        ]);

        $response = $this->from('/login')->post('/login_submit', [
            'email' => $user->email,
            'password' => 'abcd',
        ]);

        $response->assertRedirect('/login');

        $this->assertGuest();
    }
```

Figure UT1

Unit testing was also used to ensure the functionality of page links, by using Laravel to check for the redirects when DOM elements are clicked. While in this project using Laravel 4.2 this is a simulated environment, more recent versions of Laravel (5.5+) run a copy of Google Chrome on the server to check the real-world behaviour.

In total 175 tests have been written.

```
public function testLinkMenuAreas()
{
    $user = factory(User::class)->create([
        'user_type' => 'A',
    ]);

    $this->actingAs($user)
         ->visit('/adminHome')
         ->click('Constituencies')
         ->seePageIs('/adminConstituencies');
}

public function testLinkMenuChat()
{
    $user = factory(User::class)->create([
        'user_type' => 'A',
    ]);

    $this->actingAs($user)
         ->visit('/adminHome')
         ->click('Discussions')
         ->seePageIs('/messageDesktop');
}

public function testLinkMenuLogout()
{
    $user = factory(User::class)->create([
        'user_type' => 'A',
    ]);

    $this->actingAs($user)
         ->visit('/adminHome')
         ->click('Log Out')
         ->seePageIs('/login')
         ->assertGuest();
}
```

Figure UT2

```
PHPUnit 3.7.38 by Sebastian Bergmann.

Configuration read from /var/www/vhosts/ct5038group2-ct5038.uogs.co.uk/httpdocs/Blog/phpunit.xml

. . . . . 63 / 175 ( 36%)
. . . . . 126 / 175 ( 72%)
. . . . . .

Time: 1.61 seconds, Memory: 94.00MB

OK (175 tests, 175 assertions)
```

Figure UT3

To extend the unit testing coverage further there are several possible approaches:

- A more recent version of Laravel could be used to test more DOM behaviours
- The controller code could be broken down further to make individual sections more testable, by writing the code such that the behaviour is separated from the data.
- A second copy of the code could be set up on a server with its own database to allow for testing that will read or write to the database.

Data Models

The design and development of data models was an important part when the team began creating a responsive and adaptive hierarchical platform. (Bagui and Earp, 2012, p.5) The data models shown in this section will review how the evolution of the early stages of the platform grew, by reviewing initial entities and relationships. These models adopted Entity Relationship models so that an understanding of the system's structure and relation may be built before referring to how data would be stored. For details on Data Models referring to the post-developmental stage and the implementation of the database and storing data, can be found within the 'SQL' section. Not only did designing models like Entity Relationship Diagrams (ERD) improve earlier stages of development, but they also provided a solid cornerstone in which the platform could develop from. (Sumathi and Esakkirajan, 2007, p.31) These early conceptual designs reviewed real-world entities, and what the platform needed to offer as a minimal viable product, and when completed, what additional updates could be made in future sprints. (Silverston, 2001, pp.9-10; Waldock, 2015) This section provides clarity to earlier stages of development and how the team designed and built the platform using data models.

Understanding the design of the platform involved several stages, before designing an ERD that would act as a scaffold. First developers had to calculate what users/entities would interact with the platform. (Bagui and Earp, 2012, p.25) By creating an ERD the developers worked on how users would access and use the site. Two ERD were created to help the development process. The first ERD reviews how entities such as employees and bowsers interact with one another. (Bagui and Earp, 2012, p.61; Sumathi and Esakkirajan, 2007, p.32)

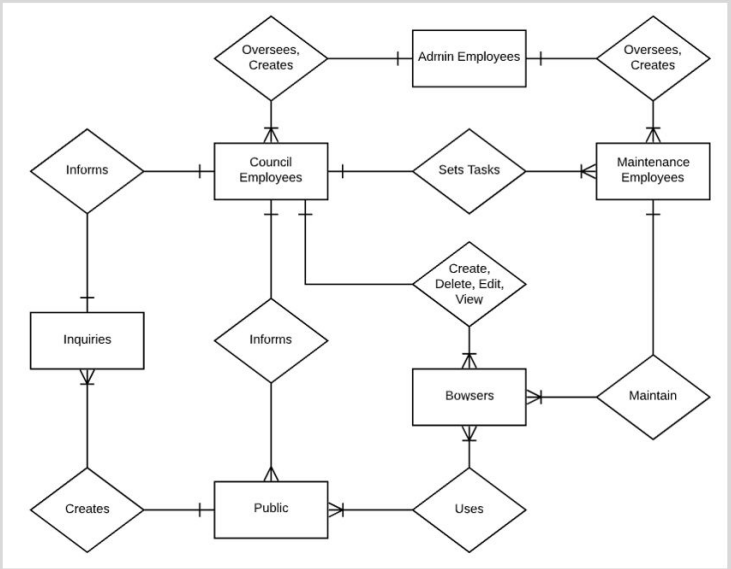


Diagram DM1: An Entity Relationship Diagram showing the relationship between the core five users

As shown in Diagram DM1 there are four human entities that will interact with the platform, this is in addition to the final fifth main entity being a bowser. From this diagram, developers were able to see each entity's relationship with bowsers and calculate how this would transpose into a working environment. For example, Diagram DM1 shows that the relationship between the public and bowser entities is utility, whereas the maintenance worker's relationships use assigned tasks to maintain bowsers, the council both inform and are informed by the public in addition to setting maintenance tasks and creating the bowsers on the platform. The most unique entity with no direct relationship with the bowser entity is the administrator users, these users oversee and create, council and maintenance users. By using this data, a second ERD was developed that helped developers design how pages would interact with the five core entities shown in Diagram DM1.

Diagram DM2 uses the following colour key for entities:

Colour Key for Diagram DM2	
	Council Employee Page
	Public Page
	Administrator Page
	Maintenance and Council Page
	Other non-page entities
	All Employee Page

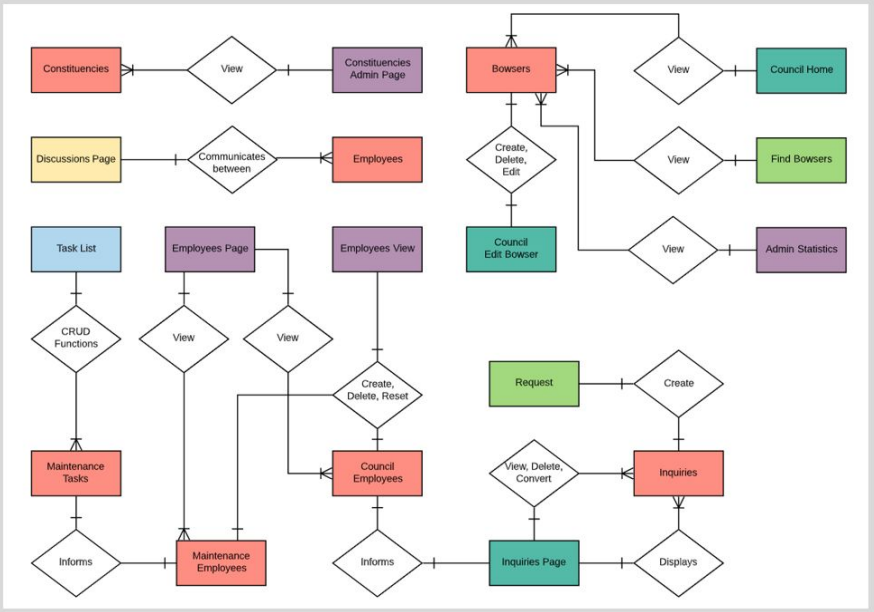


Diagram DM2: An Entity Relationship Diagram showing the relationship between pages and other entities

By using Diagram DM2, developers were able to build each page to its main entities, as well as what actions would be performed on these pages subject to the type of user accessing it. Some models in Diagram DM2 consisted of Binary Relationships whilst others consisted of a much higher degree of complexity. (Sumathi and Esakkirajan, 2007, p.40) For example, the ‘Employees View’ page ‘Creates, Deletes and Resets’ a single council or maintenance employee, whilst its sibling page ‘Employees’ gives an overview.

During early platform development, these designs reflected how the finished product would function. Inspired by this design and using the real-world entities as a mould, the agile team decided to begin working with PHP classes using a Laravel framework. The Laravel framework meant that the platform could now adopt a Model-View-Controller (MVC) design pattern, where work could be broken down into categories in addition to allowing for more intrusive unit testing. (Bourne, 1992, p.175) The MVC design pattern [Diagram DM3] allowed the team to work on the platform’s pages simultaneously, without causing unexpected collisions or adding additional redundant/repeated code. (Stadler et al, 2019) Furthermore, separating out the core components of the project meant that the user interface and business rules remained separate, this increased security by having business rules checked on the server side. (Tahaghoghi and Williams, 2007, p.487) Additionally, this meant that if a business rule changed in the future, a single document could be updated rather than the entire platform, this further reinforces the benefits of reducing code redundancy and making a flexible data model structure.

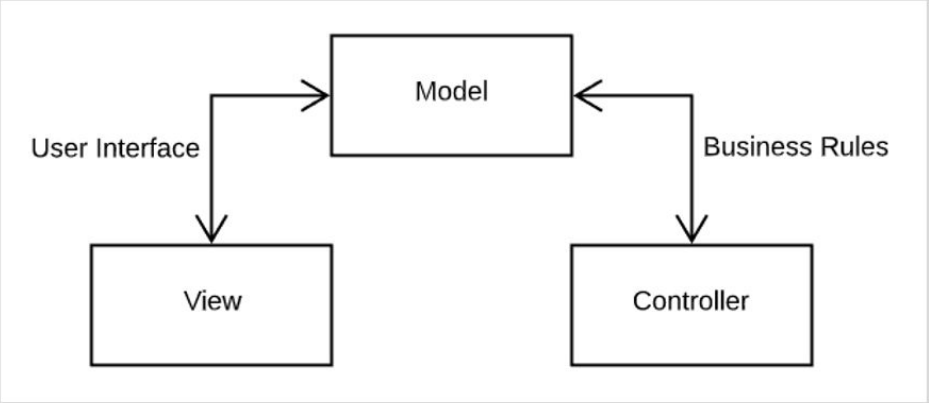


Diagram DM3: A diagram showing Model, View and Controller relation to the project

By implementing MVC patterns the project could now distribute scalable work effectively. This effectiveness was further reinforced by using Laravel's PHP class interactions. By adopting classes in website development meant that the team could reflect and develop the entities shown in Diagram DM2, in addition to increasing the projects flexibility by allowing developers to create a class featuring a set of attributes. (Laravel, 2020a) This scalable object creation was mostly used when interacting with tables (in the 'httpdocs/Blog/app/models' directory). As demonstrated in Diagram DM4, these objects can interact with a database table via the Laravel framework, meaning that several pages can use the same model object for similar SQL requests. By implementing this, the business rules can have a universal effect on the platform, this is in addition to added security measures that ensure that the correct type of data is being uploaded to the database. This process further moved the platform closer to developing the entities at the cornerstone of the platform.

```php
use Illuminate\Database\Eloquent\Model;     // import the model class

class TblCouncilMemberInformationModel extends Model { //define class and inherit from the imported


    //define the table being used
    protected $table = 'CouncilMemberInformation';


    //get the constituency IDs a council member is responsible for
    public function getCouncilMemberConstituency($userID) {
        $constituencyID = TblCouncilMemberInformationModel::where('UserID', '=', $userID)
            ->get(array("ConstituencyID"));

        return $constituencyID;
    }
}
```

Diagram DM4: Code from the TblCouncilMemberInformationModel class

Reflecting on the design shown in Diagram DM2, the structure of the implemented/inspired data models and MVC pattern combined to create functional pages in three core stages [Diagram DM5]. First the server interacts with a Controller class object. The Controller's main purpose in initiating the page is to provide relevant data for the user interface to display. (Bourne, 1992, pp.175-176) This process can additionally implement security checks to ensure that the user is accessing an authorized page. (Tahaghoghi and Williams, 2007, p.487) The data is then sent to the view objects in the 'httpdocs/Blog/app/views' directory, where the data is inserted onto the page to be displayed. By using Laravel's framework, the view files commonly inherit a layout file in the 'httpdocs/Blog/app/views/layouts' directory, these layouts allow the platform to have consistent design patterns/themes as well as reducing copies of the code. If data needs to be obtained after the page has loaded (through ajax or form submission), the file uses the 'routes.php' file. (Laravel, 2020b)

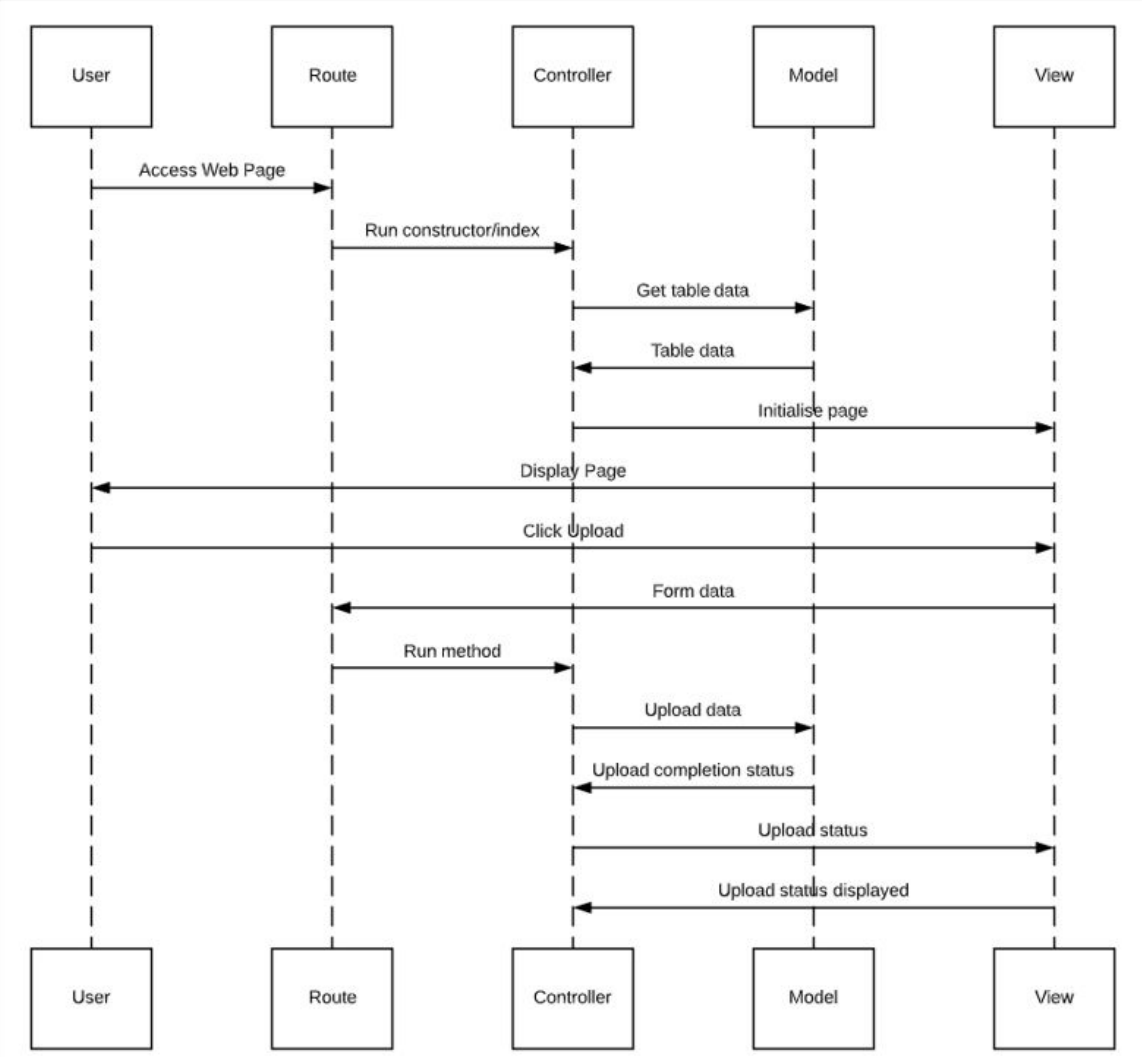


Diagram DM5: Sequence Diagram showing page load sequence with MVC

Being an example of Object-Oriented Programming, the platform files can be reviewed as UML or Sequence Diagrams. The Sequence Diagram DM5, demonstrates how a web page is loaded using the Laravel framework and how each layer communicates, allowing for unit testing and separated development. The UML Diagram DM6 denotes their associations and relationships, as each file relies upon, inherits or extends a developed class object. All page layouts extend from a ‘default.blade’ file which provides the default mobile and desktop structures. (Lethbridge and Laganière, 2005, pp.286-287; Bruegge, 2004, pp.32-33)

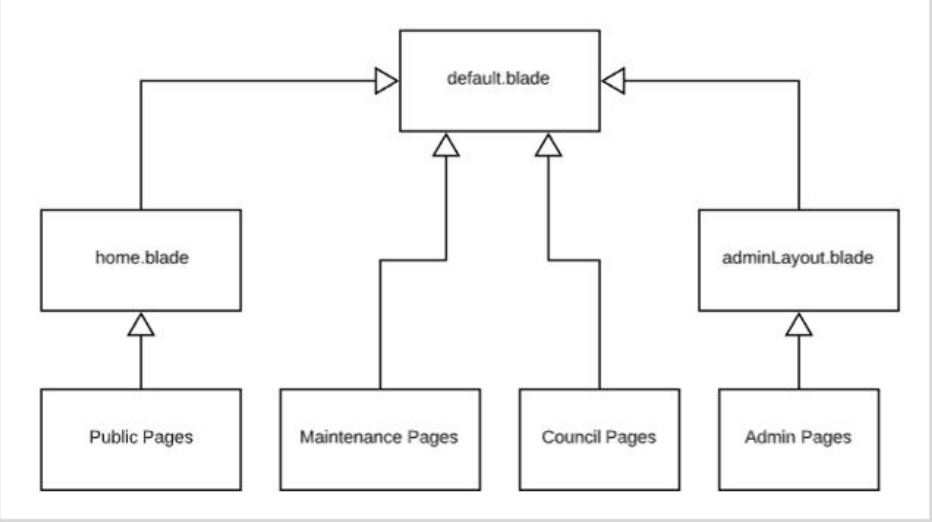


Diagram DM6: UML Diagram of page layout inheritance

In summary, the Data Models discussed helped to move the project from early developmental stages to a working implementation. The entities were inspired to best reflect real-world users and objects that would have a relationship with the project’s platform. Further implementation from these basic ERDs resulted in the implementation and development of Sequence and UML diagrams, which further represented how the project functions and the relationships and abilities each entity has.

Design, Layout and Implementation

Completing the design and layout of the application is important for the implementation of the website. This is due to the fact that the final product is going to be used by different people and the accessibility and ease of use is necessary to reduce confusion. The main objectives of this section consist of the uniformity and accessibility of the website.

Implementation:

Due to Laravel's formatting, a blade file is used to construct the page. From this the construction of the pages are possible. Using sections assists the design and development of the pages. From these, certain sections are used for most pages. This is visible in the skeleton below. Mobile checking is done on every page for the ease of use in mobile format. Additionally, there are both the routes folder to make the URLs, and the 'filter.php' is for the definition of the user authentication.

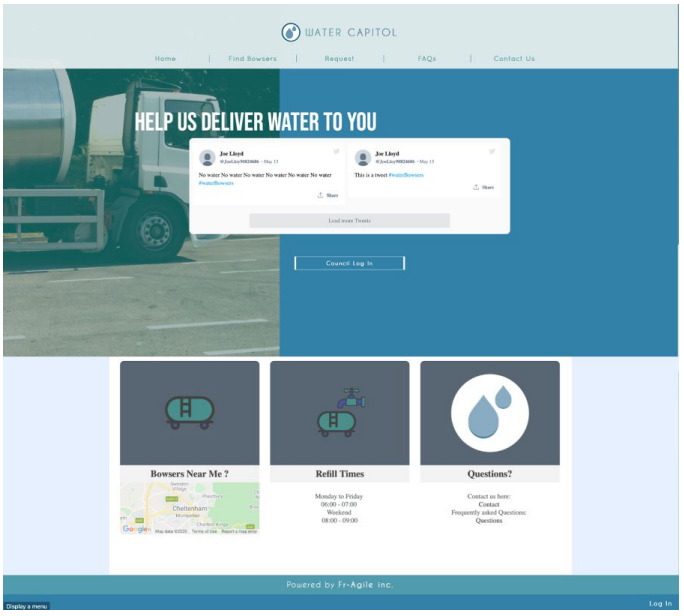
```
<?php
    // Check if this page is opened on a mobile
    require "Resources/PHP/mobileCheck.php";
?>

<!-- scrtipts used -->
@section('pageJS')
@stop


<!-- slide shows -->
@section('upperbackground')
@stop

<!-- main upper section -->
@section('uppercontents')

@stop

<!-- main contens of the page -->
@section('maincontents')
@stop

<!-- page styling and php for mobile -->
@section('pageCSS')
@stop
```

Figure DLI1

Public:

Homepage:

The home page introduces the website to incoming users, the main objective of this is to ensure that the page is clearly understood and that the user knows that this is the bowser's website. By having clear images and navigation names, clarifies that this is the homepage of the Water Capitol service. Additionally, a Twitter API is acquired to display tweets in regard to the current issues and news related to water bowsers. From the sprint retrospectives, the stakeholders wanted the login button moved to the bottom and this is visible with the page's contents in Figure DLI2. This allows for the construction of the main contents, displaying images and the Twitter API.



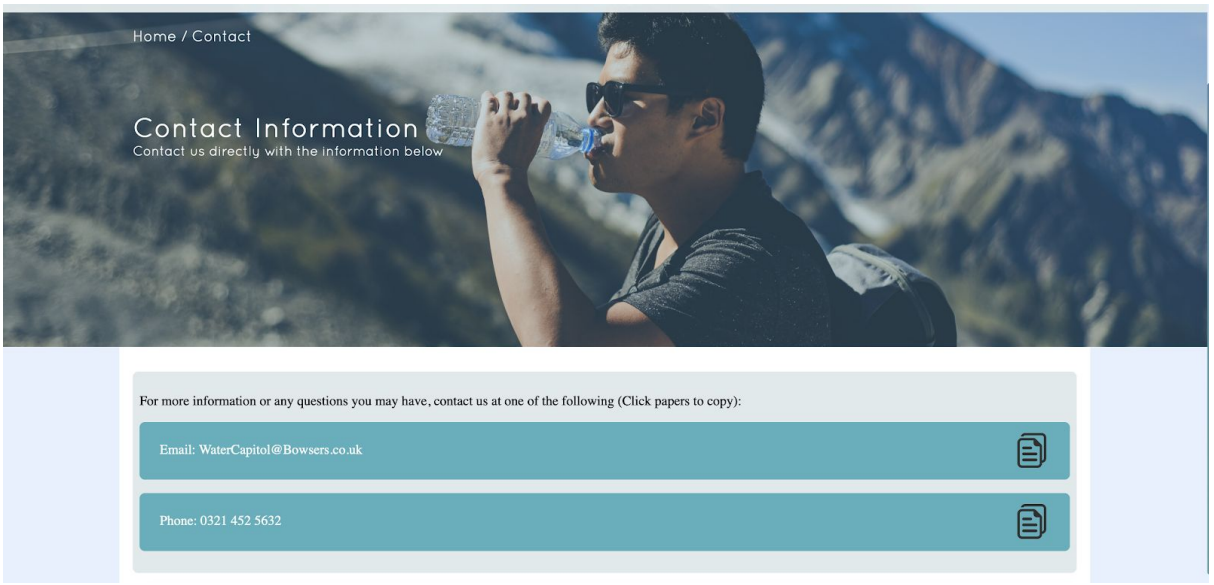
```
<!-- scrtipts used -->
@section('pageJS')
        <script src="Resources/Pages/Home/homeAnimate.js" type="text/javascript"></script>
        <script type="text/javascript"> $(document).ready(function(){window.onload = mainMap()});</script>
        <script type="text/javascript"> window.onload = main();</script>
@stop


<!-- slide shows -->
@section('upperbackground')
    @include('includes.slideShow')
    @yield('slideShow')
@stop

<!-- main upper section -->
@section('uppercontents')
<div class="slideShowTitleSector" border="0">
    <titleSector>
        <p>Help Us Deliver Water to You</p>
    </titleSector>

    <script src="https://apps.elfsight.com/p/platform.js" defer></script>
    <div class="elfsight-app-bdfc3bce-258d-444d-805a-d9fd2eb41eaa"></div>

    <button class="button logInButton">
        <p>Council Log In</p>
    </button>
</div>
@stop
@section('maincontents')
<form name="homeForm">
          <div class="bg">
              <div class="content">
<!--                Bowsers near me link -->
                  <div class="col">
                      <div class="at">
                          <img src="Resources/Images/waterMap.png">
                      </div>
                      <div class="it" >
                          <h2> <a href="https://ct5038group2-ct5038.uogs.co.uk/bowserNearMe" class = "x">Bowsers Near Me ? </a> </h2>
                      </div>
                      <div class="ba">
                          @include('includes.map')
                      </div>

                  </div>
<!--                 Refill times information -->
                  <div class="col">
                      <div class="at">
                          <img src="Resources/Images/waterMap.gif">
                      </div>

                      <div class="it">
                          <h2>Refill Times</h2>
                      </div>
                      <div class="ba">
                          <p class="details"> Monday to Friday
                              <br> 06:00 - 07:00
                              <br> Weekend
                              <br> 08:00 - 09:00
                          </p>
                      </div>
                  </div>
<!--                FAQ page link -->
                  <div class="col">
                      <div class="at">
                          <img src="Resources/Images/Logo/Medium/logo_basic_white_circle.png">
                      </div>
                      <div class="it">
                          <h2>Questions?</h2>
                      </div>
                      <div class="ba">
                          <p class="details">Contact us here:
                              <br>
                                  <a href="https://ct5038group2-ct5038.uogs.co.uk/contact" class="x">Contact</a>
                              <br>Frequently asked Questions:
                              <br>
                                  <a href="https://ct5038group2-ct5038.uogs.co.uk/faq" class="x">Questions</a>
                              <br>
                          </p>
                      </div>
                  </div>
              </div>
          </div>
</form>
```

Figure DLI2

Contact us:

This page is constructed for the public to contact the council directly. This is for any urgent use of the public in cases where the request form is not fast enough to respond. This

page also allows for the user to copy the information needed by pressing the necessary form of communication's clipboard on the right-hand side of each row. The one special function implemented on this page is the copying function.



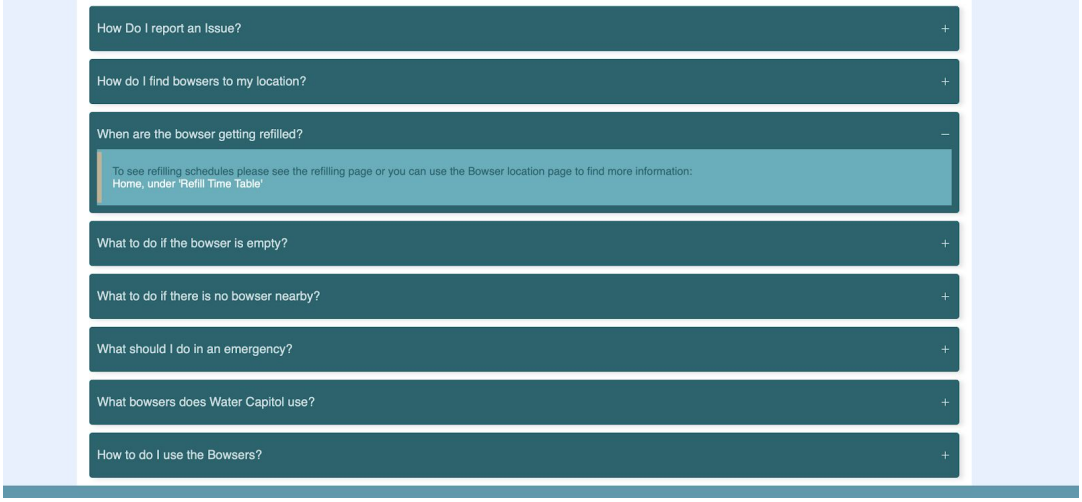

```
<!-- Get JS -->
@section('pageJS')
    <script>
        function copyDetails(text) { //function to copy contents to clipboard
            var field = document.getElementById("copyField"); //create an input (must be copied from)
            console.log(text);
            field.value = text; //fill in the dummy value with text entered
            console.log(field);
            field.select(); //select it
            field.setSelectionRange(0, 99999);
            document.execCommand("copy"); //copy it
            alert("COPIED");
        }
    </script>
@stop
```

```
<!-- Show the content for this page -->
@section('maincontents')
<form class="f">
    <input id="copyField"/>
    <div class = "mainContentsClass">
        <br>
        <p id="content">For more information or any questions you may have, contact us at one of the following (Click papers to copy): </p>
        <div class="Container">
            <p class="infoText" id="email">Email: WaterCapitol@Bowsers.co.uk</p> <input type="button" class="buttons"
            onclick=copyDetails("WaterCapitol@Bowsers.co.uk")>
        </div>
        <div class="Container">
            <p class="infoText" id="phone">Phone: 0321 452 5632</p> <input type="button" class="buttons" onclick=copyDetails("03214525632")>
        </div>
        <br>
    </div>
</form>
@stop
```

Figure DLI3

Frequently Asked Questions:

This is the page for frequently asked questions (FAQ). The reason for putting this in was so that it would lessen the workload through emails or the public requests page. This page has common questions and is animated so when a user clicks on a block of text it drops the text so it looks cleaner and simpler so people can clearly see which question relates to their issue.




