|  |
| --- |
| Module CI869 |
| Constructed Wetlands Website |
| Software Design and Development |

Contents

[1 Introduction 4](#_Toc416705079)

[1.1 Purpose 4](#_Toc416705080)

[1.2 Participants 4](#_Toc416705081)

[1.3 Timeline 5](#_Toc416705082)

[1.4 Report Outline 5](#_Toc416705083)

[2. Project Definition 6](#_Toc416705084)

[2.1 Initial Brief 6](#_Toc416705085)

[2.2 Business Driver from Project 6](#_Toc416705086)

[2.3 Glossary 7](#_Toc416705087)

[2.4 Problem Domain 7](#_Toc416705088)

[2.5 Research into Wetland Data 9](#_Toc416705089)

[2.6 Project Requirements 11](#_Toc416705090)

[3. Project Management 12](#_Toc416705091)

[3.1 Programming and Responsibilities 12](#_Toc416705092)

[3.2 Risk Analysis 13](#_Toc416705093)

[4. System Analysis and Design 14](#_Toc416705094)

[4.1 Research of similar systems 14](#_Toc416705095)

[4.2 Requirements 14](#_Toc416705096)

[4.3 Business Rules 14](#_Toc416705097)

[4.4 Use cases 15](#_Toc416705098)

[4.5 Data Modelling 16](#_Toc416705099)

[4.6 Interaction Modelling 16](#_Toc416705100)

[4.7 Project Architecture 17](#_Toc416705101)

[4.8 Selecting Technologies and Tools 17](#_Toc416705102)

[5. Implementation 20](#_Toc416705103)

[5.1 Database Structure 20](#_Toc416705104)

[5.2 Code Structure 20](#_Toc416705105)

[5.3 Components 22](#_Toc416705106)

[Bootstrap 22](#_Toc416705107)

[5.4 Data Persistence 24](#_Toc416705108)

[5.5 Security 25](#_Toc416705109)

[Cross-Site Request Forgery (CSRF) 25](#_Toc416705110)

[Password Protection 26](#_Toc416705111)

[Registering 26](#_Toc416705112)

[Validation 27](#_Toc416705113)

[SQL Injection 31](#_Toc416705114)

[5.6 Website Structure 31](#_Toc416705115)

[Sample page: 31](#_Toc416705116)

[Browse Literature: 35](#_Toc416705117)

[Forgotten Password: 36](#_Toc416705118)

[Uploading files: 36](#_Toc416705119)

[5.7 Phased development, next step 37](#_Toc416705120)

[6. Testing 37](#_Toc416705121)

[6.1 Unit Testing 37](#_Toc416705122)

[6.2 System Testing 37](#_Toc416705123)

[6.3 Interface User Experience Testing 37](#_Toc416705124)

[7. Conclusion 38](#_Toc416705125)

[7.1 Unit Testing 38](#_Toc416705126)

[7.2 System Testing 38](#_Toc416705127)

[7.3 Interface User Experience Testing 38](#_Toc416705128)

[References 38](#_Toc416705129)

[APPENDIX A Preparing for Interviews 39](#_Toc416705130)

# 1 **Introduction**

This document contains the requirements analysis and structural modelling for the Wetlands Database Management System being proposed for the storing site sample data and providing a website so that sample data can be made available online.

## Purpose

Create a constructed wetland performance database which would allow users to query the database to find and download data that matches particular search criteria.

Provide a user-friendly interface to allow interested parties to 'interrogate' the data (e.g. the end user may wish to examine the performance of wetlands, in a particular size category, in treating a specific water quality parameter, for example).

Increase awareness of the work done by The Geo-Environmental Engineering (GENE) research group, based at NUI Galway.

* Examine the material that needs to be modelled for the system. Gain an understanding of the problem domain and evolve a clear perspective on the essential requirements.
* Work with our client to develop a solution that is effective and achievable with in our limits of time and resources.
* Develop and practice group working skills that support good software development practices and encourage productivity from each team member.
* Maintain regular communications with team members, supervisor and client. Providing updates on progress.
* Produce a working demo model for the presentation.

## Participants

The development team throughout this project, 3 student from the HDip Software Design and development Course:

Ciaran Severn

Catherine Gaughan-Smith

Sean Lydon

Senior Technical advisor : Joe O’Connell

Project supervisor : Josephine Griffith.  
  
Module co-ordinator : Finlay Smith.  
  
Stakeholders:

|  |  |
| --- | --- |
| Owner | Dr. **Mark G. Healy** (room ENG-1038).  BE, MEngSc, PhD, Eur Ing, CEng, FIEI, Chartered Engineer,  Senior Lecturer in Civil Engineering |
| Administrator /  Wetlands Researcher | **Collette J Mulkeen** College of Engineering and Informatics |
| Expert User | Dr. Brian Donlon, EPA Research Manager ,  Environmental Protection Agency |

## Timeline

The key dates for the project are as follows:

Project Allocation – October 2014

First Presentation – November 2014

Phase 1 Report – November 2014

Project Demo – March 2015

Final Project Report – March 2015

## Report Outline

{Todo}

# **Project Definition**

## Initial Brief

A recent EPA-funded study is seeking to address this knowledge gap.

As part of the study, we are collecting performance data from wetlands in Ireland. However, we need a user‐friendly interface to allow interested parties to ‘interrogate’ the data (e.g. the end user may wish to examine the

performance of wetlands, in a particular size category, in treating a specific water quality parameter, for example).