```
<div class="accordion-item" id="question6">
    <a class="accordion-link" href="#question6">
        What should I do in an emergency?
        <ion-icon class ="ion ion-md-add" name="add"></ion-icon>
        <ion-icon class ="ion ion-md-remove" name="remove"></ion-icon>
    </a>
    <div class="answer">
        <p>
            The worse thing to do is panic.
            <br> If you have an emergency are in immediate danger please call 999.
            <br> Alternatively call 101 or 111 to ask for advice.
            <br> Any urgent bowser issues please call us on 0321 452 5632.
            <br> Or 24/7 number if out of 9 - 17 hours is 0324 256 2463.
        </p>
    </div>
</div>
```

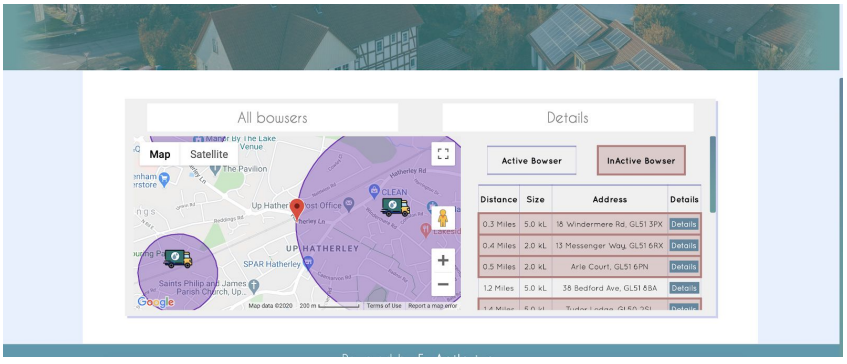
```css
/*link styling*/
.accordion-link{
    font-size: 1.6rem;
    color: rgba(255,255,255,.8);
    text-decoration: none;
    background-color: #12646f;
    width: 100%;
    display:flex;
    align-items: center;
    justify-content: space-between;
    padding: 1rem 0;
}


.accordion-link i{
    color: #e7d5ff;
    padding: .5rem;
}

.accordion-link .ion-md-remove{
    display: none;
}
```

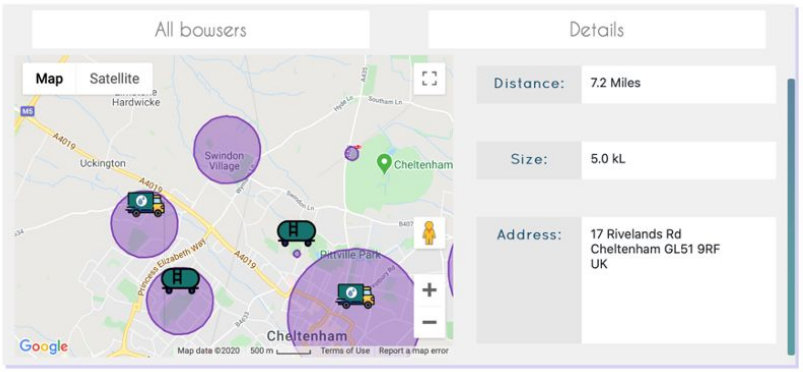
```css
/*answers*/
.answer{
    max-height: 0;
    overflow: hidden;
    position: relative;
    background-color: #53b0bd;
    transition: max-height 400ms;
}

.answer::before{
    content: "";
    position: absolute;
    width: .6rem;
    height: 90%;
    background-color: #c2b394;
    top: 50%;
    left: 0;
    transform: translateY(-50%);
}

.answer p{
    font-size: 1.4rem;
    color: #12646f;
    padding: 2rem;
}

/*accordian*/
.accordion-item:target .answer{
    max-height: 25rem;
}

.accordion-item:target .accordion-link .ion-md-add{
    display: none;
}

.accordion-item:target .accordion-link .ion-md-remove{
    display: block;
}
```

Figure DLI4

Find Bowsers:

This page allows for the display of the bowsers on a map for the ease of use of the user. This is done using the user's current location and showing them the bowsers nearest to them with Google's map API. This adds additional ease of use through the possibility of functions such as scrolling in or out of a general location for a larger view of the area. Through the second team retrospective the stakeholders requested a 50/50 display of both the map of the bowser locations and the bowser's unique information and details for the accessibility of the user. Additionally, the active and inactive bowsers can be visually represented through a colour system. From this the details of individual bowsers can be displayed through a press of the details button or the bowser itself. The majority of the construction of this page was done through JavaScript for implementing the map and its details.






```
// Update bowsers
function update() {
    // update bowser list //
    if(typeof userLocation !== 'undefined') {
        var formData = {
            'lat': userLocation.lat,
            'lng': userLocation.lng};
    } else {
        var formData = {};}

    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="_token"]').attr('content')}
    });
    // Send formData to controller
    $.ajax({
        type        : 'POST',
        url         : 'bowserNearMe_update',
        data        : formData,
        dataType    : 'json',
        encode      : true
    }).done(function(data) {
        // Was data received?
        if(data.success) {
            // update bowsers list
            rowIndex = 1;
            data.bowserList.forEach(updateRow);
            circlesA.forEach(cirle => cirle.setMap(null));
            circlesB.forEach(cirle => cirle.setMap(null));
            circlesA = [];
            circlesB = [];
            markersA.forEach(marker => marker.setMap(null));
            markersB.forEach(marker => marker.setMap(null));
            markersA = [];
            markersB = [];
            data.bowsers.forEach(updateMap);
            // remove excess rows
            while( bowsersTable.rows[rowIndex] !== undefined ) {
                bowsersTable.deleteRow(rowIndex);
            }
        } else {
            failAlert(data.error);}
    });
}
setTimeout(update, 3000); // load bowsers after page load
setInterval(update, 30000); // Update list of bowsers every 30s
```

```
// update row in table
function updateRow(row) {

    // clear row if exists / add row if it does not exist
    if( bowsersTable.rows[rowIndex] === undefined ) {
        bowsersTable.insertRow(rowIndex);
    } else {
        bowsersTable.rows[rowIndex].classList.remove('bowserRow');
    }
    // tag and style row
    let bowserRow = bowsersTable.rows[rowIndex];
    bowserRow.classList.add("bowserRow");
    bowserRow.setAttribute("id", "bowser" + row.bowserID);
    bowserRow.setAttribute('data-bowserID', row.bowserID);
    bowserRow.setAttribute('data-distance', row.distance);
    bowserRow.setAttribute('data-size', row.size);
    bowserRow.setAttribute('data-address', row.address);
    // style based on if assigned to user
    if( row.inactive ) {
        bowserRow.classList.add("inactive");
    }
    var addressParts = row.address.split(', ');
    var postcodeParts = addressParts[addressParts.length - 2].split(' ');

    // set row contents
    bowserRow.innerHTML =
        '<td width="15%">'+
            '<p>' + row.distance + '</p>'+
        '</td>'+
        '<td width="15%">'+
            '<p>' + row.size + '</p>'+
        '</td>'+
        '<td width="55%">'+
            '<p>' + addressParts[0] + ', ' + postcodeParts[1] + ' ' + postcodeParts[2] + '</p>'+
        '</td>'+
        '<td width="15%" class="details" onclick="details(' + row.bowserID + ')">'+
            '<p>Details</p>'+
        '</td>';

    rowIndex ++;
}
```

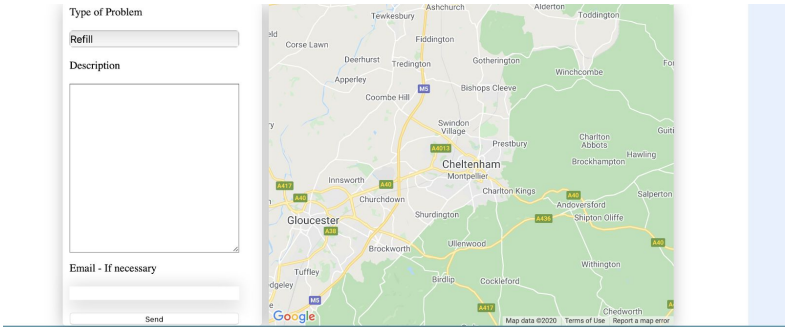
```
// Update bowsers on map
function updateMap(bowser) {
    bowserlng       = bowser.longitude;
    bowserlat       = bowser.latitude;
    bowserID        = bowser.BowserID;
    bowserStatus    = bowser.bowserStatus;
    bowserAddress   = bowser.bowserAddress;
    bowserRadius    = bowser.size;

    var myLatlng = new google.maps.LatLng(bowserlat, bowserlng);

    var url = "";
    if(bowserStatus == 0) {
        url = "Resources/Images/waterMap.png";
    } else if (bowserStatus == 1) {
        url = "Resources/Images/tools.png";
    } else if (bowserStatus == 2) {
        url = "Resources/Images/waterMap.gif";
    } else if (bowserStatus == 3) {
        url = "Resources/Images/waterMapDeploy.png";
    }
    var icon = {
        url: url,
        scaledSize: new google.maps.Size(50, 50), // scaled size
    };
    // add circle and marker to mapA
    var markerA = new google.maps.Marker({
        position: myLatlng,
        title: "Bowser ID: " + bowserID + " - " + bowserAddress,
        map: mapA,
        icon: icon
    });
    markersA[bowserID] = markerA;
    var circleA = new google.maps.Circle({
        map: mapA,
        radius: bowser.size/10,    // 10k Liter = 1km
        fillColor: '#AA0000',
        strokeColor: '#6600AA',
        strokeOpacity: 0.8,
        strokeWeight: 2,
        fillColor: '#6600AA',
        fillOpacity: 0.35,
    });
    circleA.bindTo('center', markerA, 'position');
    circlesA[bowserID] = circleA;
    // add circle and marker to mapA
    var markerB = new google.maps.Marker({
        position: myLatlng,
        title: "Bowser ID: " + bowserID + " - " + bowserAddress,
        map: mapB,
        icon: icon
    });
```

```
    markersB[bowserID] = markerB;
    var circleB = new google.maps.Circle({
        map: mapB,
        radius: bowser.size/10,    // 10k Liter = 1km
        fillColor: '#AA0000',
        strokeColor: '#6600AA',
        strokeOpacity: 0.8,
        strokeWeight: 2,
        fillColor: '#6600AA',
        fillOpacity: 0.35,
    });
    circleB.bindTo('center', markerB, 'position');
    circlesB[bowserID] = circleB;
    // Bind to marker
    google.maps.event.addListener(markerA, 'click', function() {
        $(".mainContentsSwipeTable").css("left", "-100%");
        $(".mapBackground").css("opacity", "0");
        $(".mapBackground").css("pointer-events", "none");
        $(".statsDiv").css("opacity", "1");
        $(".statsDiv").css("pointer-events", "auto")
        $(".bowserStatsButton p").addClass("focusedTabButton")
        $(".mapCoverageButton p").removeClass("focusedTabButton");

        details(bowserID);
    });
    google.maps.event.addListener(markerB, 'click', function() {
        $(".mainContentsSwipeTable").css("left", "-100%");
        $(".mapBackground").css("opacity", "0");
        $(".mapBackground").css("pointer-events", "none");
        $(".statsDiv").css("opacity", "1");
        $(".statsDiv").css("pointer-events", "auto")
        $(".bowserStatsButton p").addClass("focusedTabButton")
        $(".mapCoverageButton p").removeClass("focusedTabButton");
    });
}

// Display bowser details
function details(bowserID) {
    // update info
    bowserRow = document.getElementById("bowser" + bowserID);
    document.getElementById("bowser-distance").innerHTML   = bowserRow.getAttribute('data-distance');
    document.getElementById("bowser-size").innerHTML       = bowserRow.getAttribute('data-size');
    document.getElementById("bowser-address").innerHTML    = bowserRow.getAttribute('data-address').replace(/, /g, "</br>");
    // update map
    mapB.panTo( markersB[ bowserID ].getPosition() );
    markersB.forEach(marker => marker.setVisible(false));
    markersB[ bowserID ].setVisible(true);
    circlesB.forEach(circle => circle.setOptions({fillOpacity:0, strokeOpacity:0}));
    circlesB[ bowserID ].setOptions({fillOpacity:0.35, strokeOpacity:0.8});
    // move to details pane
    $(".mainContentsSwipeTable").css("left", "-100%");
}
```

Figure DLI5

Public Request:

For problem handling, the public can send in requests to resolve problems and issues within their local areas. An example of a request can be a faulty tap, an empty bowser or even a newly flooded area. These can be chosen, and the user can then select an area on the map for where the problem has occurred, to send out a request to the council. This is then refined in the council side of the application. The construction of this page is done to allow for assistance from the public when there are possible unknown errors and problems in local areas. Through the implementation of the request page, the map is used to gather the current location of the problem and then is sent with the additional information to the public requests table through the controller.




```
function mainRequest() {
    // Select the Map Div and focus on this location
   var mainMap = new google.maps.Map(document.getElementById('map'), {
        center: {lat: 51.8994, lng: -2.0783},
        zoom: 11,
        disableDefaultUI: true,
        mapTypeId: google.maps.MapTypeId.ROADMAP
   });

    // Create marker
        var marker = new google.maps.Marker({
            map: mainMap
        });

    // Set event listener on the google maps div
        google.maps.event.addListener(mainMap, 'click', function(e){

                // Remove the previous marker
                marker.setMap(null);

                // Create a new marker with the new position
                marker = new google.maps.Marker({
                    position: e.latLng,
                    map: mainMap
                });
                mainMap.setCenter(e.latLng);

            long = e.latLng.lng();
            lat = e.latLng.lat();
        })
}
```

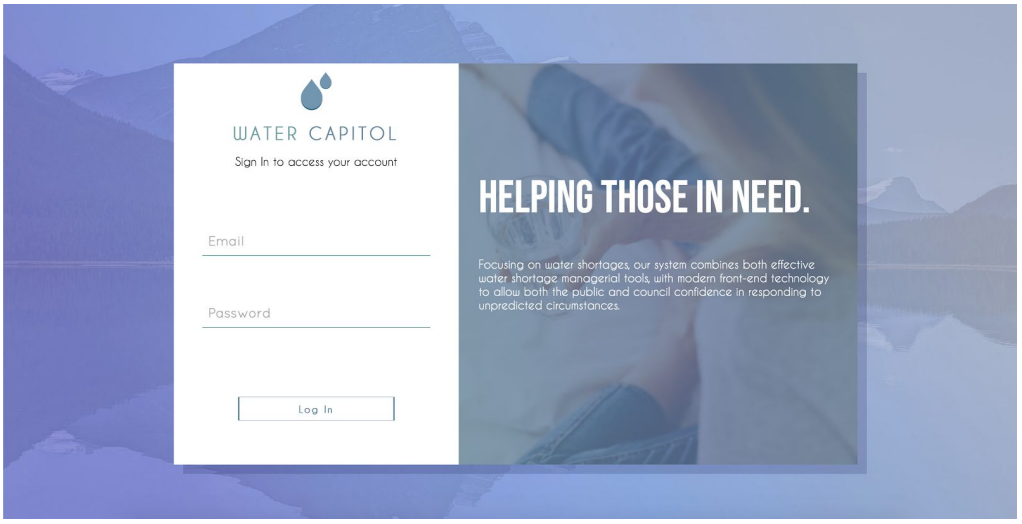
```
function newRequest() {

    // data for the inquiry
    var formData = {
        'drop_down': $('select option:selected').val(),
        'description': $('#desc').val(),
        'emailAdder': $('#eml').val(),
        'lat':lat,
        'long':long
    };


    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="_token"]').attr('content')
        }
    });

    // Send formData to class.functions.php
    $.ajax({
        type        : 'POST',
        url         : 'request_submit',
        data        : formData,
        dataType    : 'json',
        encode      : true
    }).done(function(data) {
        console.log(data)
        if(data.success) {
            // alert user that account has been made and return to homepage
            alert( "Added" );
            //window.location.href = "/request";
        } else {
            alert(data.error);
        }
    });
}
```

```php
class RequestController extends BaseController {

    // create page
    public function index()
    {

        return View::Make('/pages/request');
    }


    // add request to database
    public function submit()
    {
        $drop_down      = isset($_POST[ 'drop_down' ]);
        $description    = isset($_POST[ 'description' ]);
        $emailAdder     = isset($_POST[ 'emailAdder' ]);

        // Return if known and message
        $result = array(
           'success' => false,
           'error' => 'Unknown error: try again'
        );

        // validate inputs - as lat and long
        if(isset($_POST['drop_down'])){

            $problem = array('type' => $_POST['drop_down']);
            echo( $drop_down );

            if(isset($_POST['description'])){
                $problem['description'] = $_POST['description'];
            }
            if(isset($_POST['emailAdder'])){
                $problem['email'] = $_POST['emailAdder'];
            }
            if(isset($_POST['lat'])){ // add lat
                $problem['lat'] = $_POST['lat'];
            }
            if(isset($_POST['long'])){ // add long
                $problem['long'] = $_POST['long'];
            }

            // update report to database
            if( tblReportInfo::addReport( $problem ) ) {
                $result['success'] = true;
            } else {
                $result['error'] = "Could not submit, reload page";
            }
        }

        //return the response
        return Response::json( $result );
    }
}
```

Figure DLI6

Employee Login:

This simple designed page is the login portal for all employees accessing the site. This is done to add security to the application and for the ease of logging in. This starts a new session for the current user logging in and allows for them to edit bowsers and access their general areas of management or work. DLI7 shows the login function checking if a user has logged in correctly.

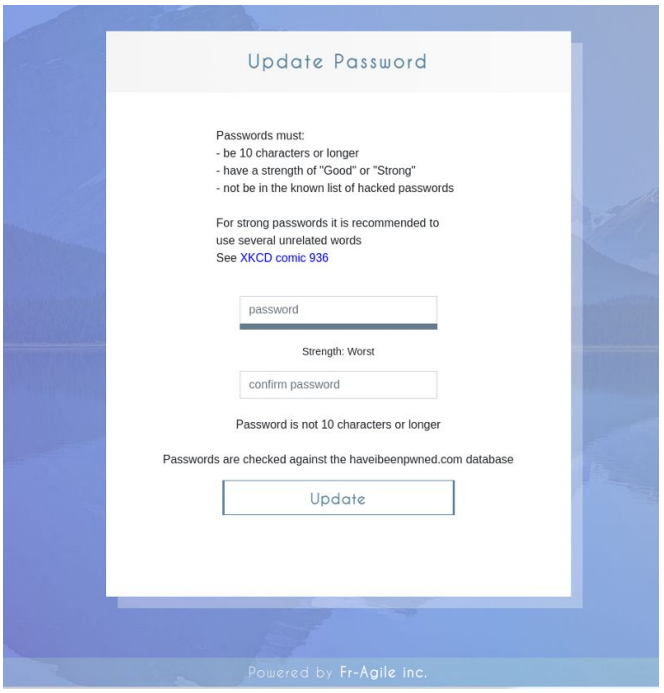


```php
use Response;       // for ajax response reply

class loginController extends \BaseController { //class extends baseController

    // create page
    public function index() {
        return View::Make('/pages/login');
    }

    // login request
    public function submit() {

        //attempt to authenticate the user and log them in
        $success = Auth::attempt(array('email' => $_POST[ 'email' ], 'password' => $_POST[ 'password' ]), true);


        if( $success ) {
            // success
            $result = array(
               'success' => true,
               'type' => "council"
            );

        } else {
            // incorrect login
            $result = array(
               'success' => false,
               'error' => "Wrong email or password"
            );
        }

        return Response::json( $result );
    }

}
```

Figure DLI7

Password changing:

When creating a new updated password, a page appears through an admin email. This tells the user what the requirements are for a new password alongside giving the user a strength result. Given the factor of security, a government run agency requires the password checker to not allow the user to enter a weak password. Several functions are required for checking the password's strength, submission and it's 'hackability'. This uses a database of passwords for the checking of its security. (Hunt, 2020)



```php
// return password strength
public function strength(){
    $password = $_POST[ 'password' ];
    if ( empty($password) ) {
        // If empty display placeholder text
        $result = array(
           'known' => true,
           'message' => 'Passwords are checked against the haveibeenpwned.com database');
    } else {
        // check against known hacked passwords
        $count = ResetController::hacked( $password );
        // Create UI message
        $message = '';
        if ($count > 0) {
            $message = 'This password has been found '.$count.' times in hacked websites';
        } else {
            $message = 'There are no known occurrences of this password being hacked';
        }
        // Return if known and message
        $result = array(
           'known' => $count > 0,
           'message' => $message
        );
    }
    return Response::json( $result );
}
```

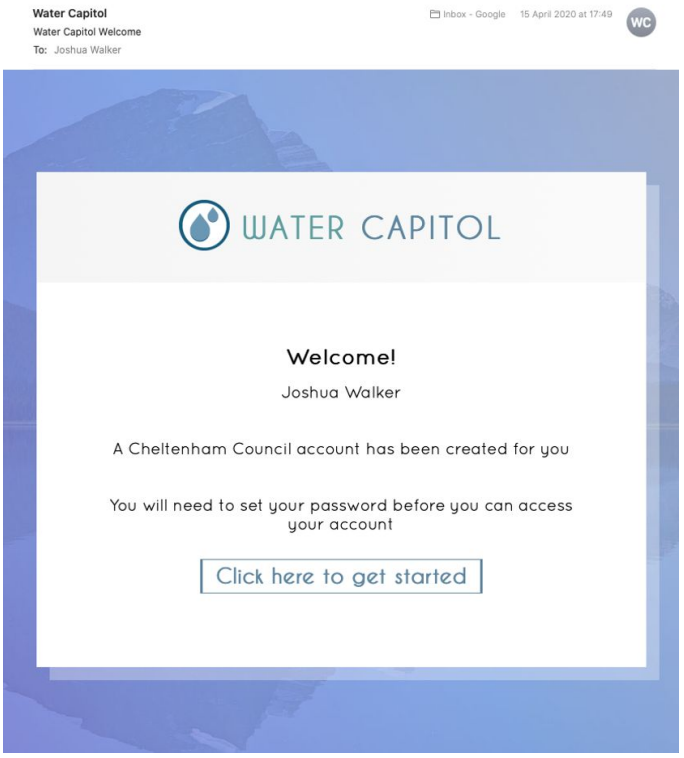
```php
// submit new password
public function submit(){
    $password   = $_POST[ 'password' ];
    $confirm    = $_POST[ 'confirm' ];
    $complexity = $_POST[ 'complexity' ];
    $token      = $_POST[ 'token' ];
    // Return if known and message
    $result = array(
       'success' => false,
       'error' => 'Unknown error: Request new email and try again'
    );
    // Ensure that both inputs are valid and match
    if( !empty($password) && !empty($confirm) ) {
        if( $password == $confirm ) {
            // Check that password has atleast 10 characters
            if(strlen($password) >= 10) {
                // Check that the password meets the complexity requirement
                if($complexity >= 3) {
                    // Check password is not the XKCD example "correcthorsebatterystaple"
                    if( $password != "correcthorsebatterystaple" ) {
                        // check that recovery token if not missing
                        if( $token != "" ) {
                            // check against known hacked passwords
                            $count = ResetController::hacked( $password );
                            if( $count == 0 ) {
                                // update password for user with token
                                if( TblUserInformationModel::setPasswordByToken( $password, $token ) ) {
                                    $result['success'] = true;
                                } else {
                                    $result['error'] = "Recovery token does not match: Request new email and try again";
                                }
                            } else {
                                $result['error'] = "This password has been found ".$count." times in hacked websites: Try another";
                            }
                        } else {
                            $result['error'] = "Recovery token missing: Request new email and try again";
                        }
                    } else {
                        $result['error'] = "This is the xkcd example password: Try another";
                    }
                } else {
                    $result['error'] = "Password does not meet minimum complexity requirement of 'Good': Try another";
                }
            } else {
                $result['error'] = "Password is not 10 characters or longer: Try another";
            }
        } else {
            $result['error'] = "Passwords do not match: Try again";
        }
    } else {
        $result['error'] = "Password fields cannot be empty: Try again";
    }
    return Response::json( $result );
}
```

```php
// get number of times password has been hacked
private static function hacked( $password ) {

    // hash passowrd to check against known hacked passwords
    $SHAhash = sha1($password);

    // Send first 5 character only of hash to get back list of possible matching hashes
    $page = file_get_contents("https://api.pwnedpasswords.com/range/".substr(strval($SHAhash), 0, 5));

    // Convert page into two arrays of all returned data and just hashes
    $returnedData = explode(PHP_EOL, $page);
    $hashList = array_map(function($string) {
        return substr($string, 0, 35);
    }, $returnedData);

    // Check for maching hash in list
    $needle = strtoupper(substr(strval($SHAhash), 5));
    $found = array_search($needle, $hashList);

    // State if found and how many times
    $known = ($found !== false);
    if($found) {
        $count = substr($returnedData[$found], 36);
    } else {
        $count = 0;
    }

    return $count;
}
```

Figure DLI8

Email Response:

An email is sent to the user once a password needs resetting. This email will send the user to the page shown in figure DLI8. This allows for additional security as the only person accessing the page is the user themselves, the administrator can send the link to change the password but cannot change it themselves. The blade for the creation of this email is visible in figure DLI9.



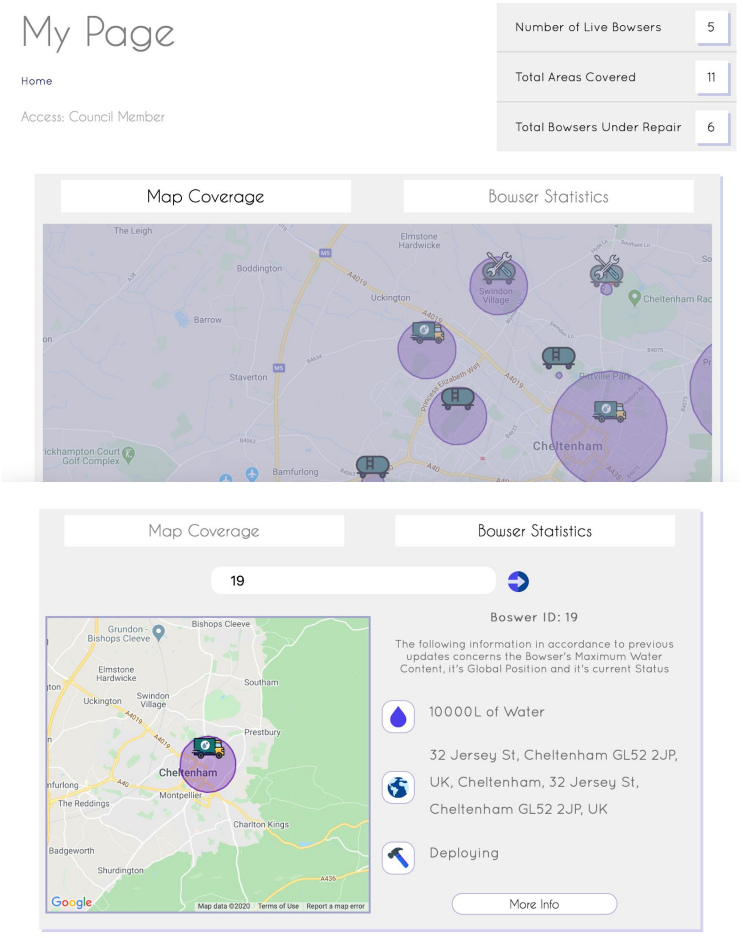

```
<!-- Page content -->
@section('content')
    <p>
        {{$name}}<br/>
        <br/>
        <br/>
        A password reset has been requested for your account<br/>
        <br/>
        <br/>
        You will not be able to access your account until you set a new password<br/>
        <button>
            <a href="https://ct5038group2-ct5038.uogs.co.uk/reset?token={{$token}}">Click here to change your password</a>
        </button>

        <br/>
        If the email does not display properly you can access your account here:<br/>
        <br/>
        https://ct5038group2-ct5038.uogs.co.uk/reset?token={{$token}}<br/>
    </p>
@stop
```

Figure DLI9

Council:

Homepage:

When logging in to the council section of the site the user is able to view all the bowsers in the local area, the specific bowser assistance and a large table of all the live bowsers. This is an easily understandable breakdown of the current progress and status of the bowsers. With the Controller alongside the JavaScript and Blade for the council homepage.



Live Bowsers

Date	ID	Status	Location	Size
28-02-2020	1	Active	56 Folly Ln, Cheltenham GL50 4BY, UK	500
26-02-2020	2	Refill	Tudor Lodge, 17 The Park, Cheltenham GL50 2SL, UK	5000
19-03-2020	3	Refill	6 Buttercross Ln, Prestbury, Cheltenham GL52 5SF, UK	10000
19-03-2020	4	Minimal Repair	Prestbury, Cheltenham GL50 4SH, UK	1000
19-03-2020	5	Aesthetic Refill	17 Rivelands Rd, Cheltenham GL51 9RF, UK	5000
21-03-2020	11	Observation	Arle Court, Hatherley Ln, Cheltenham GL51 6PN, UK	2000
06-05-2020	18	Active	38 Bedford Ave, Cheltenham GL51 8BA, UK	5000
06-05-2020	19	Deploying	32 Jersey St, Cheltenham GL52 2JP, UK	10000
06-05-2020	20	Deploying	21 Grenadier Rd, Cheltenham GL51 0WB, UK	5000
06-05-2020	21	Deploying	18 Windermere Rd, Cheltenham GL51 3PX, UK	5000

```
<?php
    // Get all bowser data
    $active_bowsers = array();
    $repair_bowsers = array();
    $refill_bowsers = array();
    $deploy_bowsers = array();
    $minimal_repair_bowsers = array();
    $aesthetic_repair_bowsers = array();
    $observation_bowsers = array();
    // Format bowser data
    foreach ($information["bowserData"] as $bowser){
        if ($bowser['bowserStatus'] == 0){
            array_push($active_bowsers, $bowser);
        }
        elseif ($bowser['bowserStatus'] == 1){
            array_push($repair_bowsers, $bowser);
        }
        elseif ($bowser['bowserStatus'] == 2){
            array_push($refill_bowsers, $bowser);
        }
        elseif ($bowser['bowserStatus'] == 3){
            array_push($deploy_bowsers, $bowser);
        }
        elseif ($bowser['bowserStatus'] == 4){
            array_push($minimal_repair_bowsers, $bowser);
        }
        elseif ($bowser['bowserStatus'] == 5){
            array_push($aesthetic_repair_bowsers, $bowser);
        }
        elseif ($bowser['bowserStatus'] == 6){
            array_push($observation_bowsers, $bowser);
        }
    }
?>
@section('pageCSS')
        <link rel="stylesheet" type="text/css" href="Resources/Pages/AdminHome/adminhomestyles.css"/>
    <?php
include "Resources/PHP/mobileCheck.php";
if(isMobile()){?>
        <link rel="stylesheet" type="text/css" href="Resources/Pages/AdminHome/adminhomemobilestyles.css"/>
    <?php }?>
@stop
@section('pageJS')
        <script src="Resources/Pages/AdminHome/adminhomeanimate.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.bundle.min.js"></script>
        <script
                src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD5Xc19v5rFKH1cXxyQoLxC_rEESDDMlJU&callback=initMap">
        </script>
<script type="text/javascript">
    $(document).ready(function() {
        // Select the Map Div and focus on this location
        var mainMap = new google.maps.Map(document.getElementById('map'), {
            center: {lat: 51.8994, lng: -2.0783},
            zoom: 8,disableDefaultUI: true, mapTypeId: google.maps.MapTypeId.ROADMAP});
        // Create data for chart
```

```
        data = {
            datasets: [{
                data: [<?php echo sizeof($information["maintenanceTasks"]["complete"]); ?>,
                    <?php echo sizeof($information["maintenanceTasks"]["uncomplete"]); ?>],
                backgroundColor: ['rgba(255, 99, 132, 0.2)','rgba(54, 162, 235, 0.2)'],
                    borderColor: ['rgba(255, 99, 132, 1)', 'rgba(54, 162, 235, 1)'], borderWidth: 1}],
            // These labels appear in the legend and in the tooltips when hovering different arcs
            labels: [
                'Completed',
                'Uncompleted'
            ]
        };
        // Create doughnut chart
        var maintenanceChart = document.getElementById('maintenanceChart');
        var myDoughnutChart = new Chart(maintenanceChart, {type: 'doughnut', data: data});
        // Create bar chart
        var ctx = document.getElementById('bowserStatusChart');
        var myChart = new Chart(ctx, {
            type: 'bar',
            data: {
                labels: ['Active', 'Repair', 'Refilling', 'Deploying', 'Observation'],
                datasets: [{
                    label: 'Bowser Status',
                    data: [<?php echo sizeof($active_bowsers) ?>,
                           <?php echo (sizeof($repair_bowsers) + sizeof($minimal_repair_bowsers) + sizeof($aesthetic_repair_bowsers)) ?>,
                           <?php echo sizeof($refill_bowsers) ?>,
                           <?php echo sizeof($deploy_bowsers) ?>,
                           <?php echo sizeof($observation_bowsers) ?>
                          ],
                    backgroundColor: [
                        'rgba(255, 99, 132, 0.2)',
                        'rgba(54, 162, 235, 0.2)',
                        'rgba(255, 206, 86, 0.2)',
                        'rgba(75, 192, 192, 0.2)',
                        'rgba(153, 102, 255, 0.2)'
                    ],
                    borderColor: [
                        'rgba(255, 99, 132, 1)',
                        'rgba(54, 162, 235, 1)',
                        'rgba(255, 206, 86, 1)',
                        'rgba(75, 192, 192, 1)',
                        'rgba(153, 102, 255, 1)'
                    ],
                    borderWidth: 1
                }]
            },
            options: {scales: {yAxes: [{ticks: {beginAtZero: true}}]}}});
```

```
        @foreach($information["bowserData"] as $bowser)
            <?php
            // For each bowser data
            $bowserInfo = $bowser["attributes"];
            $bowserlon = $bowserInfo["longitude"];
            $bowserlat = $bowserInfo["latitude"];
            $bowserID = $bowserInfo["BowserID"];
            $bowserStatus = $bowserInfo["bowserStatus"];
            $bowserAddress = $bowserInfo["bowserAddress"];
            $bowserRadius = $bowserInfo["size"];
            ?>
            // Set position and icon
            var myLatlng = new google.maps.LatLng({{$bowserlat}},{{$bowserlon}});
            var icon = {
                url: @if($bowserStatus == 0)
                        "Resources/Images/waterMap.png"
                    @elseif ($bowserStatus == 1)
                        "Resources/Images/tools.png"
                    @elseif ($bowserStatus == 2)
                        "Resources/Images/waterMap.gif"
                    @elseif ($bowserStatus == 3)
                        "Resources/Images/waterMapDeploy.png"
                    @else
                        "Resources/Images/waterMap.png"
                    @endif,
                scaledSize: new google.maps.Size(50, 50), // scaled size
            };
            // Set marker and radius
            var marker{{$bowserID}} = new google.maps.Marker({
                position: myLatlng,
                title:"<?php echo "Bowser ID: " . $bowserID . " - " . $bowserAddress; ?>",
                map: mainMap,
                icon: icon
            });
            // Add circle overlay and bind to marker
            var circle = new google.maps.Circle({
              map: mainMap,
              radius: {{$bowserInfo["size"]}}/10,    // 10k Liter = 1km
              fillColor: '#AA0000',
                strokeColor: '#6600AA',
                strokeOpacity: 0.8,
                strokeWeight: 2,
                fillColor: '#6600AA',
                fillOpacity: 0.35,
            });
            circle.bindTo('center', marker{{$bowserID}}, 'position');
        @endforeach
        window.onload = main();
    })
</script>
@stop
```

```
################################################################################
   Contents: Bowser Data
################################################################################
   -->
    <tr>
        <td style="background-color: rgba(230, 230, 250, 0.5)" colspan="2">
            <div class="centered">
                <table class="chartTable">
                    <tr>
                        <td width=40%>
                            <canvas id="bowserStatusChart" width="400" height="400"></canvas>
                        </td>
                        <td class="chartDataP">
                            <p>
                            The following chart shows data regarding the status of all Bowsers across the country.
                            The Bowser status is adjusted and created by Council Members and confirmed by maintenance workers who
                            undertake a Job. Data regarding accepted maintenance tasks can be found on this page.<br>
                                <br>
                                <br>
                                The following details explain what each Status means...<br>
                            <br>
                            </p>
                            <table style="text-align: left; width: 90%; margin: auto;">
                            <?php
                                foreach ($information["bowserTypes"] as $type){
                                    echo "<tr><td width=30%><b>" . $type['name'] . ":</td><td></b> " . $type['description'] . "<br></td>
                                    </tr>";}?>
                            </table>
                            <p>
                            <br>
                            A break down of this data can be found via a Council Log In of that constituency.
                            </p>
                        </td>
                    </tr>
                </table>
            </div>
        </td>
    </tr>
```

```
################################################################################
    Contents: Maintenance Data
################################################################################
    -->
        <tr>
            <td colspan="2">
                <div class="centered">
                    <table class="chartTable">
                        <tr>
                            <td class="chartDataP">
                                <p>
                                    The following chart shows data regarding the status of all maintenance tasks submitted by Council
                                    Members and Constituents.
                                    The Maintenance Tasks are submitted and filtered according to their type and then maintenance workers
                                    select work based on their location and type.<br><br>
                                    In the event that work is not set maintenance workers can select work for themselves or council members
                                    may submit work to maintenance workers via the Edit Bowser Page.
                                    It is recommended that workers who do not collect work be repremanded according to company guidelines.
                                    Uncompleted work will remain on the database until completed or removed via the database linked to the
                                    companies server.<br><br>
                                    An overall justification of work can only be completed once the status of the job is affected by the
                                    maintenance worker. The worker can change the status once complete, or further observations of the work
                                    can be set by council members to monitor and ongoing situation.
                                </p>
                            </td>
                            <td width=40%>
                                <canvas id="maintenanceChart" width="400" height="400"></canvas>
                            </td>
                        </tr>
                    </table>
                </div>
            </td>
        </tr>
```

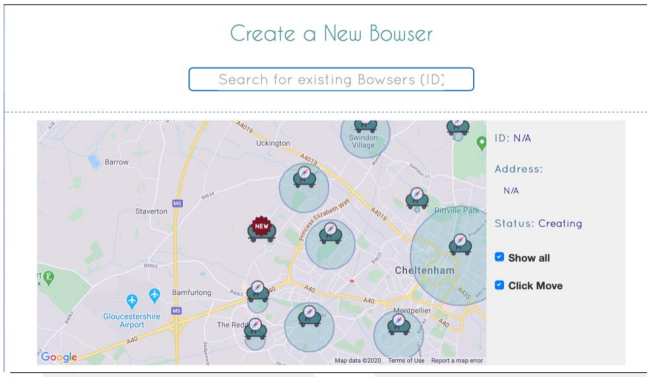
```
class CouncilHomeController extends \BaseController {

    // Get Load in information
    public function index()
    {
        $userID = Auth::user()->id;

        $constituencyID = TblCouncilMemberInformationModel::getCouncilMemberConstituency($userID);
        $constituencyID = $constituencyID[0]["attributes"]["ConstituencyID"];

        $usersName = TblUserInformationModel::getUserInformaiton($userID);
        $usersName = $usersName[0]["attributes"]["name"];

        $bowserInformation = TblBowserInformationModel::getBowsersUnderConstituencyID($constituencyID);

        $bowserLiveInformation = TblBowserInformationModel::getLiveBowsersUnderConstituencyID($constituencyID);

        $constituencyData = TblConstituencyModel::getConstituency($constituencyID);
        $constituencyData = $constituencyData[0]["attributes"];

        $profilePicture = TblUserProfilePictureModel::getProfilePicture($userID);
        $profilePicture = $profilePicture[0]["attributes"]["picture_location"];


        $councilInformation = array("name" => $usersName,
                                    "constituencyData" => $constituencyData,
                                    "bowserData" => $bowserInformation,
                                    "bowserLiveData" => $bowserLiveInformation,
                                    "picture" => $profilePicture
                                   );

        //$constituencyID = $councilInformation[0]["attributes"]["ConstituencyID"];


        return View::Make('/pages/councilhome')->with('information', $councilInformation);
    }
```

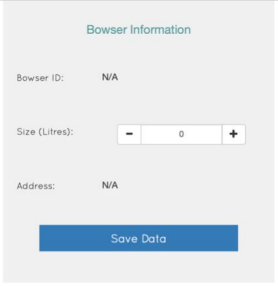
```
function main(){
    // Animate
    $(".mapCoverageButton p").addClass("focusedTabButton")

    // Move in and out of screen when necessary
    $(".bowserStatsButton").click(function(){
        if (!$(".bowserStatsButton p").hasClass("focusedTabButton")){
            $(".mainContentsSwipeTable").css("left", "-100%");
            $(".mapBackground").css("opacity", "0");
            $(".mapBackground").css("pointer-events", "none");
            $(".statsDiv").css("opacity", "1");
            $(".statsDiv").css("pointer-events", "auto")
            $(".bowserStatsButton p").addClass("focusedTabButton")
            $(".mapCoverageButton p").removeClass("focusedTabButton");
        }
    })
    $(".mapCoverageButton").click(function(){
        if (!$(".mapCoverageButton p").hasClass("focusedTabButton")){
            $(".mainContentsSwipeTable").css("left", "0");
            $(".mapBackground").css("opacity", "1");
            $(".mapBackground").css("pointer-events", "auto");
            $(".statsDiv").css("opacity", "0");
            $(".statsDiv").css("pointer-events", "none")
            $(".mapCoverageButton p").addClass("focusedTabButton")
            $(".bowserStatsButton p").removeClass("focusedTabButton");
        }
    })
}
```

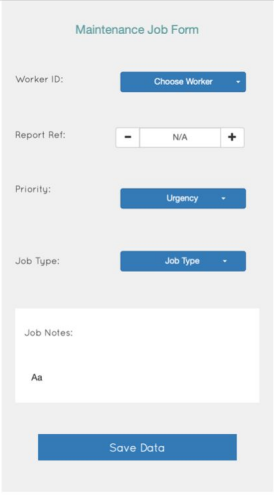
Figure DLI10

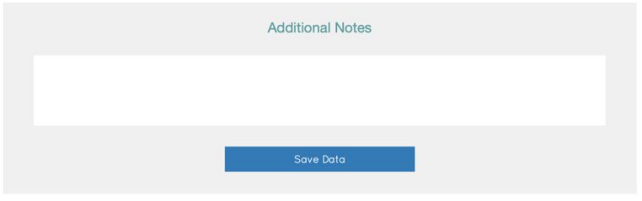
Bowser New/Edit:

The creation and updating of the bowsers are managed through this page. The map visualises all the bowsers and their coverage. The site's construction makes the modification and creation of the page simple for the user. Creating a new bowser is as simple as dropping a bowser on the map, and the modification of the bowser is easily done through pressing the bowser for modification. With the assistance of the Blade, Controller and JavaScript the development of the site page is completed by constructing the 'updating' and 'creation' functions alongside gathering the bowser information for the page.










```php
class BowserEditController extends \BaseController {
    // Get Load in information
    public function index(){ ··· }

    public function set_additional_notes(){ ··· }

    public function set_data(){ ··· }

    public function set_maintenance(){ ··· }

    public function getBowserTypesByName(){ ··· }

    public function get_all_constituency_bowsers(){ ··· }
```

```
function main(bowserID, constituency_id, userID){

    // Save button
    $("#bowserInformationSave").click(function(){ ··· })

    // Save notes
    $("#additionalNotesButton").click(function(){ ··· })

    // Submit to maintenance
    $("#maintenanceSubmission").click(function(){ ··· })

    // Number spinner
    $(document).on('click', '.number-spinner button', function () { ··· });
}
```

```
@extends('layouts.default')

@section('title')
    Bowser Edit
@stop

<?php

    $userID = $information["userID"];

    $urgencyTypes = $information["urgencyTypes"];
    $bowserTypes = $information['bowserStatusTypes'];

    $constituencyID = $information["constituency"]["ConstituencyID"];

    // Get all new bowser data
    $bowserID = $information["bowser_data"]["BowserID"];
    $bowserAddress = $information["bowser_data"]["bowserAddress"];
    $bowserStatus = $information["bowser_data"]["bowserStatus"];
    $bowserAdditionalNotes = $information["bowser_data"]["AdditionalNotes"];
    $bowserConstituencyID = $information["bowser_data"]["ConstituencyID"];
    $bowserLat = $information["bowser_data"]["latitude"];
    $bowserLong = $information["bowser_data"]["longitude"];
    $bowserCreationDate = $information["bowser_data"]["creation_date"];
    $bowserSize = $information["bowser_data"]["size"];

    $bowserStatusText = $bowserTypes[$bowserStatus]['name'];

?>


@section('pageCSS')
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-
        Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">
        <link rel="stylesheet" type="text/css" href="Resources/Pages/EditBowser/bowsereditstyles.css"/>
    <?php

include "Resources/PHP/mobileCheck.php";
if(isMobile()){ ··· }?>
@stop
```

```
@section('pageJS')

        <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
        J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-
        Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
        wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>

        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

        <script src="Resources/Pages/EditBowser/bowsereditanimate.js"></script>

        <script
                src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD5Xc19v5rFKH1cXxyQoLxC_rEESDDMlJU&callback=initMap">
        </script>

<script type="text/javascript"> ··· </script>

@stop




    <!--
################################################################################
    Type: Body
    Contents: Creates the main body
################################################################################
    -->

<!-- Page content -->
@section('content')

    @include('includes.alertSuccess')

    <!-- Page Created by Luke Gassmann -->
    <table class="mainTableContent"> ··· </table>
@stop
```

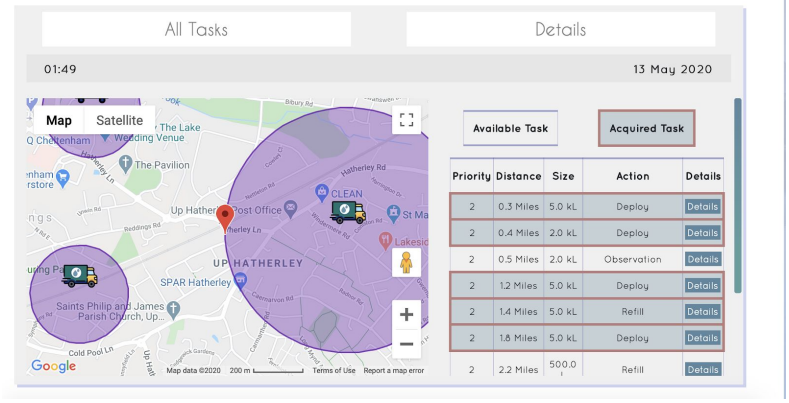
Figure DLI11

Tasks:

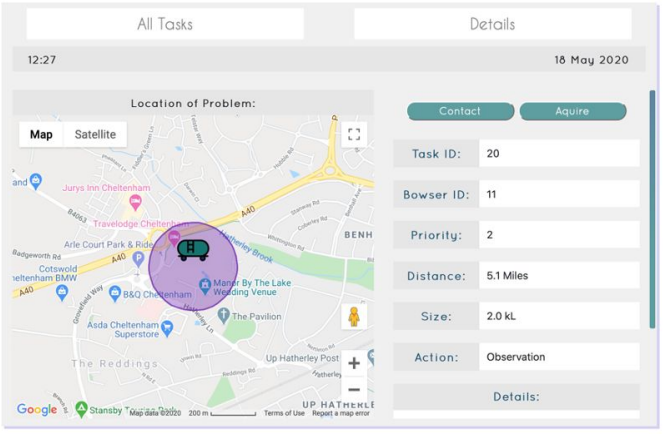
Council and maintenance workers require an area for displaying current tasks being done and the individual single tasks through a sliding panel. This easily understandable formatting that shows the current available and acquired tasks give the employee the ability

to carry out their job at ease. There are two possible ways to access the bowser task. This can be done through pressing on the bowser on the map or by clicking on the details button alongside the tasks table. through the council side of the site, the task page gives the management view of all the tasks, whereas in the maintenance side of the application, the user can acquire the task and contact the council member if anything is wrong with the task or the current bowser. This is beneficial to the company given the additional level of communication between employees and management. By constructing a Blade, Controller, Model and JavaScript for the page, the construction of the page is viable as the page requires connection to the database because accessing permissions between the two employee types using the page site are necessary.

Access: Council Member



Access: Maintenance Personnel




```
@section('swapTableArea1')
    <section class="container"> ... </section>
@stop

@section('swapTableArea2')
    <section class = "container"> ... </section>
@stop
```

```php
use Response;       // for ajax response reply

class taskListController extends \BaseController { //class extends baseController

    // create page
    public function index() {

        return View::Make('/pages/taskList')->with('mapDataA', array("track" => true, "bowsers" => []))->with('mapDataB', array("track" =>
        false, "bowsers" => []));
    }


    // ajax send list of tasks to page
    public function update() { ··· }


    // sort tasks by distance
    public function SortByDistance(array &$list) { ··· }


    // ajax assign task to user
    public function assign() { ··· }



    // ajax unassign task
    public function unassign() { ··· }



    // ajax finish task
    public function finish() { ··· }


    //  martinstoeckli and Hein, D., 2016. Measuring The Distance Between Two Coordinates In PHP. [online] Stack Overflow.
    //  Available at: <https://stackoverflow.com/a/10054282> [Accessed 26 April 2020].
    /**
     * Calculates the great-circle distance between two points, with
     * the Vincenty formula.
     * @param float $latitudeFrom Latitude of start point in [deg decimal]
     * @param float $longitudeFrom Longitude of start point in [deg decimal]
     * @param float $latitudeTo Latitude of target point in [deg decimal]
     * @param float $longitudeTo Longitude of target point in [deg decimal]
     * @param float $earthRadius Mean earth radius in [m]
     * @return float Distance between points in [m] (same as earthRadius)
     */
    public static function vincentyGreatCircleDistance(
      $latitudeFrom, $longitudeFrom, $latitudeTo, $longitudeTo, $earthRadius = 6371000)
    { ··· }
}
```

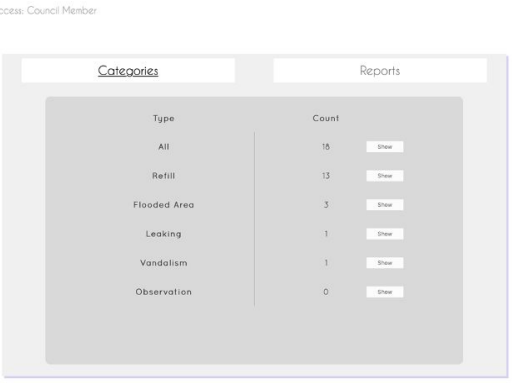
```php
use Illuminate\Database\Eloquent\Model;     // import the model class


class TblMaintenanceTasks extends Model { //define class and inherit from the imported model class

    //table used by the model and remove the requirement of timestamps
    protected $table = 'tblMaintananceTask';

    public $timestamps = false;

    //function to write a new task record to the database
    public function createTask($values) { ··· }

    //used to assign a task to a maintenance user
    public function assignMaintenance($Task, $User) { ··· }

    //used to unassign a task if taken by a maintenance user
    public function unassignMaintenance($Task) { ··· }

    //update the done column to be 1 if the task is completed
    public function finishTask($Task) { ··· }

    //get a specific record by the unique taskID
    public function getTask($ID) { ··· }

    //used to compare the stats of maintanance users. Used by council users.
    public function getComparisonStatsReporter($userID){ ··· }

    //used by council users to compare statistics between the maintenance users
    public function getComparisonStatsMaintenance($userID){ ··· }

    //get all completed tasks from the database
    public function getAllComplete(){ ··· }

    //get all NOT completed tasks from the database
    public function getAllNotComplete(){ ··· }
}
```

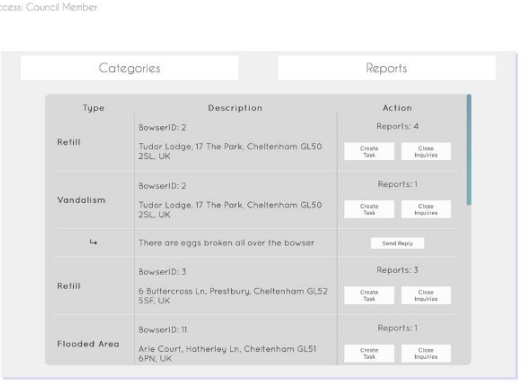
```
// Show current date and time
function getTime() { ··· }
getTime(); // call function on page load to prevent space being empty for 1000ms
setInterval(getTime, 1000); // Call function every second to update the time and date

let rowIndex = 1;
const tasksTable = document.getElementById("tasksTable");

// Update Tasks
function update() { ··· }
setTimeout(update, 3000); // load tasks after page load
setInterval(update, 30000); // Update list of tasks every 30s


// update row in table
function updateRow(row) { ··· }

// Update bowsers on map
function updateMap(bowser) { ··· }

// Display task details
function details(taskID) { ··· }

// Aquire task
function aquireTask() { ··· }

// Return task to unassigned
function unassignTask() { ··· }

// Finish task
function finishTask() { ··· }
```

Figure DLI12

Public Inquiries:

The public have a page for giving the council requests in case of specific problems with a bowser. From this the council requires a page to be able to access the inquiries that the public have given the council. This page gives the council a count of all the types of requests sent from the public and the actual reports in detail through a sliding panel. From this the council can set up a new task or close the task. The stakeholder of the application has requested a similar outtake on the inquiries by giving the exact number of each task type. The second panel covers how multiple reports for one bowser can be spotted through a report number above each type of problem, for ease of visualisation for the council member using the page. Due to the use of public requests, the public reports table is used for the execution of the page, alongside functions for locating the nearest bowser, emailing responses when descriptions are given, creating and closing tasks.





```
@section('swapTableTitle1')
    Categories
@stop
@section('swapTableTitle2')
    Reports
@stop

@section('swapTableArea1')
    <!-- Summary of inquiry catagories -->
    <div class="col col2"> ··· </div>
@stop
@section('swapTableArea2')
    <!-- Details of inquiries -->
    <div class="col"> ··· </div>
@stop


@section('breadcrumb')
    / <a href="/Inquiries">Public Inquiries</a>
@stop

@section('introtabletitle')
    Public Inquiries
@stop

@section('introtableusertype')
    Council Member
@stop


<!-- Page content -->
@section('content')
    <br>
    @include("includes.introTable")
    <br>
    <br>
    @include("includes.swapTable")
    <br>
    <br>

    <!-- form to compose email -->
    <div id="emailForm" class="hidden"> ··· </div>
@stop
```

```
use Response;       // for ajax response reply

class inquiriesController extends \BaseController {

    // create page
    public function index() { ··· }

    // ajax send reply email
    public function send_email() { ··· }

    // ajax create task for inquiry
    public function create_Task() { ··· }

    // ajax delete inquiries matching bowser and type
    public function delete_Inquiries() { ··· }

    // sort inqueries by bowser and type
    public function SortInqueries(array &$list) { ··· }

    // find nearest bowser to lat lng co-ordinates
    public function findNearestBowser($compLat, $compLong) { ··· }
}
```

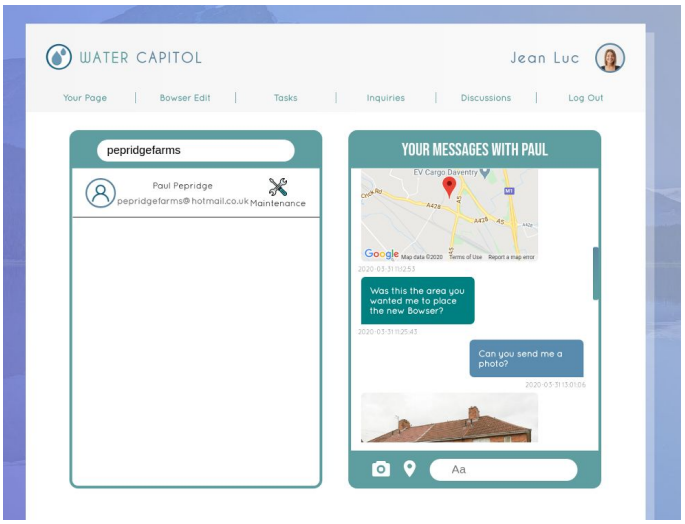
```
use Illuminate\Database\Eloquent\Model;     // import the model class


class tblPublicRequests extends Model {  //define class and inherit from the imported model class


    //define the table being used and remove the need for timestamps
    protected $table = 'tblPublicRequests';

    public $timestamps = false;


    //get all requests from the table
    public function getRequests() {
        $taskData = tblPublicRequests::all();
        return $taskData;
    }

    //get a specific record from the table based on the ID
    public function getFromID($ID) {
        $Data = DB::table("tblPublicRequests")->where('sendID', $ID)->first();
        return $Data;
    }

    //used to remove requests if their ID and type are the same to help prevent duplicate entries to the database
    public function removeByIDandType($ID, $type) {
        $success = tblPublicRequests::where('sendID', $ID)->where('type', $type)->delete();
        return $success;
    }
}
```

```
// show subset of inquiries to user
function showDetails(type) { ··· }


// show form to create email
function createEmail(reportID, description) { ··· }


// send email using form
function sendEmail() { ··· }


// create task based on inqueries
function createTask(bowserID, type) { ··· }


// close all inqueries for bowser and type combination
function closeInquiries(bowserID, sendID, type, quantity) { ··· }
```

Figure DLI13

Discussions page:

All employees within the business require a format for communication. That said the discussions page allows for any logged in user to enter names of the employee they wish to contact and are able to send them a message. This communication is done for the ease of working together and the ability to answer questions quickly between employees. This form of communication allows for more informal connections between maintenance workers when completing a task and for an ease of communication between all parties due to reducing the time taken conducting a formal email to one another throughout the day. The execution of the messaging requires various functions from loading user messages, to the scrolling feature of the application.




```
<!-- Page content -->
@section('content')


    <table class="mainTableContent">
        <tr class="topRowDetails">
            <td width=50%>


                <!--
###############################################################################
    Contents: Search Users
###############################################################################
    -->

                <div class="searchUsersDiv"> ... </div>
            </td>
            <td>


                <!--
###############################################################################
    Contents: Display Messages
###############################################################################
    -->

                <div class="messagesDiv"> ... </div>

            </td>
        </tr>
        <!--
###############################################################################
    Contents: Back options
###############################################################################
    -->
        <?php if (isMobile()){?>
        <tr>
            <td>
            </td>
            <td> ... </td>
        </tr>
        <?php } ?>
    </table>


<!--
###############################################################################
    Contents: Google map sending
###############################################################################
    -->

    <div class="googleMapsDisplay"> ... </div>

@stop
```

```
use Illuminate\Database\Eloquent\Model;     // import the model class


class TblMessages extends Model { //define class and inherit from the imported model class


    //set the table being accessed and remove the need for time stamps
    protected $table = 'tblMessages';

    public $timestamps = false;

    //get a specific record based on the ID
    public function get_messages($id) { ... }


    //used to insert a new entry to the database. This will be done when a message is sent
    public function sendMessage($values){ ... }
}
```

Figure DLI14

```
class MessageDesktopController extends \BaseController {

    // Get Load in information
    public function index(){ ··· }

    /*
    ################################
    Purpose: Get accounts
    ################################
    */
    public function getUsersAccounts(){ ··· }

    /*
    ################################
    Purpose: Get the users messages
    ################################
    */
    public function get_messages(){ ··· }

    /*
    ################################
    Purpose: Send a message
    ################################
    */
    public function send_message(){ ··· }

    /*
    ################################################
    Purpose: Create a connection between users
    ################################################
    */
    public function create_connection(){ ··· }

    /*
    ################################
    Purpose: Save an image
    ################################
    */
    public function save_image(){ ··· }

}
```

```
/*
############################################################
Purpose: In the event that no users are found
############################################################
*/
function noUsersFound(){ ··· }
/*
############################################################
Purpose: Connect to users to enable messaging between them
############################################################
*/
function connect_users(friend, council){ ··· }
/*
############################################################
Purpose: Loads the messages onto the page
############################################################
*/
function loadMessages(data_set, creation_date, council_member, friend){ ··· }
/*
############################################################
Purpose: Scroll the messages to the bottom
############################################################
*/
function scrollToBottom(){ ··· }
/*
############################################################
Purpose: Load all users to select for messaging
############################################################
*/
function loadUsers(users_data, council_id){ ··· }
/*
############################################################
Purpose: Loads the messages for users
############################################################
*/
function post_load_users(friend, council_id){ ··· }
/*
############################################################
Purpose: Allows users to send photo messages
############################################################
*/
function activate_photo(council, friend){ ··· }
/*
############################################################
Purpose: Get the users messages with their other user
############################################################
*/
function getMessages(friend_id, council_id){ ··· }
/*
############################################################
Purpose: Set new message attributes
############################################################
*/
function postGetMessages(data){ ··· }
```

```
/*
############################################################
Purpose: Sends a message for a user
############################################################
*/
function sendMessage(message, council, friend, type){ ··· }
/*
############################################################
Purpose: Gets Users from the database
############################################################
*/
function getUsers(search, council_id){ ··· }
/*
############################################################
Purpose: Create a map
############################################################
*/
function createMap(map, lat_, lng_){ ··· }
/*
############################################################
Purpose: Google maps main
############################################################
*/
function activate_google_maps(council, friend){ ··· }
mobile_mode = false;
/*
############################################################
Purpose: Begining function
############################################################
*/
function main(council_id, friend, mobile_mode_var){ ··· }
```

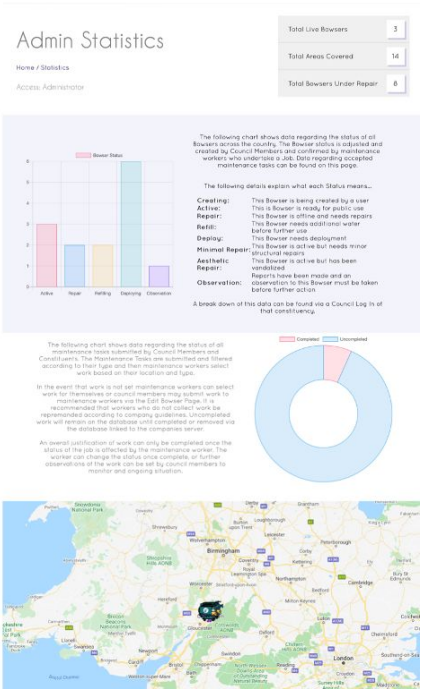
Figure DLI15

Admin:

Homepage:

The management of both the council and the maintenance users is required to safely overview the work being done. An administrator section was created for monitoring the work being done alongside the overall span of the bowsers and constituencies. A bar chart is utilised for breaking-down different parameters that need to be monitored. Additionally, there is a pie chart for visualising both completed and uncompleted work. On this site page there is also a map of the general area with all bowsers. From this a general view can be given for

the administrator to monitor the overall functions of the site. The Blade and Controller are required for the creation of the charts alongside the mapping and overall ease of use of the page.




```
@extends('layouts.adminLayout')

@section('title')
    Admin Statistics
@stop
<?php
    // Get all bowser data
    $active_bowsers = array();
    $repair_bowsers = array();
    $refill_bowsers = array();
    $deploy_bowsers = array();
    $minimal_repair_bowsers = array();
    $aesthetic_repair_bowsers = array();
    $observation_bowsers = array();
    // Format bowser data
    foreach ($information["bowserData"] as $bowser){ ··· }
?>
@section('pageCSS')
        <link rel="stylesheet" type="text/css" href="Resources/Pages/AdminHome/adminhomestyles.css"/>
    <?php
include "Resources/PHP/mobileCheck.php";
if(isMobile()){ ··· }?>
@stop
```

```
@section('pageJS')
        <script src="Resources/Pages/AdminHome/adminhomeanimate.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.bundle.min.js"></script>
        <script
                src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD5Xc19v5rFKH1cXxyQoLxC_rEESDDMlJU&callback=initMap">
        </script>
<script type="text/javascript">
    $(document).ready(function() {
        // Select the Map Div and focus on this location
        var mainMap = new google.maps.Map(document.getElementById('map'), { ··· });
        // Create data for chart
        data = { ··· };
        // Create doughnut chart
        var maintenanceChart = document.getElementById('maintenanceChart');
        var myDoughnutChart = new Chart(maintenanceChart, { ··· });
        // Create bar chart
        var ctx = document.getElementById('bowserStatusChart');
        var myChart = new Chart(ctx, { ··· });
        @foreach($information["bowserData"] as $bowser)
            <?php
            // For each bowser data
            $bowserInfo = $bowser["attributes"];
            $bowserlon = $bowserInfo["longitude"];
            $bowserlat = $bowserInfo["latitude"];
            $bowserID = $bowserInfo["BowserID"];
            $bowserStatus = $bowserInfo["bowserStatus"];
            $bowserAddress = $bowserInfo["bowserAddress"];
            $bowserRadius = $bowserInfo["size"];
            ?>
            // Set position and icon
            var myLatLng = new google.maps.LatLng({{$bowserlat}},{{$bowserlon}});
            var icon = { ··· };
            // Set marker and radius
            var marker{{$bowserID}} = new google.maps.Marker({ ··· });
            // Add circle overlay and bind to marker
            var circle = new google.maps.Circle({ ··· });
            circle.bindTo('center', marker{{$bowserID}}, 'position');
        @endforeach
        window.onload = main();
    })
</script>
@stop
@section('breadcrumb')
    / <a href="/adminHome">Statistics</a>
@stop
@section('introtabletitle')
    Admin Statistics
@stop
@section('breadcrumbRoot')
    <a href="/adminHome">Home</a>
@stop
@section('introtableusertype')
    Administrator
@stop
```

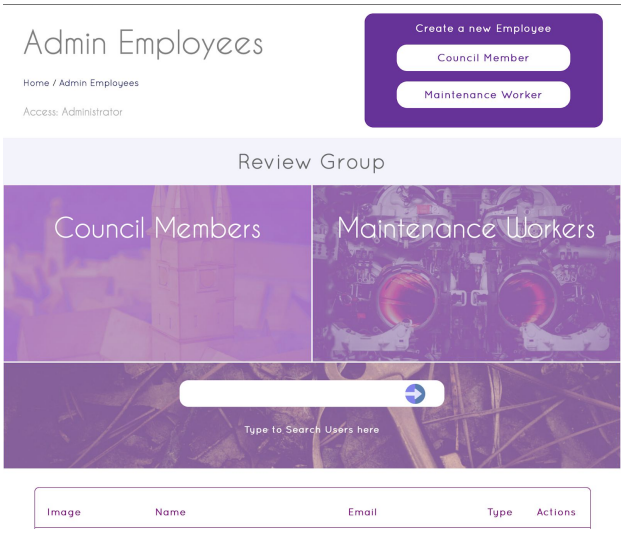
```
@section('bbih1heading')
    Total Live Bowsers
@stop
@section('bbih1number')
    {{sizeof($information["bowserLiveData"])}}
@stop
@section('bbih2number')
    {{sizeof($information["bowserData"])}}
@stop
@section('bbih3number')
    {{sizeof($information["bowserOfflineData"])}}
@stop
    <!--
##################################################
    Type: Body
    Contents: Creates the main body
##################################################
    -->
<!-- Page content -->
@section('content')
    <table class="mainTableContent">
        <tr> ··· </tr>
        <!-- ··· -->
        <tr> ··· </tr>
        <!-- ··· -->
        <tr> ··· </tr>
        <!-- ··· -->
        <tr> ··· </tr>
        <tr> ··· </tr>
    </table>
@stop
```

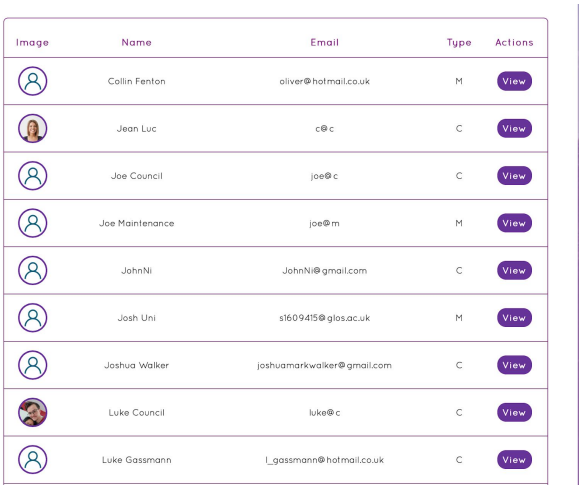
```php
class AdminHomeController extends \BaseController { //inherit default functionality from the baseController

    // Get Load in information
    public function index(){
        //get the authorised user ID
        $userID = Auth::user()->id;

        //get the constituency ID of the authenticated user
        $constituencyID = TblCouncilMemberInformationModel::getCouncilMemberConstituency($userID);
        $constituencyID = $constituencyID[0]["attributes"]["ConstituencyID"];

        //get the user information
        $usersName = TblUserInformationModel::getUserInformaiton($userID);
        $usersName = $usersName[0]["attributes"]["name"];


        //get all the bowsers in seperate arrays for all, live bowsers and offline bowsers
        $bowserInformation = TblBowserInformationModel::getAllBowsers();
        $bowserLiveInformation = TblBowserInformationModel::getAllLiveBowsers();
        $bowserOfflineInformation = TblBowserInformationModel::getAllOfflineBowsers();

        //get 2 arrays showing the tasks complete and not complete
        $maintenance_complete = TblMaintenanceTasks::getAllComplete();
        $maintenance_uncomplete = TblMaintenanceTasks::getAllNotComplete();

        $maintenance_tasks = array("complete"=>$maintenance_complete, "uncomplete"=>$maintenance_uncomplete);

        //get the task types for the bowsers
        $bowserTypes = TblBowserTaskID::getBowserTaskTypes();
        $bowserTypesDict = array();
        foreach($bowserTypes as $bowserType){ ... }

        //get the constituency
        $constituencyData = TblConstituencyModel::getConstituency($constituencyID);
        $constituencyData = $constituencyData[0]["attributes"];

        //create an array with the returning data
        $councilInformation = array("name" => $usersName,
                                    "constituencyData" => $constituencyData,
                                    "bowserData" => $bowserInformation,
                                    "bowserLiveData" => $bowserLiveInformation,
                                    "bowserOfflineData" => $bowserOfflineInformation,
                                    "bowserTypes"=>$bowserTypesDict,
                                    "maintenanceTasks"=>$maintenance_tasks
                                    );


        //return the view to create the page
        return View::Make('/pages/adminhome')->with('information', $councilInformation);
    }
```

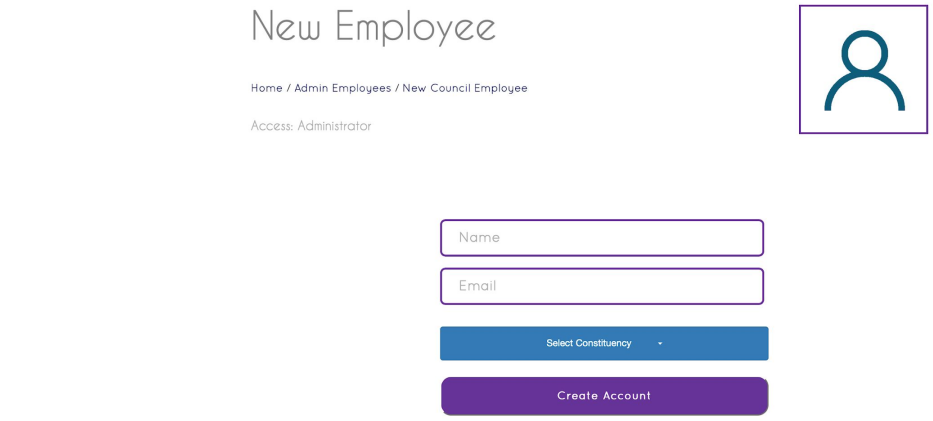
Figure DLI16

Employee View:

This directory page is used for creating and viewing all employees. The page gives a full list of all employee names, their emails and a photo of the user. There is also a button for viewing the specific employee in more detail. Additionally, a search box is used for searching any specific employee. This page requires loading in necessary information from the tables, a filtration function for specific types of employees to be visualised, a function for the searching employees and specific functions for the creation, deletion and updating of employees.








```
@extends('layouts.adminLayout')
@section('title')
    Admin Employees
@stop
@section('pageCSS')
        <link rel="stylesheet" type="text/css" href="Resources/Pages/AdminEmployees/adminemployeesstyles.css"/>
    <?php
include "Resources/PHP/mobileCheck.php";
if(isMobile()){ ··· }?>
@stop
@section('pageJS')
        <script src="Resources/Pages/AdminEmployees/adminemployeesanimate.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.bundle.min.js"></script>
        <script
                src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD5Xc19v5rFKH1cXxyQoLxC_rEESDDMlJU&callback=initMap">
        </script>
<script type="text/javascript"> ··· </script>
@stop
@section('breadcrumb')
    / <a href="/adminEmployees">Admin Employees</a>
@stop

@section('introtabletitle')
    Admin Employees
@stop

@section('breadcrumbRoot')
    <a href="/adminHome">Home</a>
@stop

@section('introtableusertype')
    Administrator
@stop
<!--
##########################################################################
    Type: Body
    Contents: Creates the main body
##########################################################################
    -->
<!-- Page content -->
@section('content')
    <table class="mainTableContent"> ··· </table>
@stop
```

```
class AdminEmployeesController extends \BaseController {
    //class inherits from the baseController containing default functionality

    // Get Load in required data for the page
    public function index()
    { ··· }


}
```

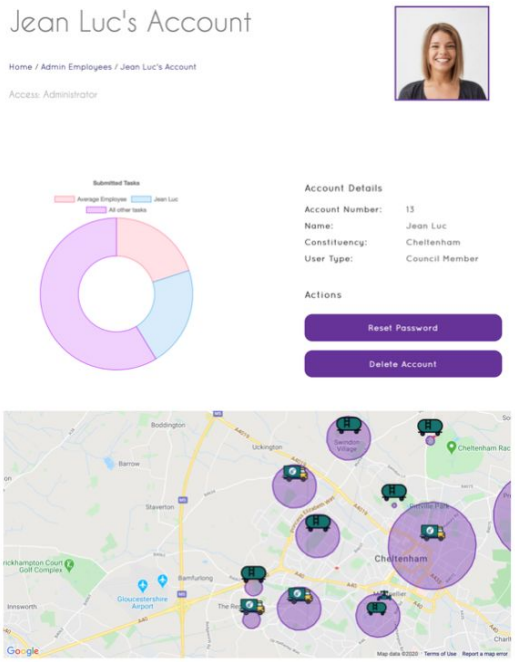
```
class AdminEmployeeViewController extends \BaseController {

    // Get Load in information
    public function index()
    { ··· }

    // ajax reset user
    public function reset_user(){ ··· }


    // ajax delete user
    public function delete_user(){ ··· }


    // ajax create user
    public function create_user(){ ··· }

}
```

```
function main(){
    // Council Search
    $("#reviewGroupImage1").click(function(){ ··· })

    // Maintenance Search
    $("#reviewGroupImage2").click(function(){ ··· })

    // Email Search
    $("#searchUsersButton").click(function(){ ··· })

    // Create Users
    $(".userViewButton").click(function(){ ··· })

    $(".createButton").click(function(){ ··· })
}
```

Figure DLI17

Employees:

Unlike the previous section, specific employees can be visualised with additional information such as their submitted tasks. On this page, the site admin can reset the passwords and delete the specified employees. In addition, the specific bowsers the employee is managing is also visible. Using functions to delete, reset password and gather user information from the tables in SQL, this page gives a good visual representation for the admin to manage the employees.




```
@extends('layouts.adminLayout')
@section('title')
    Admin Employees
@stop
@section('pageCSS')
        <link rel="stylesheet" type="text/css" href="Resources/Pages/AdminEmployeeView/adminemployeeviewstyles.css"/>
    <?php
include "Resources/PHP/mobileCheck.php";
if(isMobile()){ ... }?>
@stop
@section('pageJS')
        <!-- provide the csrf token -->
        <meta name="_token" content="{{csrf_token()}}" />
        <script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
        crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-
        Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
        wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="Resources/Pages/AdminEmployeeView/adminemployeeviewanimate.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.bundle.min.js"></script>
        <script
                src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD5Xc19v5rFKH1cXxyQoLxC_rEESDDMlJU&callback=initMap">
        </script>
<script type="text/javascript"> ... </script>
@stop
@section('breadcrumb')
    / <a href="/adminEmployees">Admin Employees</a> / <a href="/adminEmployeeView?user_account=<?php echo $_GET['user_account']; ?>"><?php
        if ($_GET['user_account'] == "new"){ ... }
        else{ ... }
      ?></a>
@stop
@section('introtabletitle')
    <?php
        if ($_GET['user_account'] == "new"){ ... }
        else{ ... }
        ?>
@stop
@section('breadcrumbRoot')
    <a href="/adminHome">Home</a>
@stop
@section('introtableusertype')
    Administrator
@stop
<?php
    if ($_GET['user_account'] == "new"){ ... }
?>
```

```
class AdminEmployeeViewController extends \BaseController {

    // Get Load in information
    public function index()
    { ... }


    // ajax reset user
    public function reset_user(){ ... }


    // ajax delete user
    public function delete_user(){ ... }


    // ajax create user
    public function create_user(){ ... }

}
```

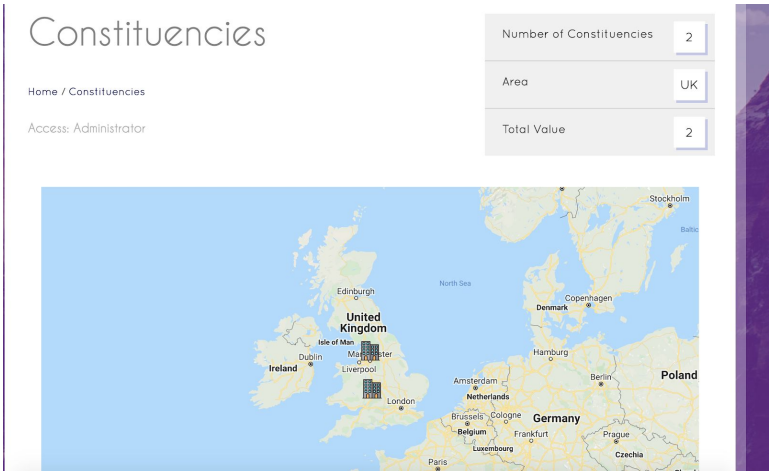
```
    <!--
##############################################################################
    Type: Body
    Contents: Creates the main body
##############################################################################
    -->
<!-- Page content -->
@section('content')
@include('includes.alertSuccess')
    <table class="mainTableContent">
        <tr>
            <td width=75%>
                @include('includes.introTable')
            </td>
            <!--
##############################################################################
    Contents: Users Image
##############################################################################
    -->
            <td> ... </td>
        </tr>
        <tr> ... </tr>
    </table>
@stop
```

```
function main(user_account_number){

    // Reset a users password
    $("#resetPasswordAccount").click(function(){ ... });

    // Remove a users account
    $("#deleteAccount").click(function(){ ... });
}


function mainCreate(user_type){ ... }
```

Figure DLI18

Constituencies:

The administrator requires access to the constituencies in the current location for management and visualisation purposes. This page gives the administrator the privilege of viewing all constituencies in a map and in a table. This page requires the location of constituencies from the constituencies table.



```
@extends('layouts.adminLayout')

@section('title')
    Admin Employees
@stop

@section('pageCSS')
<!-- page CSS -->
        <link rel="stylesheet" type="text/css" href="Resources/Pages/AdminConstituencies/adminconstituenciesstyles.css"/>
    <?php
//if on mobile, use the mobiles styles sheet as well
include "Resources/PHP/mobileCheck.php";
if(isMobile()){ ··· }?>
@stop
@section('pageJS')
        <!-- include the javascript files -->
        <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
        J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-
        Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
        wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6" crossorigin="anonymous"></script>
        <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
        <script src="Resources/Pages/AdminConstituencies/adminconstituenciesanimate.js"></script>
        <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.9.3/Chart.bundle.min.js"></script>
        <script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD5Xc19v5rFKH1cXxyQoLxC_rEESDDMlJU&callback=initMap">
        </script>
<script type="text/javascript"> ··· </script>
@stop
@section('breadcrumb')
    / <a href="/adminConstituencies">Constituencies</a>
@stop
@section('introtabletitle')
    Constituencies
@stop
@section('breadcrumbRoot')
    <a href="/adminHome">Home</a>
@stop
@section('introtableusertype')
    Administrator
@stop
@section('bbih1heading')
    Number of Constituencies
@stop
@section('bbih2heading')
    Area
@stop
@section('bbih3heading')
    Total Value
@stop
@section('bbih1number')
    {{sizeof($information["constituencies"])}}
@stop
@section('bbih2number')
    UK
@stop
```

```
@section('bbih3number')
    {{sizeof($information["constituencies"])}}
@stop
    <!--
##############################################################################
    Type: Body
    Contents: Creates the main body
##############################################################################
    -->
<!-- Page content -->
@section('content')
@include('includes.alertSuccess')
    <table class="mainTableContent">
        <tr> ··· </tr>
        <!--
##############################################################################
    Contents: Map
##############################################################################
    -->
        <tr> ··· </tr>
        <!--
##############################################################################
    Contents: Constituency Data
##############################################################################
    -->
        <tr> ··· </tr>
    </table>
@stop
```

```php
<?php
//namespace Bowser;

use Illuminate\Database\Eloquent\Model;     // if database access only

//use Illuminate\Auth\UserTrait;            // if user should be logged in and database access is needed
//use Illuminate\Auth\UserInterface;        // if user should be logged in and database access is needed


    // note if you get errors like "Call to undefined method" there are two likely reasons
    // first laravel has cached the files before you created the new one - if so enter the commans from the following
    //      https://stackoverflow.com/questions/45474010/laravel-call-to-undefined-method-which-only-happens-in-production
    //
    // second you are using a name of an already existing class eg Password - if so try another name or change the namespace
//INSERT INTO `tblBowserInformaiton`(`bowserAddress`, `bowserPostcode`, `bowserStatus`, `AdditionalNotes`, `ConstituencyID`) VALUES
("Hatherley Ln", "GL51 6PN", "repair", "This is a test", 1)

class TblConstituencyModel extends Model {


    /* ... */
    protected $table = 'tblConstituency';       // if database table needs loading as properties of class for js access (js access because
    of controller's "return View::Make('/pages/password')->with('password', $pass);" )

    /* ... */
//  protected $hidden = array('Password', 'ResetToken');        // if loaded values need to not be sent to js


    // class all() already exists

    // get name
    public function getConstituency($constituencyID) { ... }

    public function getAll(){
        return TblConstituencyModel::all();
    }
}
```

```php
class AdminConstituenciesController extends \BaseController {

    // Get Load in information
    public function index()
    {
        $userID = Auth::user()->id;

        $constituencies = TblConstituencyModel::getAll();

        foreach ($constituencies as $constituency){ ... }

        $usersName = TblUserInformationModel::getUserInformaiton($userID);
        $usersName = $usersName[0]["attributes"]["name"];

        $profilePicture = TblUserProfilePictureModel::getProfilePicture($userID);
        $profilePicture = $profilePicture[0]["attributes"]["picture_location"];


        $councilInformation = array("name" => $usersName,
                                    "picture" => $profilePicture,
                                    "constituencies" => $constituencies
                                   );

        //$constituencyID = $councilInformation[0]["attributes"]["ConstituencyID"];


        return View::Make('/pages/adminconstituencies')->with('information', $councilInformation);
    }

}
```

Figure DLI19

Page Navigation

Public site:

Within the public's site section, a navigation bar is visible at the top of each page. This allows for ease of navigation though the application. As visible in NV1 the navigation bar reads from most important to least important site page for ease of use. This allows for users to navigate to more important links easily. Within certain sections, links to other pages are used for an additional ease of use. Hidden at the bottom right of the home page is a link to the main login of the council, admin and maintenance workers section of the site.

Figure NV1

The FAQ page adds links to other aspects of the site in case the user requires other pages such as the bowsers near me and the refill times. NV2 has examples of the links visible on the FAQ page.

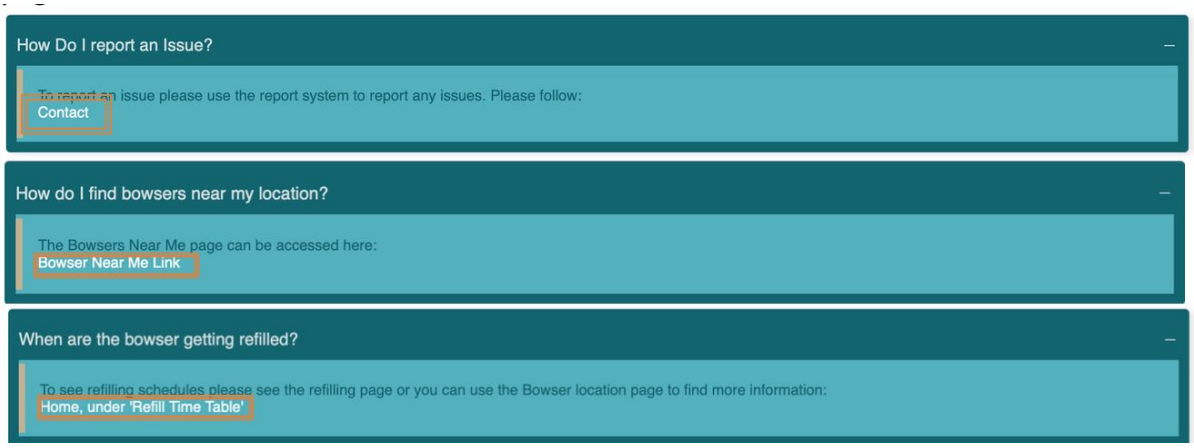


Figure NV2

Council member's site:

This section contains a similar navigation system for employees. Based on the required task needed to be done the employee can navigate from page to page with ease to visibly access the public inquiries or the discussion page for example.

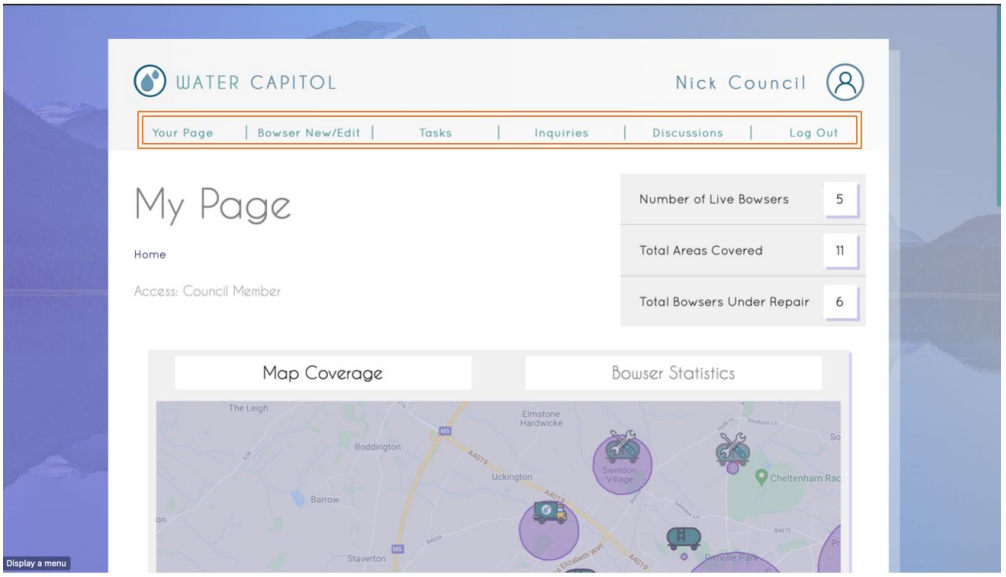


Figure NV3

Maintenance member's site:

Maintenance workers have a refined version for the specific requirements to their jobs, from being able to access the tasks required to be done and the discussion page for asking questions to the council members that have set the tasks required to be completed.

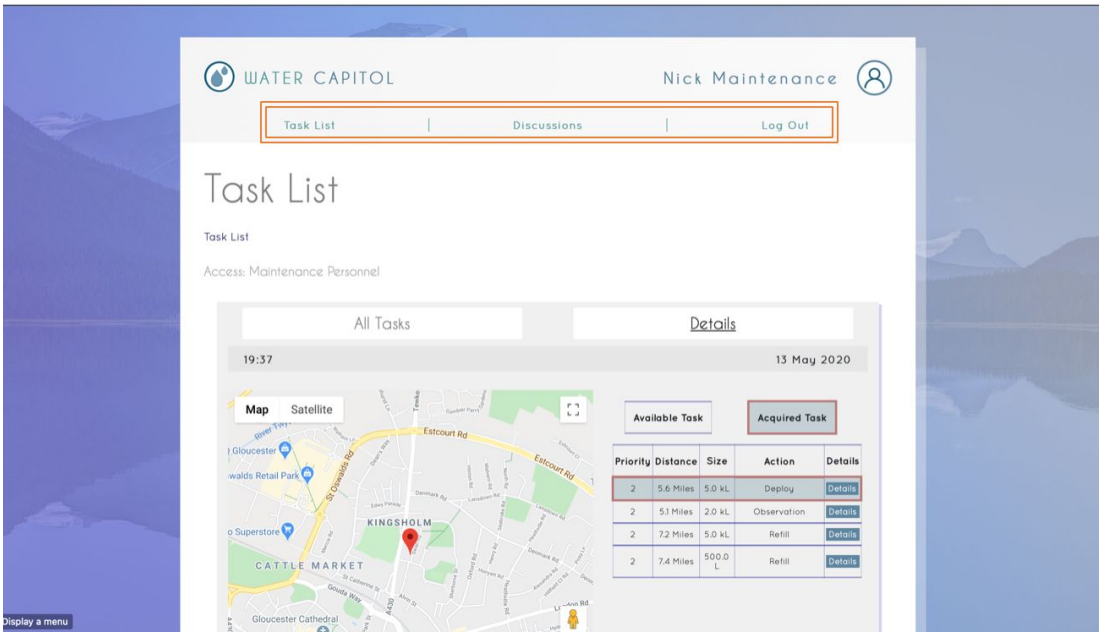


Figure NV4

Administrator site:

A similar view can be seen within the administrator's site section as the previous with constituencies and employees' sections over that of the council or maintenance navigation bar options. This is done as the administrator requires access to employee information and constituency information for management purposes.

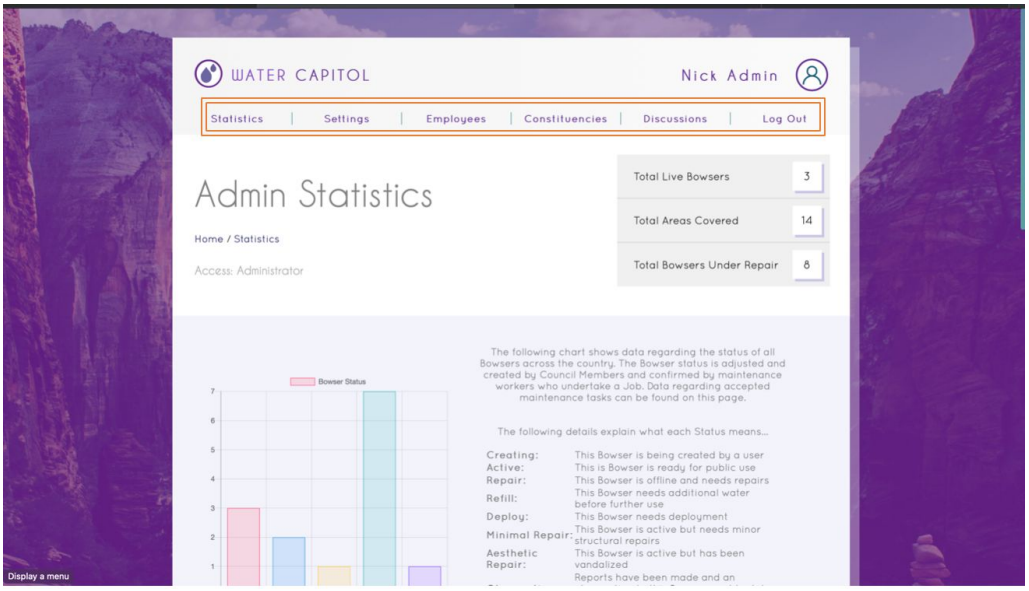


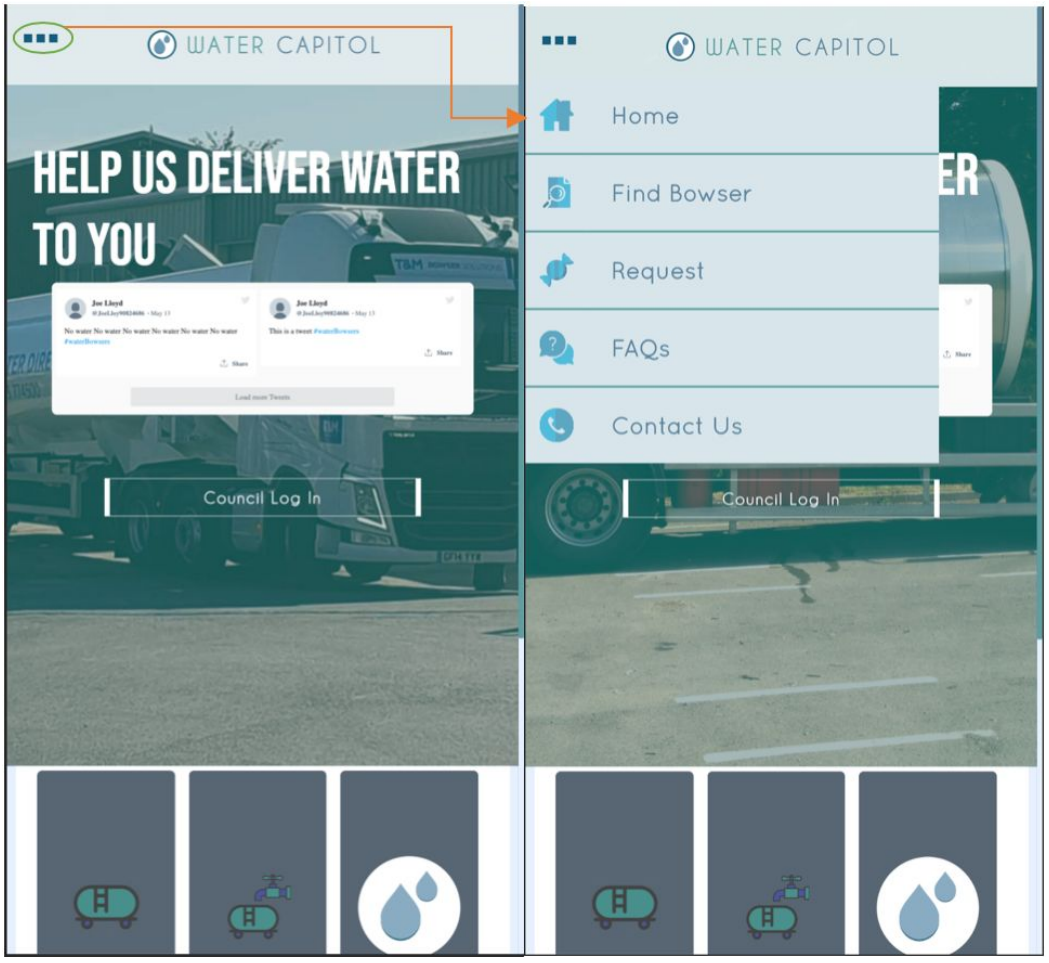
Figure NV5

Mobile navigation:

The mobile versions of the site differ slightly in the displaying and navigation. As visible in the previous section navigation bars with the titles of the differing sections allow for an ease of navigation of all possible pages for the user. Within the mobile versions of these pages the navigation bar is hidden for additional space to be given to the rest of the page information. The use of a hamburger menu is used as it easily displays all the site links and hides it to allow for the main page information to be given. Located at the top of pages once pressed the menu of all pages appears. Visible below are the different sections that have the ability of using the hamburger menu. The main user of the mobile version of the page is the maintenance user as they are the workers that will be remotely located away from the main offices. This also adds additional mobility as the website can be easily used with any mobile device when on the job repairing bowsers and picking up new tasks while away from the main headquarters.

Sections with hamburger menu

Below is an example of the public home section with the on click example of the hamburger menu. The administrator, maintenance and council members all have the same button on the top-left of the site.



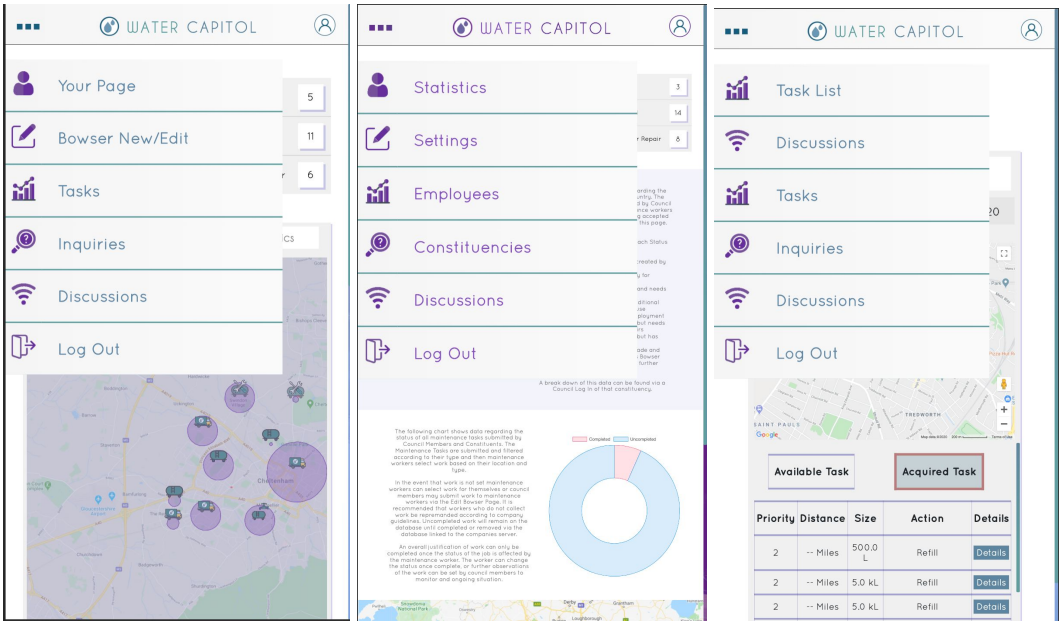

WATER CAPITOL
Your Page
Bowser New/Edit
Tasks
Inquiries
Discussions
Log Out
WATER CAPITOL
Statistics
Settings
Employees
Constituencies
Discussions
Log Out
WATER CAPITOL
Task List
Discussions
Tasks
Inquiries
Discussions
Log Out
Available Task
Acquired Task
Priority
Distance
Size
Action
Details
Refill
Details

Figure NV6

Security

Security should never be overlooked, and so features are implemented to try and protect against common threats.

SQL injection is a very common attack which could result in loss of user data on a large scale and potentially lead to lawsuits due to GDPR. SQL injections can be achieved by adding a speech mark to break a query. If this is successful, there will be a server error and from there an attacker can find the names of tables using the ‘table_schema’ and return all the information from them on screen. This means a user table could be compromised so the attacker could see the password hashes and email addresses of all users. (Boyd and Keromytis, 2004)

Eloquent uses parameter binding behind by default when a query is executed to prevent commands being executable in the search bars. (Fidao, 2018)

Parameter binding works by building a query using placeholders where a user input is needed. This means a query is essentially prebuilt and a user input is inserted into a placeholder protecting the database from attack. (Winand, n.d.)

As mentioned, this can be demonstrated on the website by entering a quote mark into a search query. If a server error is returned, the site is not protected against SQL injection. Water Capitol though returns to the council home page as there is a list of bowsers at the bottom.

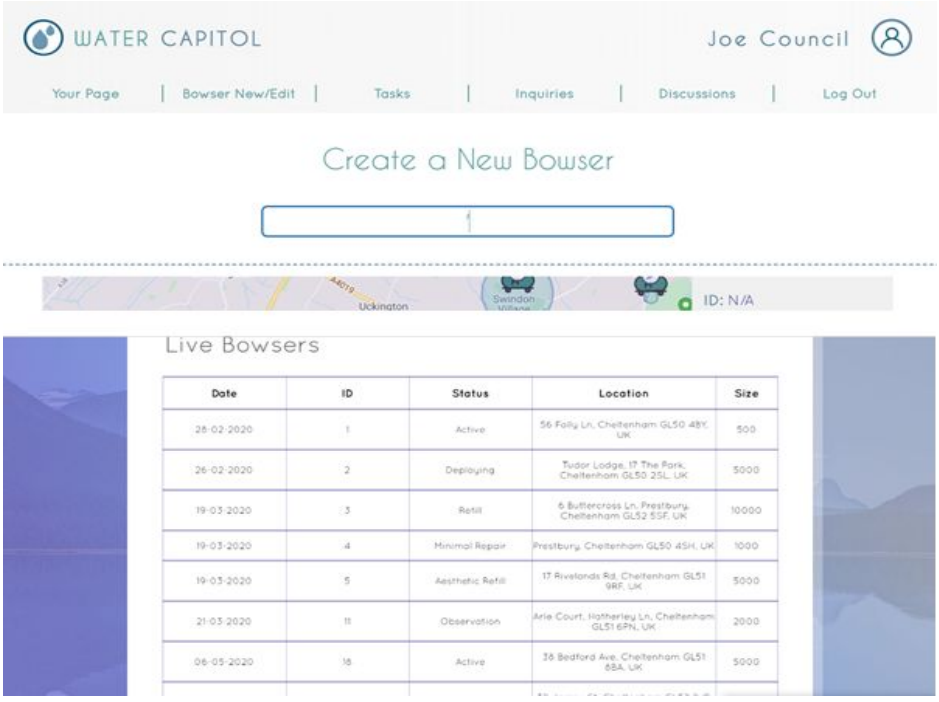


Figure STY1: Demonstrating Inserting a Quote Mark in Water Capitol to Show Protection Against SQL Injection

Passwords are also inserted into the database hashed when an account is created. This is done by running the entered password through a hashing algorithm before inserting it and makes the password more difficult to read than just having it in plain text. (Almeshekah, Gutierrez, Atallah and Spafford, 2015). The image below shows the hashed passwords in the testing phase of the database.

id	name	password	email	resetToken
13	Jean Luc	$2y$13$AaVrHBG6oJvzLEtAYvZ7fOmnH.GU0n.ISxbBKj0pN8/...	c@c	NULL
14	Paul Pepridge	$2y$13$fCeESbUoL2Yi8cJDPwNc1ugAFQ6pW8WobSqy2MuZI4b...	pepridgefarms@hotmail.co.uk	NULL
15	Collin Fenton	$2y$13$/qOYvvbgoN14dusWhnp53OOv6WR.4Kz2LEuk4tiRJwY...	oliver@hotmail.co.uk	NULL
24	JohnNi	$2y$13$xudjQrY4mo5CAARSQYqAuOUyjol0lEnzPmAI3GgEQQU...	JohnNi@gmail.com	NULL

Figure STY2: Password Hashing in the Database Sample

Hashing is not infallible though as the same password will have the same hash which means common passwords will appear the same which compromises multiple accounts. There are also a lot of tools online which are used for de-hashing a password including rainbow tables which are publicly available. (Kumar et al., 2013)

Building on this, salting can be used to essentially create a random key to base an encryption off which makes it much more difficult to break. This is used in the Water Capitol implementation using BCrypt which is a form of salt-hashing based on the blowfish cipher. (Hatzivasilis, Papaefstathiou and Manifavas, 2020) This means the passwords are stored in a very secure way and the SQL injection prevention previously mentioned also means viewing the tables at all is difficult and the implementation is above recommended standards for a website.

However, the server may be vulnerable to a slow loris attack. Slow loris is a form of Denial of Service (DoS) attack which requires a low bandwidth to perform. (Grabovsky et al., 2018) Code is also available publicly which makes this attack extremely simple to run. It works by sending a request to a server but sending bytes slowly to prevent a timeout. Servers running apache are vulnerable to the attack because apache works by allotting threads to each request sent. (Menasce, 2003) This means when the slow loris is running, threads are created for each request being sent and kept open because bytes are sent to prevent a timeout, but the threads are kept open which slows the server drastically and prevents legitimate users trying to access it. (Shorey, 2018)

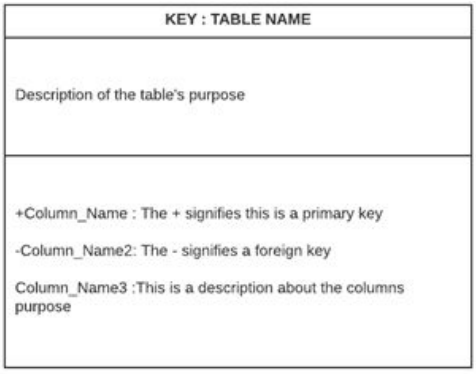
Water Capitol is running apache on the servers as shown below which makes it vulnerable to a slow loris.

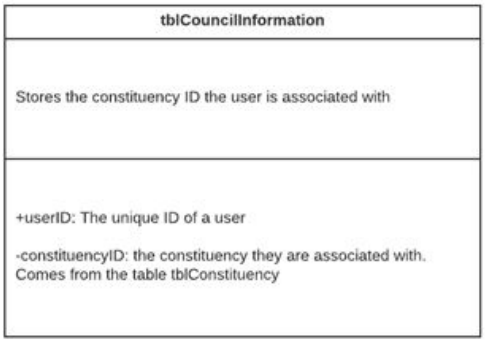
Figure STY3: Server Data to Show Apache is Being Run

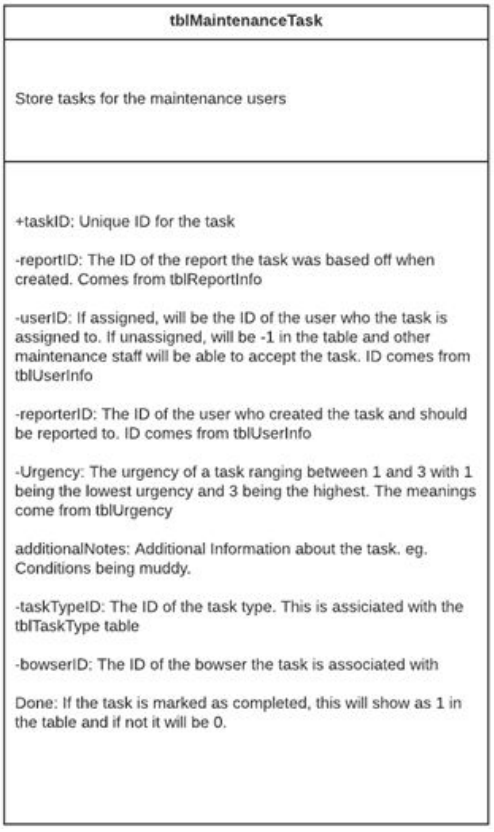
SQL

Table UMLs and Explanations

There are a total of 11 tables used for this website. The descriptions of each table are outlined below with the table name, table purpose and the column names including markings to show primary and foreign keys (see key).







tblConstituency

Stores information about the constituencies

+constituencyID: unique ID of the constituency

constituencyName: The name of the constituency. eg. Cheltenham, or Manchester

constituencyPostcode: The postcode of the constituency area

additionalNotes: Notes added by a council member about the constituency

longitude: The Longitude of the constituency to show on a map

latitude: the latitude of the constituency to show on a map

radius: The radius size of constituency area to be shown on a map

tblBowserInformation

Stores the information about the bowsers themselves

+bowserID: Unique ID of the bowser

bowserAddress: The location of the bowser in an address format which makes it more human-readable

-bowserStatus: Used to represent the current status of a bowser.. the meaning of the value is shown on the tblTaskType table.

additionalNotes: Notes added by members of staff about the bowser. eg. one is consistently leaking or in a high-demand area.

-constituencyID: The ID of the constituency the bowser is in. Comes from tblConstituency table.

latitude: The latitude of the bowser used for integration with the Google Maps API

longitude: The longitude of the bowser used for integration with the Google Maps API

creation_date: The date and time the bowser was created

size: The capacity of the bowser

tblMessages

Stores messages being sent to users

+message_ID: the iunique ID of the message

-thread_ID: The unique ID for the thread. Messages between the same user have the same thread ID. It comes from tblMessageOverview in the messaging_id column

date_sent: The date and time the message was sent

Message_Type: The type of message sent. T = Text, M = Map and P = Photo

Contents: The contents of the message itself

-Sender_ID: The ID of the person who sent the message

tblMessageOverview

Used to store an overview of the messaging thread

+Messaging_ID: The unique thread ID to hold information about the thread

creation_date: The date and time the thread was created

-user_1: The ID of the user who sent the first message. ID comes from tblUserInfo

-user_2: The ID of the user who received the first message. ID comes from tblUserInfo

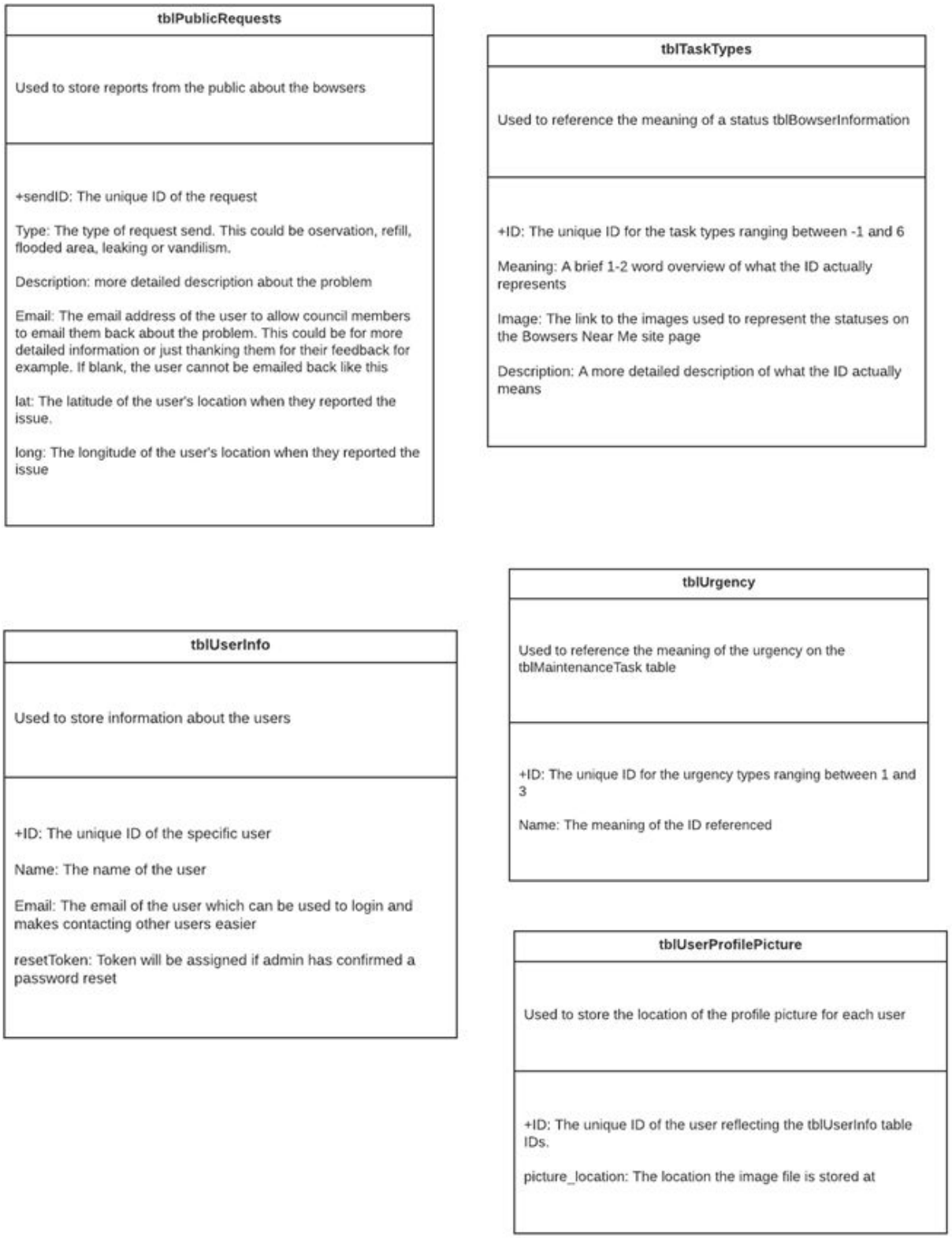


Figure SQL1: Database Explanation Diagrams. Created Using Lucidchart (Lucidchart, 2020)

Entity Relationship Diagram (ERD)

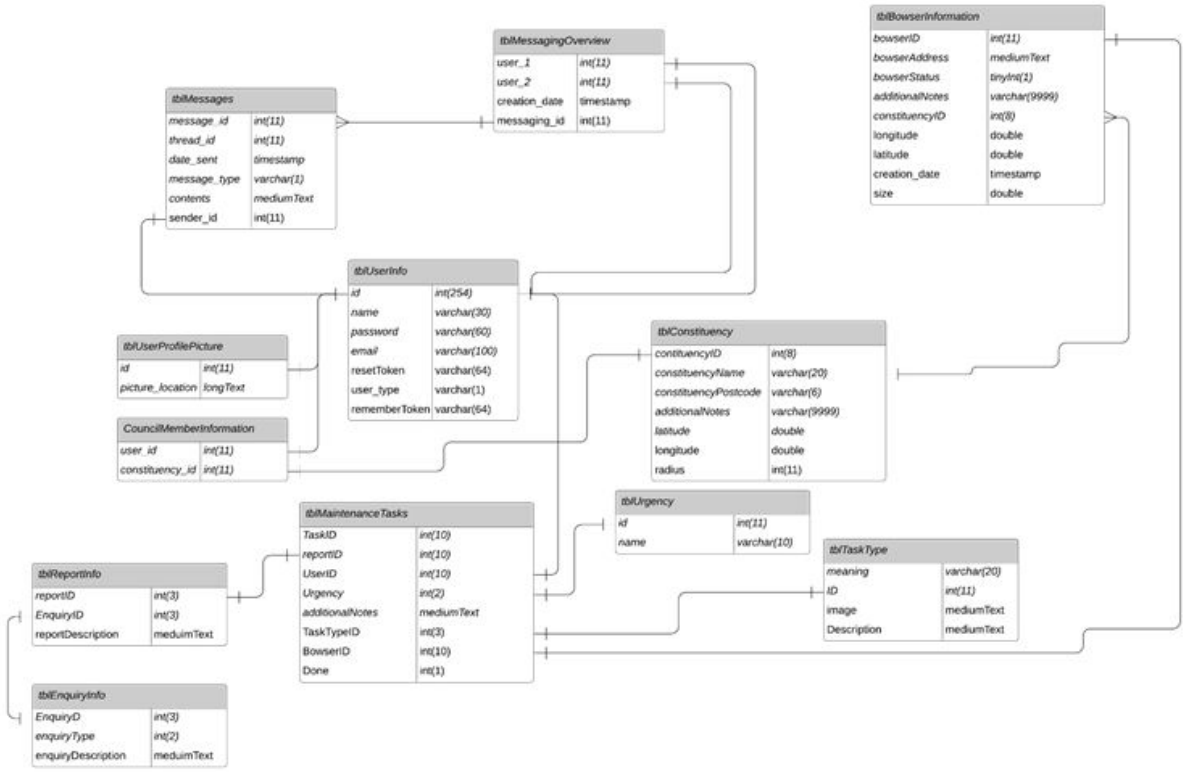


Figure SQL2: Water Capitol ERD. Created Using Lucidchart (Lucidchart, 2020)

The ERD demonstrates the connections between the tables and shows where the foreign keys on tables are coming from and whether there is a one-to-one relationship or a one-to-many relationship.

Eloquent

Eloquent is a class-based framework used by Laravel for database interaction. Each table will have a model with functions written to make the referencing of tables easier. (Laravel Team, 2018) The framework is used by extending the model class included with the package and then defining the Table as shown below.

```php
use Illuminate\Database\Eloquent\Model;     // if database access only

class TblBowserInformationModel extends Model {


    /**
     * The database table used by the model.
     *
     * @var string
     */
    protected $table = 'tblBowserInformation';
```

Figure SQL3: Example Code for Eloquent Setup

The model class contains functions to join tables, view all entries, insert entries and delete entries as well as many others by default. By inheriting from it, new functions can be created unique to that specific table. This is usually in the form of specific queries and created using functions already defined in eloquent as shown below.

```php
// get name
public function getLiveBowsersUnderConstituencyID($constituencyID) {
    $bowserData = TblBowserInformationModel::where('ConstituencyID', '=', $constituencyID)
        ->where('bowserStatus', '=', '0')
        ->orwhere('bowserStatus', '=', '4')
        ->orwhere('bowserStatus', '=', '5')
        ->orwhere('bowserStatus', '=', '6')->get();
    return $bowserData;
}
```

Figure SQL4: Example Query Writing Using Eloquent

The above query is used to get the bowsers which are currently active which means only bowsers with the same constituency entered and where the bowserStatus column is 0, 4, 5 or 6 will be returned. This is used to show the members of the public what the nearest bowsers to them are as seeing bowsers which are not in use may not be useful to them.

It works by having the constituencyID parsed and then setting the bowserData variable by calling the where method (default model method) from its own class and parsing

the columns name, operation and variable to check against. It then checks for the bowser statuses using where again to build the query. The get function is then used to execute the query and the resulting IDs are returned in a dictionary.

The getAddress method is used to simply select all entries in a single column and uses the get method to execute the query again. This method is being used to get all the addresses of the bowsers to show the user. It is the equivalent of SELECT “bowserAddress” from tblBowserInformation in SQL. The code is shown below

```
public function getAddresses() {
    $data = TblBowserInformationModel::select('bowserAddress')->get();

    return $data;
}
```

Figure SQL5: Select Query Using Eloquent

The model class also has a method known as all to get everything from the able. This does not use the get method to execute as all takes care of the execution as well because it is a prewritten query. It represents the equivalent of SELECT * from tblBowserInformation in SQL.

```
// get name
public function getAllBowsers() {
    $bowserData = TblBowserInformationModel::all();
    return $bowserData;
}
```

Figure SQL6: Select All Query Using Eloquent

Overall, eloquent has been used to make database access easier and more secure than the traditional mySQL database operations due to the provided features and the encapsulated nature of the model class. (Porzio, 2020)

User Stories

Public Users

User Story	As a customer I would like to report if a bower is empty so that the council can refill it.
Description of Done	Members of the public can access, fill in and submit a report form to inform the council of the status of a bowser.
Acceptance Criteria	There is a report form that sends the status and location of a bowser to the database. It is available to the public. The reports on the database are accessible to the council.
Story points	5

Figure US1

This user story was implemented on the Request page by adding a form that allows members of the public to select the type of problem, its location on a map, provide a description, and enter an email for a response if needed.

The code used for this was:

- request.blade.php
- request.js
- Google Map api
- RequestController.php
- tblReportInfoModel.php

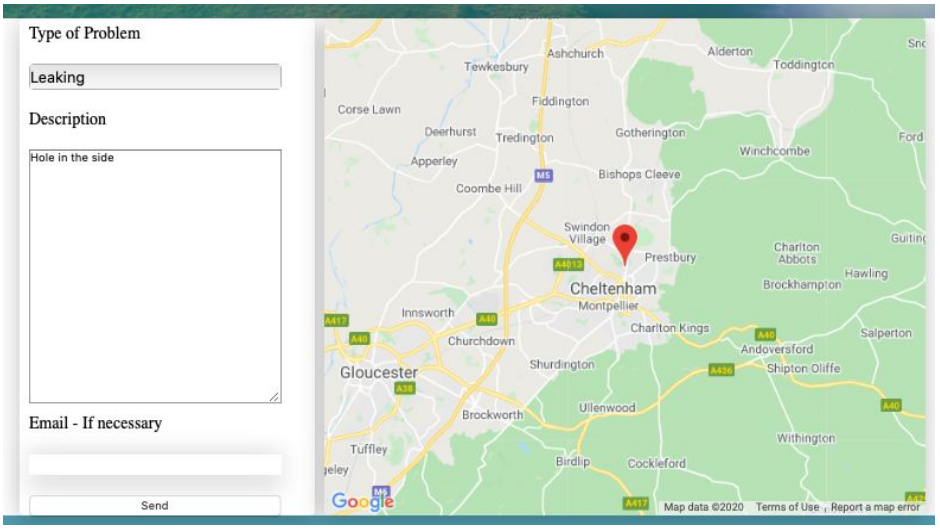


Figure US2

For council members to view reports, the Inquiries page shows two tabs, the first summarises by type of problem allowing the council user to filter the next tab, which shows each problem and the option to reply if there is an email attached.

The code used for this was:

- tblReportInfoModel.php
- inquiriesController.php
- councilinquiry.blade.php
- councilInquiry.js

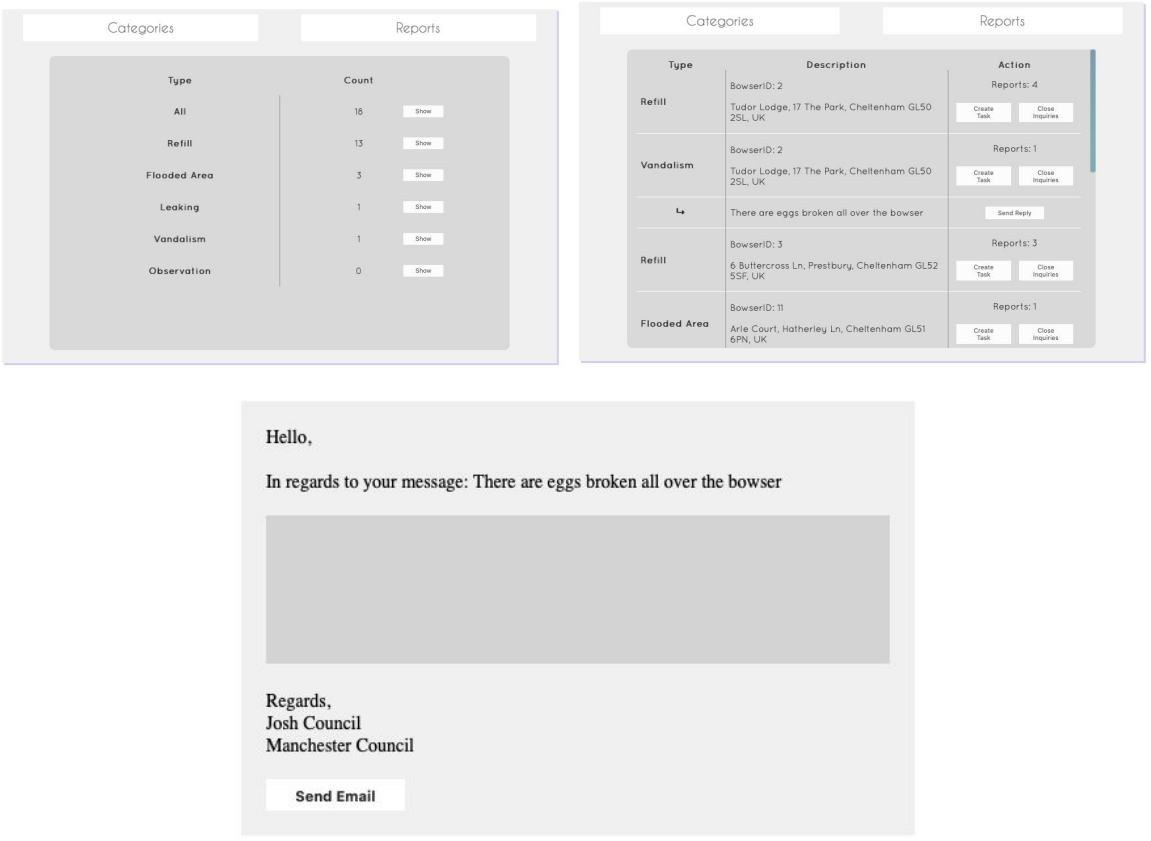


Figure US3

User Story	As a customer I would like an FAQ page so that I can find answers to common questions.
Description of Done	There is a page which has Q&A's of frequently asked questions to assist customers
Acceptance Criteria	There is a page with many FAQ's. It is available to the public.
Story points	5

Figure US4

This user story was implemented by creating the FAQ page that is populated with a set of questions that can be expanded to view their answers, many of which contain links to other parts of the website to improve the user's experience.

The code used for this was:

- faq.blade.php

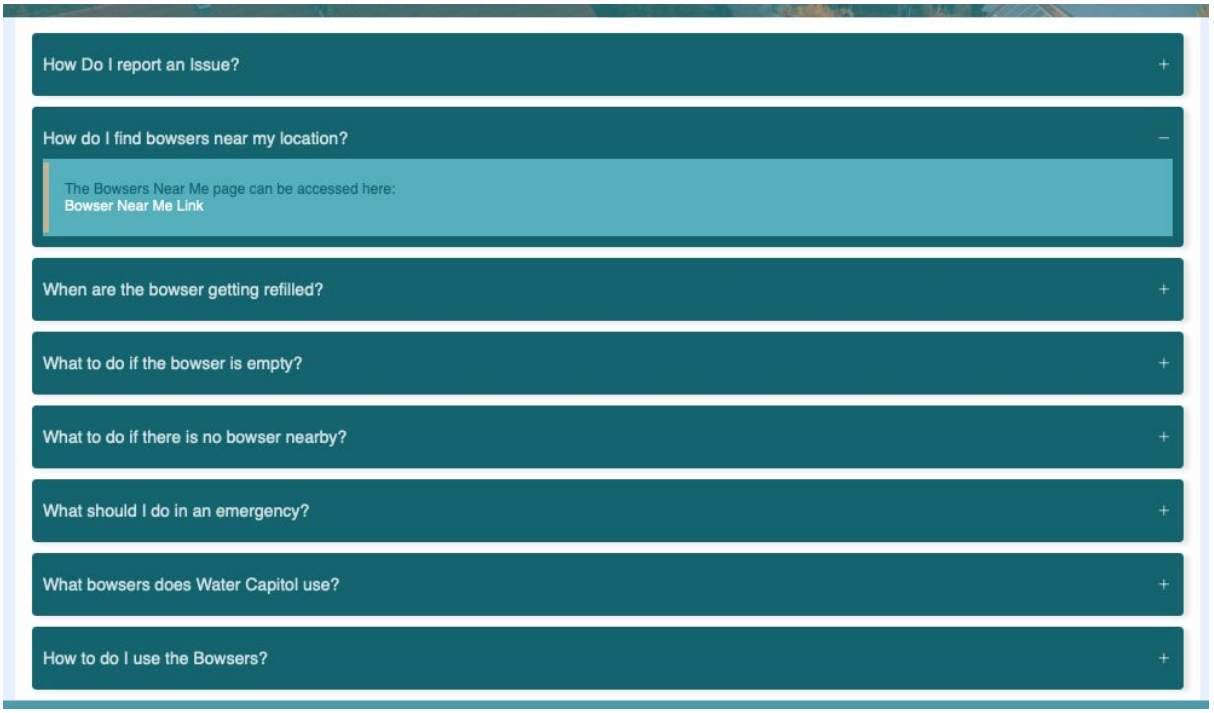


Figure US5

User Story	As a customer I want bowsers recommended to me so that I can access the closest bowser with the most water in it.
Description of Done	There is page which shows nearby bowsers
Acceptance Criteria	The page has a map showing bowsers. There is a sorted list of the nearby bowsers. It is available to the public.
Story points	8

Figure US6

User Story	As a customer I want to be informed of the status of bowsers so that I know if I can access water from them.
Description of Done	There is a page which shows the status of each nearby bowser.
Acceptance Criteria	The page has a map showing bowsers and their statuses. More information can be viewed about each bowser by clicking on them. It is available to the public.
Story points	8

Figure US7

This user story was implemented by creating the public's bowserNearMe page which contains a pair of tabs. The first shows all bowsers and their status both on a map and as a list sorted by distance. The second tab shows details for a selected bowser.

The code used for this was:

- TblBowserInformationModel.php
- BowserNearMeController.php
- vincentyGreatCircleDistance() & SortByDistance() in taskListController.php
- Google map api
- bowserNearMe.blade.php
- bowsers.js

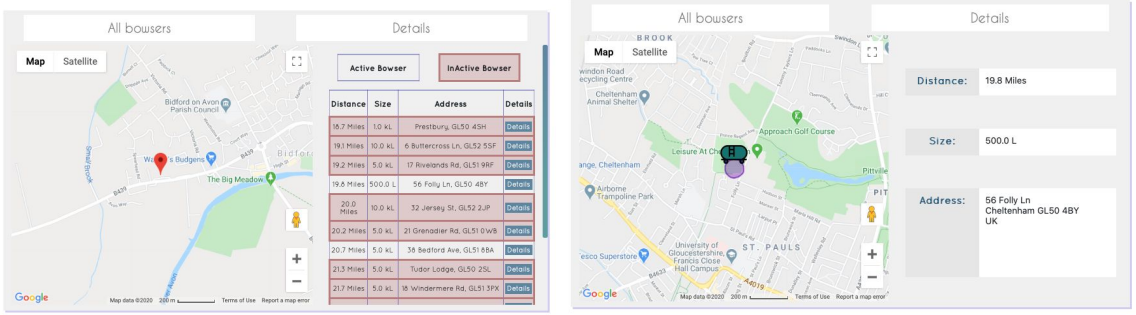


Figure US8

User Story	As a customer I want a live feed on the homepage of all issues that are being dealt with to reassure me progress is being made in my local area.
Description of Done	Once a member of the public has shared their location, they can see a localised list of recent developments.
Acceptance Criteria	There is a localised list of recent developments. They have the system time of when the update was posted. It is available to the public.
Story points	13

Figure US9

User Story	As a customer I want a twitter hashtag so that I can have an alternative means of communication.
Description of Done	There is a live feed on the homepage which shows tweets containing a specified hashtag
Acceptance Criteria	The homepage has a live feed of tweets. It is available to the public.
Story points	5

Figure US10

Due to the similarities of these user stories it was decided by the client that the twitter hashtag was the lower priority and was therefore the only one implemented in the time available. It is shown as a live feed on the public homepage of the website.

The code used for this was:

- home.blade.php
- https://apps.elfsight.com/p/platform.js api

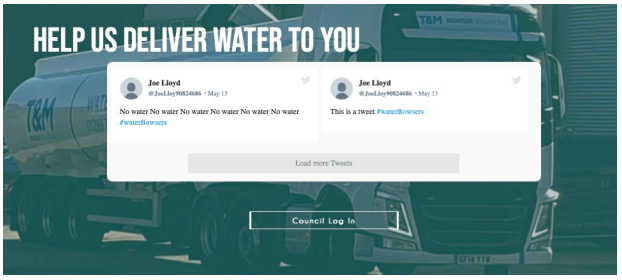


Figure US11

User Story	As a customer I want a quick way to access questions such as a search box, so that I don't spend too much time looking for an answer.
Description of Done	There is a search box for answers to questions related to the website.
Acceptance Criteria	There is a box that can be used to search the website. It is available to the public.
Story points	3

Figure US12

Due to time constraints this user story was not implemented as it was a lower priority than implementing other features that were deemed a core part of the website.

User Story	As a customer I want to be informed of the refill schedule of bowsers so that I know when to expect them to be refilled.
Description of Done	There is a publicly available refill schedule.
Acceptance Criteria	There is a refill schedule on the homepage. It is available to the public.
Story points	5

Figure US13

This user story was implemented by adding a refill schedule in a readable location on the website's homepage.

The code used for this was:

- home.blade.php

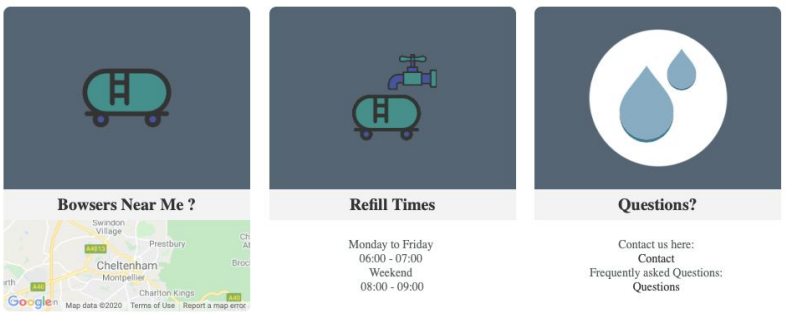


Figure US14

Logged in Users

User Story	As an employee I want a login system so that I can securely view details relevant to me in more in depth.
Description of Done	There is an account creation and login system that provides access to restricted pages specific to the user.
Acceptance Criteria	Users can securely login. User pages will not be accessible without logging in. There are different account access types for different types of users.
Story points	2

Figure US15

This user story was implemented by adding the login page to enter user credentials and using Laravel Auth to restrict access.

The code used for this was:

- login.blade.php
- login.js
- LoginController.php
- filters.php
- routes.php
- User.php

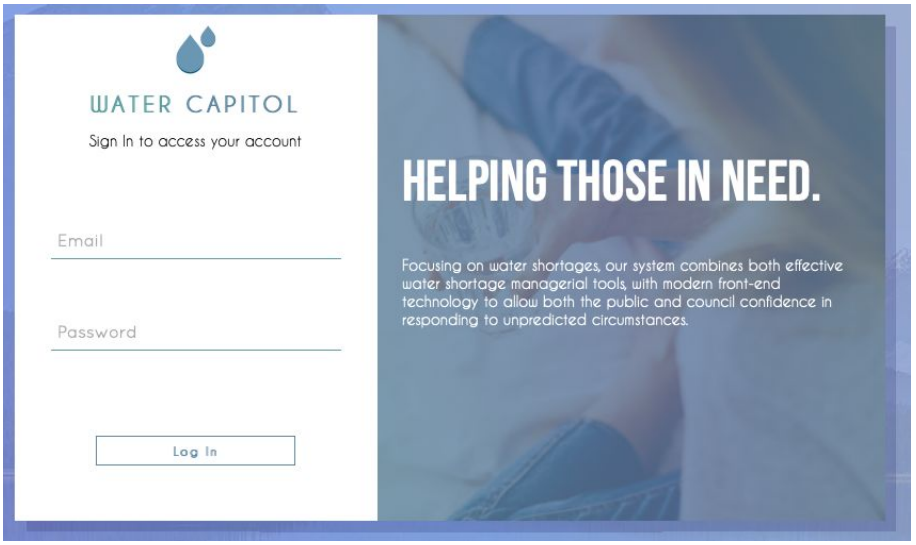


Figure US16

User Story	As a user of this system I want to be able to view this on a mobile device so that I can access it on the go.
Description of Done	The website is fully compatible with mobile devices
Acceptance Criteria	Users can access the complete functionality of the website on mobile devices. The website responds dynamically to the size and type of device it is on.
Story points	13

Figure US17

This user story was implemented for the public and maintenance user pages as the council and administrator users will use laptops or desktops. To detect if it is a mobile device the user agent is checked in a custom mobileCheck function.

The code used for this was:

- mobileCheck.php
- BowserNearMe/mobilestyles.css
- contactMobileStyles.css
- faqStyles.css
- homeStylesMobile.css
- homeLayoutMobileStyles.css
- Login/styles.css
- messageMobileStyles.css
- TaskList/mobileStyles.css

Council Users

User Story	As a council user I want a recommended distribution of bowser locations so I can have assistance in deciding deployment.
Description of Done	There will be a system which can recommend bowser locations, while allowing the user to adjust them to fit with priorities and locations.
Acceptance Criteria	There is a visual recommendation of deployment. Users make the final decision on where and what bowsers will be deployed. It is available to the Council.
Story points	20

Figure US18

User Story	As a council user I want to be able to change the status of a bower so that I can alert the public and maintenance.
Description of Done	Council users can select a bowser and change its status and this information will be shown to the public and maintenance.
Acceptance Criteria	There is a map of bowsers that can be selected by council users. The status of a selected bowser can be changed by council users. Changing the status will generate a task for maintenance. The updated status will show on the public map.
Story points	2

Figure US19

User Story	As a council user I want to be able to adjust the priority of maintenance tasks so that I can prioritise based on cost and need.
Description of Done	Council users can set the priority of maintenance tasks and this will be shown to maintenance users.
Acceptance Criteria	The priority for a maintenance task can be set by council users. The priority of tasks is visible to maintenance users.
Story points	2

Figure US20

These three stories were implemented using the bowseredit page, the distribution user story was partially implemented by displaying the approximate coverage for a bowser based on its volume when it is placed on the map. Due to time constraints it was not possible to further expand this by calculating an estimated population for each street and then using that to directly suggest locations or show a heat map of how good the implemented coverage should be.

The bowser status user story was implemented by allowing users to select a bowser on the same map and then either edit its information or assign a maintenance task to change its status.

The maintenance task priority was implemented by allowing the user to define the priority of a task when requesting it.

The code used for this was:

- councilbowseredit.blade.php
- Google Maps api
- bowsereditanimate.js
- BowserEditController.php
- TblBowserInformationModel.php
- TblMaintenanceTasks.php

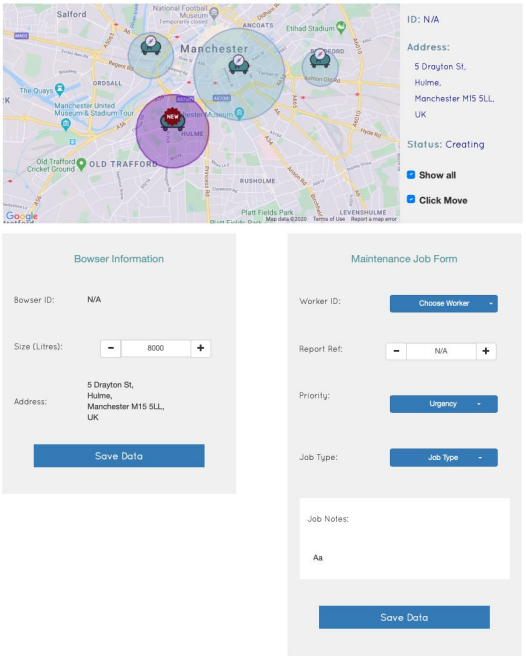


Figure US21

Maintenance Users

User Story	As a maintenance user I want to record completed tasks so that I can report back to the council.
Description of Done	Maintenance users will be able to assign and mark as complete jobs for themselves.
Acceptance Criteria	Maintenance users can assign themselves tasks. Maintenance users can un-assign themselves tasks. Maintenance users can mark as completed tasks assigned to them. Completed tasks will update the bowser status.
Story points	3

Figure US22

This user story was implemented by creating the taskList page, this page was given two tabs, the first shows all available tasks and lists them by distance from the user, the second tab then shows the details for a task and enables the user to assign, unassign or complete tasks.

The code used for this was:

- User.php
- TblCouncilMemberInformationModel.php
- TblMaintenanceTasks.php
- tblTaskType.php
- TblBowserInformationModel.php
- taskListController.php
- tblReportInfo.php
- taskListController.php
- taskList.blade.php
- tasks.js

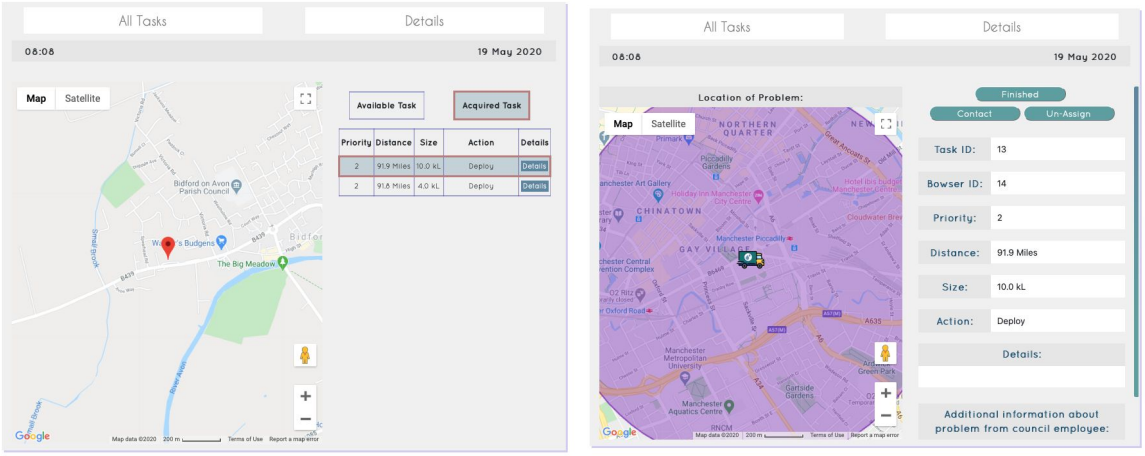


Figure US23

User Story	As a maintenance user I want to inform the council of unforeseen issues with bowsers or their location so that I can complete my job properly.
Description of Done	There will be a messaging system to facilitate communication between users.
Acceptance Criteria	There is a two-way messaging system available between users. Users can select another user and contact then and get a reply.
Story points	2

Figure US24

This user story was implemented by creating the messageDesktop page. The page has two columns, on the left the user can search for a person then on the right send messages, images and/or their location.

The code used for this was:

- messageDesktop.blade.php
- messageDesktopAnimate.js
- messageDesktopController.php
- user.php
- TblCouncilMemberInformationModel.php
- TblUserInformationModel.php
- TblConstituencyModel.php
- TblMessagingOverview.php
- TblMessages.php

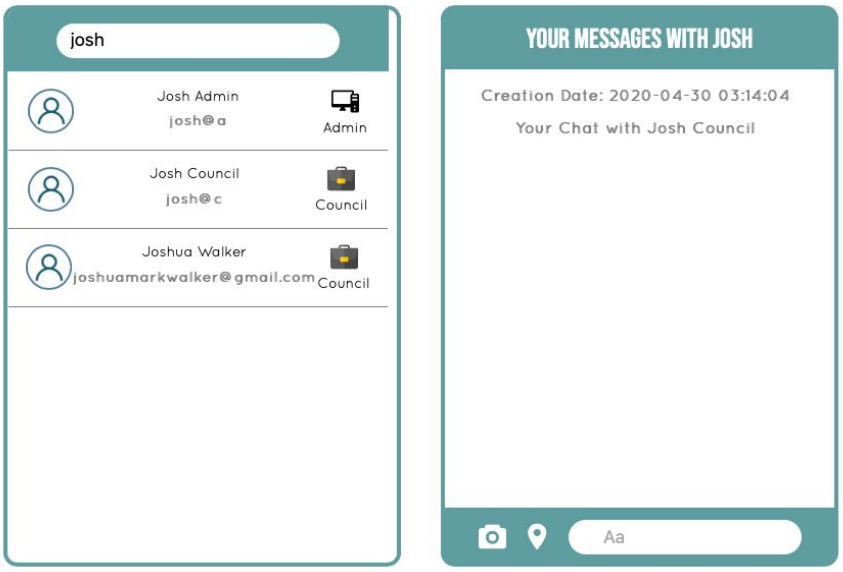


Figure US25

Admin Users

User Story	As a council administrator I want to be able to create users so that I can add users to my constituency.
Description of Done	Administrations are able to create accounts with the privileges appropriate for their jobs and specify their constituency.
Acceptance Criteria	Admins can create users defining the type and constituency of a user. Creating a user will send a welcome email to them allowing them to select a password.
Story points	5

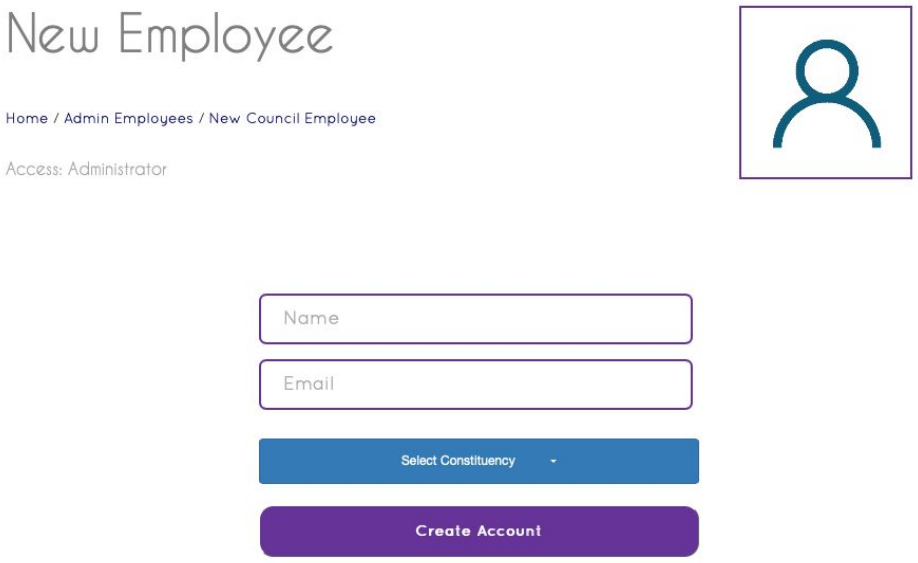
Figure US26

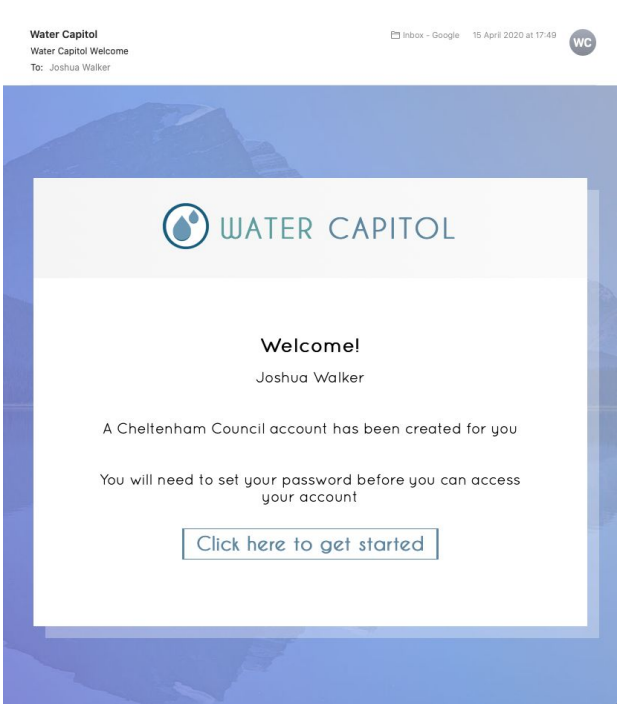
This user story was implemented by creating the adminEmployeeView page in which the administrator can define a new user and their constituency, with the links to this page being used to define the type of user as a GET variable. A welcome email was created that mimics the design of the website providing a link to set up a new account. Finally, the reset page was created to securely set the password for a user using a 32byte token.

The code used for this was:

- adminemployeeview.blade.php
- adminemployeeviewanimate.js

- AdminEmployeeViewController.php
- TblUserInformationModel.php
- User.php
- TblConstituencyModel.php
- mail.blade.php
- welcome.blade.php
- reset.blade.php
- reset.js
- ResetController.php
- https://api.pwnedpasswords.com/range/ api





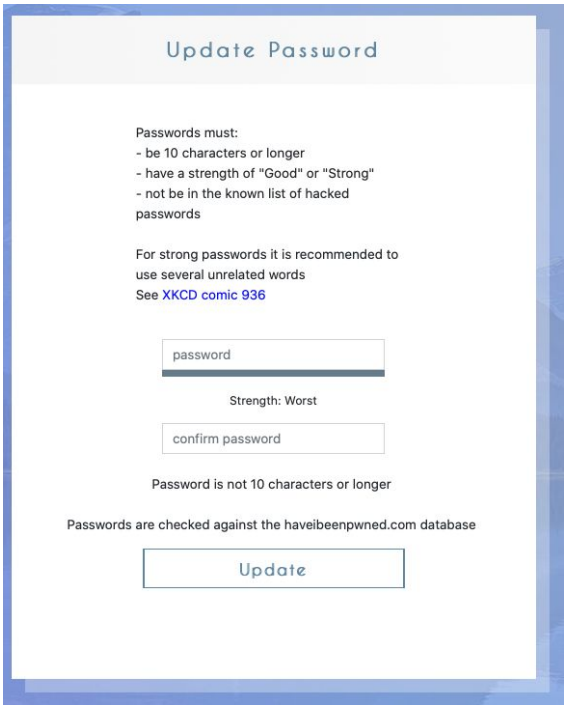


Figure US27

User Story	As council administrator I want to access user statistics so that I can monitor productivity.
Description of Done	There are user statistics available that show what each user has achieved.
Acceptance Criteria	There are statistics for overall user productivity. There are statistics available for each user. The statistics are only available to administrators.
Story points	8

Figure US28

This user story was implemented by creating both the adminHome page for statistics about the constituency and adminEmployeeView page which can show the statistics for an existing user.

The code used for this was:

- AdminHomeController.php
- adminhome.blade.php
- TblBowserInformationModel.php
- TblMaintenanceTasks.php
- AdminEmployeeViewController.php
- TblUserInformationModel.php
- TblConstituencyModel.php
- adminemployeeview.blade.php
- adminemployeeviewanimate.js
- User.php

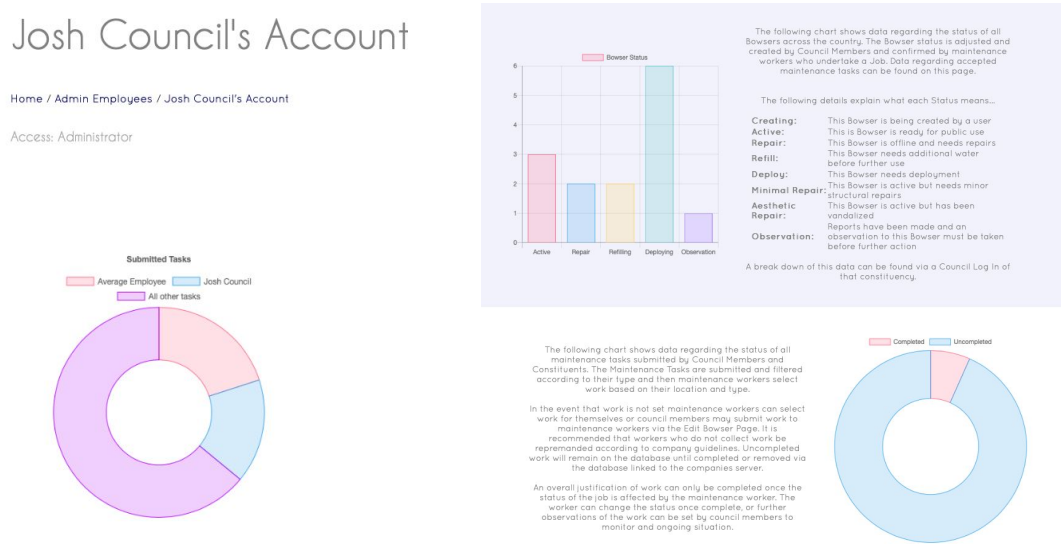


Figure US29

User Story	As council administrator I want to reset user passwords so that I can respond to users having forgotten their password or their accounts having been hacked.
Description of Done	Administrators can reset user accounts passwords.
Acceptance Criteria	Administrators can select to reset the password of a user. The users account will be inaccessible until a new password is chosen. Anyone currently accessing the account will be logged out. The reset will send an email to the user providing them access to set a new password.
Story points	3

Figure US30

This user story was implemented by adding a password reset button to the adminEmployeeView page and a password reset email that provides a link and token to reset their password using the reset page. Laravel Auth is then used to logout any instances of the user after their current password is overwritten with a random 32byte string.

The code used for this was:

- AdminEmployeeViewController.php
- TblUserInformationModel.php
- TblConstituencyModel.php
- adminemployeeview.blade.php
- adminemployeeviewanimate.js
- User.php
- mail.blade.php
- emails/reset.blade.php
- pages/reset.blade.php
- reset.js
- ResetController.php
- https://api.pwnedpasswords.com/range/ api

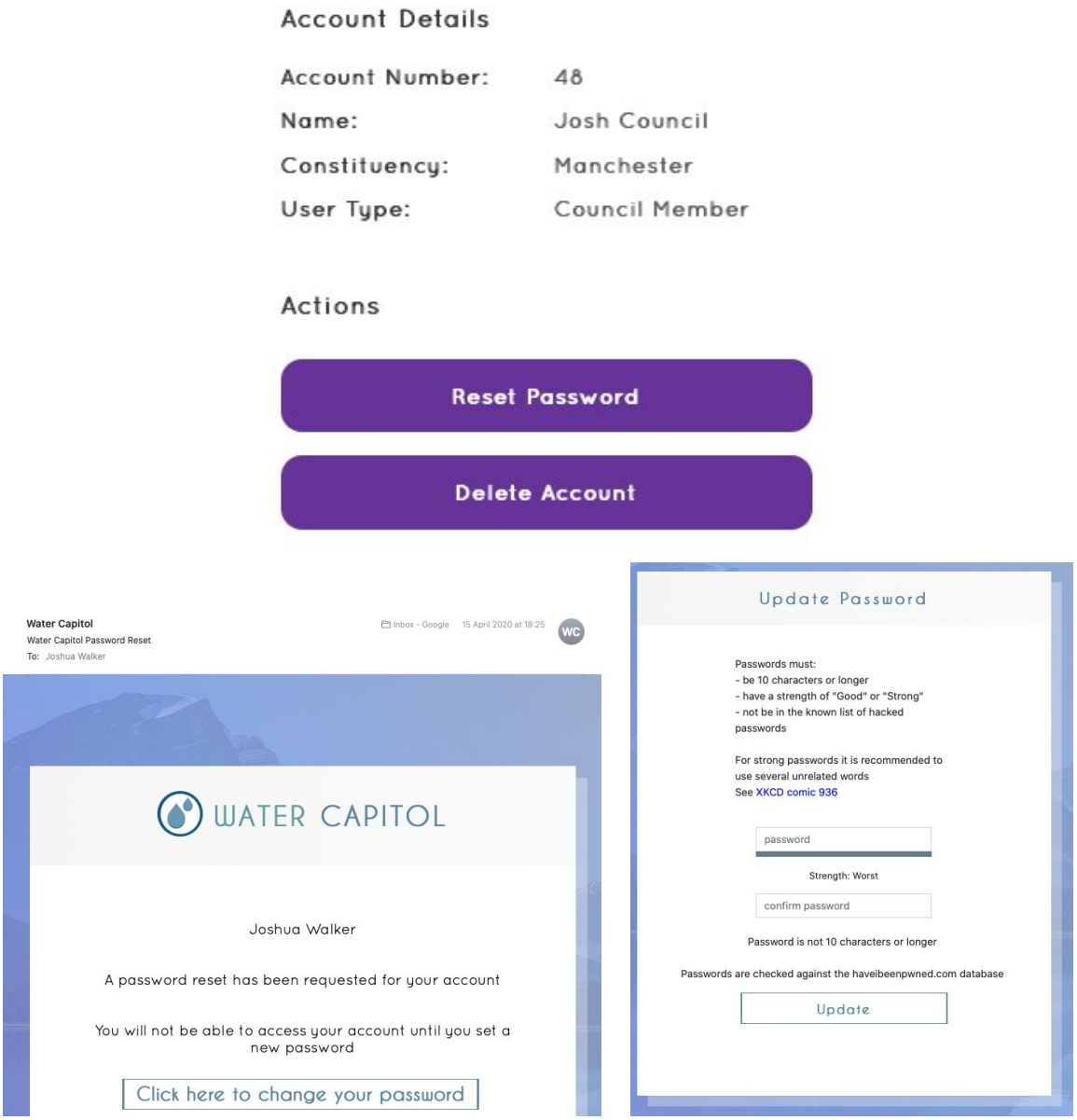

Account Details
Account Number: 48
Name: Josh Council
Constituency: Manchester
User Type: Council Member
Actions
Reset Password
Delete Account
Water Capitol
Water Capitol Password Reset
To: Joshua Walker
Inbox - Google 15 April 2020 at 18:25
WC
WATER CAPITOL
Joshua Walker
A password reset has been requested for your account
You will not be able to access your account until you set a new password
Click here to change your password
Update Password
Passwords must:
- be 10 characters or longer
- have a strength of "Good" or "Strong"
- not be in the known list of hacked passwords
For strong passwords it is recommended to use several unrelated words
See XKCD comic 936
password
Strength: Worst
confirm password
Password is not 10 characters or longer
Passwords are checked against the haveibeenpwned.com database
Update

Figure US31

Future Improvements

Some improvements for moving forward is that the team stays together for future WaterCapitol developments. This is key because the team is now used to working together breaking them up and putting them into different teams wouldn't be beneficial as they would need to learn and adapt to a new team. If the team were to stay together then they need to learn from the negatives and not let them impede the next project. Avoiding the same mistakes can help in reducing the possible errors and time to get the project completed and also learn to adapt quickly.

In future updates for security, Apache should be phased out to make the website more secure and prevent a slow loris vulnerability. If there is a disaster and the water bowsers are needed, members of the public will rely on accessing the site to work out when and where they should get their water. If a DoS attack is run, people will not be able to access the site and this can increase their levels of uncertainty and could result in people wasting their time by going to a bowser which is not working. Updating the website once it is deployed to run on NGINX instead will eliminate the vulnerability.

Additionally, the team would benefit from having more sprint retrospectives so that the team can have constant feedback to ensure the customers are satisfied with the project once it is completed.

Finally, the version of Laravel should be updated to 7.0 as this is the currently recommended version and the version on the server (4.2) is the oldest version currently supported and therefore likely to stop being supported soon, furthermore the newer version will enable better unit testing and improved security.

Conclusion

To conclude there has been a discussion about how business functions are a higher priority than business needs. By using functions, the website can execute and resolve the tasks layed out. Sprint backlogs are there to organise user stories and prioritize them. Talking about how agile has helped the team helps people understand the benefits of it can increase customer satisfaction as well as having constant customer feedback to ensure the product is what they want. Additionally added, is the sprint review minutes along with the sprint review meetings to show how the team progressed as a whole. Designs and layout discussions also allowed the team to decide on page designs and specifications. Overall, this project has met and tried to excel the requirements set to execute a usable system for managing bowsers throughout a crisis by using agile.

References

Almeshekah, M., Gutierrez, C., Atallah, M. and Spafford, E., 2015. ErsatzPasswords. *Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015*, [online] Available at: <https://dl.acm.org/doi/abs/10.1145/2818000.2818015> [Accessed 14 May 2020].

Bagui, S. and Earp, R., 2012. Database Design Using Entity-Relationship Diagrams. Boca Raton, FL: CRC Press.

Bourne, 1992. Object-Oriented Engineering: Building Engineering Systems Usig Smalltalk-80. CRC Press.

Boyd, S. and Keromytis, A., 2004. SQLrand: Preventing SQL Injection Attacks. Applied Cryptography and Network Security, [online] pp.292-302. Available at: <https://link.springer.com/chapter/10.1007/978-3-540-24852-1_21> [Accessed 14 May 2020].

Bruegge, B. (2004) Object-oriented software engineering : using uml, patterns, and java. 2nd edn. Upper Saddle River, NJ.: Pearson Education.

Carey, M. (2014) Developing quality technical information: a handbook for writers and editors, third edition. 3rd edn. Upper Saddle River, N.J.: IBM Press. Available at: INSERT-MISSING-URL (Accessed: May 14, 2020).

Fidao, C., 2018. *Raw Queries In Laravel*. [online] Fideloper.com. Available at: <https://fideloper.com/laravel-raw-queries> [Accessed 13 May 2020].

Grabovsky, S., Cika, P., Zeman, V., Clupek, V., Svehlak, M. and Klimes, J., 2018. Denial of Service Attack Generator in Apache JMeter. *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, [online] 10(1). Available at: <https://ieeexplore.ieee.org/abstract/document/8631212> [Accessed 14 May 2020].

Hatzivasilis, G., Papaefstathiou, I. and Manifavas, C., 2020. Password Hashing Competition - Survey and Benchmark. *Research Gate*, [online] pp.1-20. Available at: <https://www.researchgate.net/profile/George_Hatzivasilis/publication/273959978_Password_Hashing_Competition_-_Survey_and_Benchmark/links/55116d2a0cf29a3bb71dd066.pdf> [Accessed 14 May 2020].

Hunt, T., 2020. *HIBP Home*. [online] HIBP. Available at: <https://haveibeenpwned.com> [Accessed 18 May 2020].

Kumar, H., Kumar, S., Joseph, R., Kumar, D., Shrinarayan Singh, S., Kumar, P. and Kumar, H., 2013. Rainbow table to crack password using MD5 hashing algorithm. *2013 IEEE CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGIES*, [online] Available at: <https://ieeexplore.ieee.org/abstract/document/6558135> [Accessed 14 May 2020].

Laravel. 2020a. Database: Getting Started - Laravel - The PHP Framework For Web Artisans. [online] Available at: <https://laravel.com/docs/7.x/database> [Accessed 14 May 2020].

Laravel. 2020b. Routing - Laravel - The PHP Framework For Web Artisans. [online] Available at: <https://laravel.com/docs/7.x/routing#route-parameters> [Accessed 14 May 2020].

Laravel Team, 2018. *Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans*. [online] Laravel.com. Available at: <https://laravel.com/docs/7.x/eloquent> [Accessed 9 May 2020].

Lethbridge, T. C. and Laganière Robert (2005) Object-oriented software engineering : practical software development using uml and java. 2. edn. London: McGraw-Hill.

Lucid Chart, 2020. *Lucid Chart*. [online] Lucidchart.com. Available at: <https://www.lucidchart.com > [Accessed 15 May 2020].

Menasce, D., 2003. Web server software architectures. *IEEE Internet Computing*, [online] 7(6), pp.78-81. Available at: <https://ieeexplore.ieee.org/abstract/document/1250588> [Accessed 14 May 2020].

Mohamedelgendy.com. 2020. Business Needs Vs. Requirements | Moelgendy Blog. [online] Available at: <http://mohamedelgendy.com/blog/business-needs-vs-requirements.html> [Accessed 14 May 2020].

Porzio, C., 2020. *Extending Models In Eloquent*. [online] Tighten. Available at: <https://tighten.co/blog/extending-models-in-eloquent/> [Accessed 9 May 2020].

Shorey, T. (2018). Performance Comparison and Analysis of Slowloris, GoldenEye and Xerxes DDoS Attack Tools. *IEE*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8554590 [Accessed 10 Dec. 2019].

Silverston, L. (2001) The data model resource book : a library of universal data models for all enterprises. Rev. edn. New York: Wiley. Available at: INSERT-MISSING-URL (Accessed: May 14, 2020).

Stadler, M., Vallon, R., Pazderka, M. and Grechenig, T., 2019. Agile Distributed Software Development in Nine Central European Teams: Challenges, Benefits, and Recommendations. International Journal of Computer Science & Information Technology (IJCSIT) Vol, 11.

Srivastava, A., Bhardwaj, S. and Saraswat, S., 2017. SCRUM model for agile methodology. In: 2017 International Conference on Computing, Communication and Automation (ICCCA). [online] Greater Noida, India:IEEE, pp.864-869. Available at: <https://ieeexplore-ieee-org.glos.idm.oclc.org/document/8229928> [Accessed 25 April 2020].

Sumathi, S. and Esakkirajan, S., 2007. Springer Science & Business Media.

Sun Technical Publications (2010) Read me first! : a style guide for the computer industry. 3rd edn. Upper Saddle River, N.J.: Prentice Hall.

Tahaghoghi, S. and Williams, H., 2007. Learning Mysql. Beijing: O'Reilly.

Waldock, B., 2015. Being Agile In Business : Discover Faster, Smarter, Leaner Ways To Work.

Winand, M., n.d. *SQL Bind Variables/Parameters In Databases*. [online] Use-the-index-luke.com. Available at: <https://use-the-index-luke.com/sql/where-clause/bind-parameters> [Accessed 14 May 2020].

Appendix 1 User manual

Four User Manuals have been developed for this report. Each user manual will cater and has been designed for the platforms core users, Public, Council, Maintenance, Administrators. The manuals were designed to be both visually appealing and practical in use. Each section is broken down into clear headings that then follow with text consisting of little jargon, humour and ambiguity (Sun Technical Publications, 2010, pp.58, 90, 120). Furthermore, each section has questions that the user is likely to ask on the corresponding web-page, this means that an answer to each question can be given directly and with accuracy (Carey, 2014, p.27; Sun Technical Publications, 2010, p.90). The design and layout for the site was implemented to best reflect the authorised pages and colour schemes that the user will see (Carey, 2014, p.261).

The user manuals can be located within the project folder.

Appendix 2 - Personal Statements

Luke Gassmann

I have felt that I have contributed my best work to this team's project. Over the course of development our team worked very well together with trust and collaboration. As with all teams, there were obstacles and difficulties along the way, however, I feel that I would be encouraged to use Agile again, especially if it meant working with this team again.

Contribution

I should state that I feel that all team-members pulled their weight throughout this project, at any point when a team member was stuck, another team member was always happy to guide and help for the benefit of the team. My contribution as a developer was developing several council and administrator pages that would help move the project forward to reach and occasionally surpass the minimal viable product. This was in no part only my work as several other members also helped design the database and model files that would help run this, but I always tried to complete as many effort-points or user stories as I could each week.

The following pages I developed were:

- Front-end of the Log in Page
- Admin Statistics Page
- Admin Employees Page
- Admin Employee View Page

- Admin Constituencies Page
- Discussions Page
- Council Home Page
- Council Edit Bowser Page

In the earlier stages of development, Josh and I developed the platforms layouts which would span across the website. The team's bond strengthened over time, which soon led to paired programming over the internet/skype. It would often be the case that myself, Nicolas or Joe, would pair program during the week if one of us had a bug in the development process. Josh was a reliable source of programming knowledge and the two of us worked on the early stages of understanding and implementing Laravel onto the server and into our code. Sophie as our scrum leader was always a reliable communication source to customers from the development team, in addition to writing documentation, recording minutes and setting us on track at the beginning of each sprint.

I also enjoyed presenting and talking to our products customers. This allowed me to create the presentation video for the second scrum meeting:

Main Sprint Update Video

https://youtu.be/TRnfUvBpUb4

Smaller Week Sprint Update Video

https://youtu.be/Ty0U05sro6c

Adapting to the team

My first response to group work was apprehension. I had previously worked in groups which idled and meant that work was placed on me to complete. This was why when selecting which team to be in, Sophie and I had discussed beforehand who we wanted in our team to provide the best working environment for everyone. Through self-organization we managed to get the team we had initially wanted, unfortunately this did involve heated discussions with friends that were not considered. We merged very well and our velocity increased week on week as we began to trust and work with each other more. Because of this, converging to distributed teams in the middle of the term had little to no effect on our team's morale.

Sophie Jones

During the agile project, I took the role of the scrum master with elements of a product owner. As a team we went through the strengths and weaknesses of the group and since I am not a strong programmer I thought the best way to help the team was to do the paperwork side of the project.

What I started with doing are the user stories. Nicolas wrote them down but as the first sprint started I took it the opportunity to sort out the definition of done and the acceptance criteria on the rest and get them uploaded on to Jira. I had input from the group when I wasn't sure about them, but I used common sense for most.

Alongside this, I made the content for all the webpages since it was only fair since I couldn't code. I also wrote up the minutes from each meeting alongside holding the sprint meetings and facilitating them and ensuring they were on track. I also started the documentation since I am not technical. I wrote about how agile the additional benefit to the project.

When Coronavirus started impacting our lives, mine got busier due to my job. My job required me more since I wasn't hugely affected due to the fact I didn't have children, so my job took up most of my time. Due to this, I had to take a back seat which wasn't fair on the others since I had to miss some sprint meetings since I was working. When I had a chance I always added into our WhatsApp group what I did so I could still be part of the team.

I feel like I could have been more there to help them, and I would have been if the circumstances didn't change so drastically. However, when I was there I was a team player and did listen to the others and tried to support them where I could.

There were a couple of bumps in the road towards the end of the assignment, but we worked through them and discussed them and then proceeded to work as a team. I have learned for future projects is to be open to trying new things such as learning to read code and comment on it or try testing the code. Also, I learned to be so headstrong because it is a team effort and I need to learn to trust my peers.

During the progress of this development, I felt like I wasn't doing enough since I couldn't help them but when looking back on it there wasn't much for me to do since it was all technical but I feel like by me doing most of the documentation would be a great way to compensate. I felt like I was an asset to the team since I was there to facilitate team meetings and keep the meetings on track as the team can get quite distracted. I believe I did as much as I could do during this project.

Nicolas Euliarte Veliez

Throughout the development my understanding of Laravel and PHP has expanded. The work I contributed has been completed to the best of my ability. I feel that throughout the construction of the development I have made good connections with my teammates and this has enhanced our overall construction of the project.

Through the start of the project, pair programming was done between Joe and I due to our lack of understanding of Laravel and my personal knowledge of HTML, CSS and JavaScript. PHP and Laravel were the two weak points between us and therefore the pair programming have contributed towards the successful execution of sections of the application. Through the positive effect pair programming has had between us, the execution of the pages and most database tables have been done efficiently, giving both of us the ability to also focus on other aspects of the project alongside the execution of other projects.

During the development of the project, I have contributed to various aspects of the site. As Luke and Josh are the most experienced in web development out of the group, they helped Joe and I significantly in areas we felt less confident in given the circumstances in both PHP and Laravel. I have contributed throughout various pages, controllers, models and blades within the page. The main pages allocated to me where:

- Home

- Frequently Asked Questions
- Public Requests
- Bowsers Near Me – With Joe
- Council Public Inquiries – With Joe

As visible above, certain pages have been added to by others and pair programming has been done on certain pages with Joe. Since Covid-19, Joe and I have continued to have meetings to execute pair programming through Teams and Skype. In addition to this, I have had some meetings with Luke and Josh where we pair programmed on other aspects of the website. This worked well with the team, especially with Josh given the ability to face time and even control our Macs remotely when bugs needed fixing. Through the communication and pair programming the stress of the situation was eased given the constant assurance of others being in the same situation and being able to communicate efficiently nonetheless.

At certain points in development personalities did clash due to stress, but due to the current situation the team has carried on strongly and the execution of the site was completed. Sophie executed the role of scrum master through her extensive communication with Abu. From this we had been given advantages to execute the development with more ease and understanding.

In the report, I have written the introduction, aspects of the project, layout, design and implementation and the navigation sections. In addition to this Luke and I have gone through the entire report to format and proofread everything.

Overall, I believe that the whole team has given a good contribution to the development and design of the website. I personally feel as if the development team was not at the level of understanding of the application as us we would not have been able to accomplish the amount we have produced.

Joseph Lloyd

The work contributed to the team has been done to the best of my ability despite not being a knowledgeable website developer. Going into this project, I had never used PHP, and laravel requires some prior knowledge. I feel as though the team members and I have worked extremely well together and there were no upsets during the length of the product.

During the first couple of weeks, Nicolas and I spent the majority of the time trying to get up to speed with PHP and Laravel as we both had very little experience. This took a little longer than expected but during that time we also built the majority of tables needed for the database although some were added later by Josh and Luke when developing some of the pages and realising an extra table would be needed as well as appending the old ones.

During development, I contributed to pages in minor ways including the controllers and blades, some of the database model’s functionality and bug fixes at various times but the

main pages I contributed to were:

- Contact Us
- Single Maintenance Task (Incorporated into the task list page)
- Bowsers Near Me (With Nicolas)
- Council Inquiry (With Nicolas)

Several of these were completed pair programming with Nicolas. After the coronavirus lockdown was announced, Nicolas and I decided to call weekly on Skype or Teams to work together on a page. This was done in a pair programming style where one of us would watch and the other would code and share their screen so the other person could see. This was done in an attempt to further both of our knowledge about the language and to help each other find more effective solutions to problems we may encounter. This made the experience less stressful overall and I feel as though it was a very effective strategy that helped us both learn more than we would have individually.

Throughout the process we also had Josh and Luke helping us when we were struggling which came in useful a couple of times to give us an outline of alternate ways a problem could be solved and also for teaching us the basics of Laravel and filling in gaps in my HTML5 knowledge in general. I have learned a lot from them as well as Nicolas during this project and would like to think they learned from me as well.

I think Sophie was the right person for the scrum master role. I considered her the pseudo-product owner as well because she was the main means of communication between us on the development team and Abu, the product owner. This meant she was able to keep us on track during meetings where we would often go off on tangents.

For this report, I also wrote the SQL where the tables use and eloquent were explained and the Security section where the measures taken to protect the website and its users were outlined as well as a known vulnerability also talked about in the Future Improvements section.

Overall, I think all team members made valuable contributions to the team and I feel as though the project would not have been completed without them.

Joshua Walker

Going into this project I looked forward to working as part of an agile team because while I have experience programming professionally, the work I do is mostly solo programming. From my experiences of this team I would be interested in doing more agile team work again, as I felt the team worked very well together in this agile project despite the unusual circumstances. Starting this project, I also looked forward to learning new frameworks such as Laravel as it would help to further broaden my skills, and having done so I look forward to taking any future project with Laravel even further to try its more esoteric features.

Having already known each other from the first half of the year and in some cases longer we worked very well together from the start, with our biggest issue initially being that we would drift off topic in our discussions, though thanks to Sophie we normally quickly got back on topic.

While it is normally easy enough to work fluidly as a team if you are working with each other every day face to face, it is much more difficult to continue doing so when limited to remote communication, therefore, I was very pleased to see how well as a team we managed continue to work when the lockdown occurred. With weekly meetings and other semi-regular calls to discuss specifics we managed to find a way that we could remain in sufficient contact to work together cohesively, and while there were one or two minor disagreements, further communication and planning together as a team helped to resolve any issues.

At the start of the project while Sophie documented the agile side and Nicolas and Joe worked on setting up the database, Luke and I worked on understanding how to use Laravel. With Laravel having so many parts working with Luke was enormously helpful as he was able to provide observations and insights that would have taken me much longer to understand, following on from this we then worked together to develop the layout template files that all webpages were then based around.

Following on from this each of the programmers in the team started to allocate themselves pages, and as such the code I in particular worked on was:

- Password Reset Page
- Task List Page – With Joe
- Welcome, Reset and Reply Emails
- 404 page
- Updating Joe & Nicolas', Bowsers Near Me & Inquiries Pages based on client feedback
- Laravel Authentication
- Unit Testing

I also worked occasionally with the other programmers in a technical support role to help come up with solutions to both technical and logical programming problems either over the phone, asynchronously or by remotely accessing their computer.

Overall, I feel the team worked well together and I was able to provide valuable contributions to both it and the website.

Appendix 3

Sprint review minutes- Sophie

Week 1:

- Got the team organised: Sophie, Luke, Joe, Josh and Nicolas
- Set team expectations: to achieve high grades, Clear communication, Issues need to be voice and resolved quickly
- Started on user stories
- Agreed which users' stories which are: Maintenance, Council and the public

Week 2:

- Spoke about what needs to be done: acceptance criteria, planning poker, product backlog, starting to put everything on Jira, refine user stories, breakdown epics.
- Talked about colour scheme
- Layout: talked about the ideas agree to split the page design
- Name of website
- Pages required: decided on pages needed
- Started sprint: Sophie: finishing user stories with definition of done and acceptance criteria. Joe, Nicolas, Josh, Luke design the pages

Week 3:

- Agreed the pages
- Talked about starting the whole programming of the website and the database
- Agreed next sprint: Sophie: Start the content for the pages, Joe and Nicolas start sorting the database with primary keys and foreign keys. Josh and Luke start coding the website

Week4:

- Daily stand up alongside with sprint retrospective
- Discussed questions to add in FAQ page
- Show work to each other to see if its ok
- Discussed how to link website with database

Week 5:

- Agreed to lighten load as other assignments
- Group input on FAQ questions

- Luke and Josh starting to code more on the webpages
- Sophie start writing the minutes up from all the meetings

Week 6:

- Luke did homepage
- Josh did Laravel with password page
- Sophie caught up with all mins from previous meetings
- Get ready for sprint review meeting
- Sophie writes up points to discuss with the Powerpoint and ERD.
- Luke making the website full compatible on mobile and universal style sheet
- Joe doing log in page and FAQ page content
- Nicolas helping joe
- Josh doing the drawings, adding content to pages and sort documentation page

Week 7:

- Sprint review with stakeholders
- They wanted encryption for passwords which is high priority
- They are not fussed on the twitter hashtag- low priority
- They want override on distribution of where the bowsers are located
- This done by GPS location
- They want priority of repairs done by the system
- On public inquires instead of emailing the reporter just put out a public update on the website
- Change the welcome button to a login button
- Luke- finish council pages
- Nicolas- finish homepage
- Joe- maintenance task list
- Josh- maintenance page list
- Sophie- Look at code comments and start looking at the documentation

Week 8:

- On skype
- Sophie- started the documentation
- Luke- did council homepage, water bottle gifs for refill, maintenance page
- Joe- layout concept for single task pages needs to add google API
- Josh- debugging maintenance task list page
- Nicolas- sorted out the homepage
- This week’s sprint is to refine all 4 pages
- Joe and Luke - Connecting database to website
- Nicolas- Sort priorities out for maintenance
- Josh- adding map to pages and sorting out encryption

- Sophie- skeleton documentation

Week 9:

- Joe- Pulling information from database issues
- Luke- focus view or general view on bowsers and assisted into getting information pulled from the database
- Josh- encrypted passwords stored in database
- Nicolas- finished single task public page
- Sophie- documentation
- Nest sprint:
- Joe- finish connecting database and pages
- Luke- internal messaging and recommended bowser distribution
- Josh- adding admin password changes and email
- Sophie- finalise skeleton documentation
- Nicolas- sort out blade files and public request page

Week 10:

- Joe- sorted the database connections
- Luke- started on internal messaging and recommended bowser distribution
- Josh- sorted out admin changes and the emails for password changes
- Nicolas- sorted out blade files and started to finalise public request page
- Sophie- finalised skeleton documentation
- next sprint:
- Luke- admin page with statistics, statistics for bowsers and admins being able to add and delete users
- Josh- maintenance list page, map and lists done, finalising emails
- Joe- acquire task button working, twitter hashtag working
- Nicolas- add map on main page and public request page and priority list
- Sophie- start writing documentation

Week 11:

- Luke- admin page with statistics, statistics for bowsers and admins being able to add and delete users done
- Josh- maintenance list page, map and lists done, finalised emails
- Joe- task button working, twitter hashtag working
- Nicolas- added map on main page and public request page and priority list
- Sophie- start writing documentation
- Next sprint:
- Josh- connecting email bits and sorting out the log in page
- Luke- Video for sprint review
- Sophie- split the documentation for everyone
- Joe- create uniform webpages and finalise hashtags

- Nicolas- finish off pages allocated to him

Week 12:

- Josh, Luke, Nicolas, Joe- finish programming ready for sprint review meeting
- Nicolas and joe- done work on enquiries page with longitude and latitude
- Josh- finished finer details on his pages
- Luke- doing basic CSS and making more accurate burndown charts
- Sophie- started on documentation and looking at putting on google docs so it is easier for everyone to work on
- Next sprint:
- Joe- finish page and fixing bugs to get it sorted
- Nicolas- finish page and fixing bugs
- Josh- planning to get all pages linked and allow multiple sessions at once
- Luke- planning to make changes to the CSS and begin some documentation
- Sophie- sort out PowerPoint for next sprint review and carry on with documentation

Week 13:

- Sprint review 2:
- Likes how clear the whole website is
- Slide pictures look clunky
- Likes how it asks for location
- Adding images of bowsers so it is clearer what the purpose is
- Contact us page needs text to show
- The icons on the bowsers page to actual bowsers
- API on that page should be half and half
- Need there's to be a key so people know the status of each bowser
- On this page if there is an issue there is a button to go straight to the enquires page with the bowser automatically filled in.
- The request page: a definition of what the repair status means
- Change log in button to possibly in the footer so the public don't see it
- Move contact us and FAQ page to the end of the navigation
- Admin side:
- Needs an option to add different constituencies instead of going through the SQL database
- Council homepage:
- Colour code repairs on list to break it down
- Change date format
- Bowser edit page:
- Add a new bowser tab instead of searching
- Hide latitude and longitude
- Task page:
- Filtering system so the same repair isn't reported twice
- Add notification so that user knows it's been reported
- Maintenance add a key for what tasks have been assigned.

- To complete by next week's sprint:
- Sophie: To carry on with documentation
- Luke: change the background as to the request above, move the log in, change the navigation bar as to the requests, change images on the bowser, removing long and lat from the bowsers page and start the user manual
- Josh: do the half and half as per to the requests, users only showing for which constituency they are in, change the date format, sort out the FAQ page and contact us page, modify the links, add keys to the requests and to sort out how to only have one request and not many of the same
- Nicolas and Joe: change items on the requests page to make it easier to understand, change the text on contact us page

Week 14:

- Done this week:
- Sophie: To carry on with documentation
- Luke: change the background as to the request above, move the log in, change the navigation bar as to the requests, change images on the bowser, removing long and lat from the bowsers page and start the user manual
- Josh: do the half and half as per to the requests, users only showing for which constituency they are in, change the date format, sort out the FAQ page and contact us page, modify the links, add keys to the requests and to sort out how to only have one request and not many of the same
- Nicolas and Joe: change items on the requests page to make it easier to understand, change the text on contact us page
- Next week:
- All to start on documentation
- Finalise code commenting

Week 15:

- Sophie: working on documentation
- Luke: did user manuals and data models
- Joe: started documentation and done code comment
- Nicolas: done documentation and code commenting
- Josh: unit testing and started documentation
- Next week:
- The group to finish documentation