No such database exists in Europe, and this would be a unique opportunity to advance the state-of-the-art in wetland analysis.

What we require:

• A constructed wetland performance database similar to an existing one in the USA (http://firehole.humboldt.edu/wetland/twdb.html), which would allow users to query the database to find and download data that matches particular search criteria.

• The data would be returned to the end user in tabular format.

• The database would need to be designed such that it may be continuously updated by the research group. Ideally, it may also be possible for County Council technicians to upload data onto the portal, using a login password.

• Ideally this database would be housed on our research group’s webpage

(http://www.nuigalway.ie/gene/)

## Business Driver from Project

The Geo-Environmental Engineering (GENE) research group, based at NUI Galway, investigates the fate of **phosphorus** and **nitrogen in soil-plant-water systems** and the effects of agricultural management on soil and water quality. GENE has helped develop decision-making tools to target management alternatives and remedial measures that have reduced the risk of nutrient loss from farms. Currently, GENE’s main research interests are:

(1)   resource recovery and resource efficacy – use of materials for wastewater and water treatment; waste materials and chemicals as phosphorus and nitrogen adsorbents and soil amendments; waste as energy sources and slow-release fertilisers

(2)   forestry – protection of water courses from forestry activities, greenhouse gas emissions

(3)   soil physics – modelling of water movement through soil and management impacts for farmers



## Glossary

|  |  |
| --- | --- |
| **EPA** | **Environmental Protection Agency** |
| Flow rate | Total effluent volume discharged over the 24-hour period in which the composite sample is collected shall be recorded. |
| BOD | BOD 5 day Biochemical Oxygen Demand (without nitrification suppression). |
| cBOD | 5 day Carbonaceous Biochemical Oxygen Demand (with nitrification  suppression). |
| COD | Chemical Oxygen Demand |
| DO | Dissolved Oxygen |
| **RBC** | **Rotating Biological** **Contactor,** for example in Hollymount, Fenagh and Newtowngore. |

## Problem Domain

|  |
| --- |
| **Fig. 18.11A** Raw sewage moves from the grit chamber to primary treatment, where sludge is removed and the clarified water then proceeds to secondary treatment (here shown as activated sludge treatment).    From <<http://apesnature.homestead.com/chapter17.html>> |
| **Fig. 18.11B** In primary treatment sludge is removed and the clarified water then proceeds to secondary treatment. Raw sewage moves from the grit chamber to primary treatment, where sludge is removed and the clarified water then proceeds to secondary treatment.    From <<http://apesnature.homestead.com/chapter17.html>> |
| **Fig. 18.11C** Raw sewage moves from the grit chamber to primary treatment, where sludge is removed and the clarified water then proceeds to secondary treatment (here shown as activated sludge treatment).    From <<http://apesnature.homestead.com/chapter17.html>> |
| **Fig. 18.13** The secondary treatment, activated sludge process may be modified to remove nitrogen and phosphate while at the same time breaking down organic matter.    From <<http://apesnature.homestead.com/chapter17.html>> |

## Research into Wetland Data

|  |  |  |  |
| --- | --- | --- | --- |
| Wetlands | Identification: Wetland name (for example - Moycullen, Galway) Location: GPS coordinates,  Type: Bog, fens, cutover and cutaway bog areas.   |  |  | | --- | --- | | Wetland location   * latitude and longitude * city * state * EPA region * EPA facility ID * USGS watershed ID | Wetland owner, operator, designer, and regulator contact information   * name * addresses * phone numbers * email * web page entries |  * Design characteristics descriptors   + include population of the service community   + scale of the operation (bench, pilot, or full-scale)   + source of wastewater   + period and type of hydraulic loading   + type of pre-treatment   + type of pre disinfection   + objectives and beneficial uses of the wetland system   + dates of start-up and full operation   + capital costs   + O/M costs   + hydraulic type and loading rate   + total wetland footprint area |
| Site Source Type | Municipal,  Agricultural,  Industrial |
| Types of Pre-treatment | Primary,  Secondary |
| sample points | The sample points data table contains a description of all sample point locations where data is collected for the wetland systems included in the database. Each entry in this table contains   * identification number of the associated wetland * name and description of the point * cell in the wetland (if any) where the sample point is located * indication whether this sample point is a treatment system influent or effluent point |
| constituent measured | The samples data table contains the actual measured wetland descriptive or performance data. Each entry in this table contains   * identification number of the associated wetland * date the data was collected * identification number of the constituent measured * observed value of the constituent or sample variable * text comment associated with the sample |
| Meteorological characteristics | * monthly average precipitation * monthly minimum temperature * monthly maximum temperature * monthly evapotranspiration * annual average temperature, precipitation, and evapotranspiration * average first day of ice cover * average number of ice cover days |

## Project Requirements

#### Negotiate Scope

{Discuss}

#### Identify Tasks

{Discuss}

#### Estimate Task Durations

{Discuss}

#### Specify Inter-task Dependencies

{Discuss}

#### Assign Resources

{Discuss}

#### Direct the Team Effort

{Discuss}

#### Assess Project Results and Experiences

{Discuss}

# **Project Management**

This document

## Programming and Responsibilities

Our approach to managing the project development was to break it down into components. Through out the development we re-evaluated which components were more essential and then assigned the development of components to each team member. When the work was completed and tested, we went through a process if merging changes into the repository and then did some integration testing the confirm that the components still worked individually and also as part of the system.

The Product Breakdown Structure (PBS) is a hierarchical structure of things that the project will make or outcomes that it will deliver.

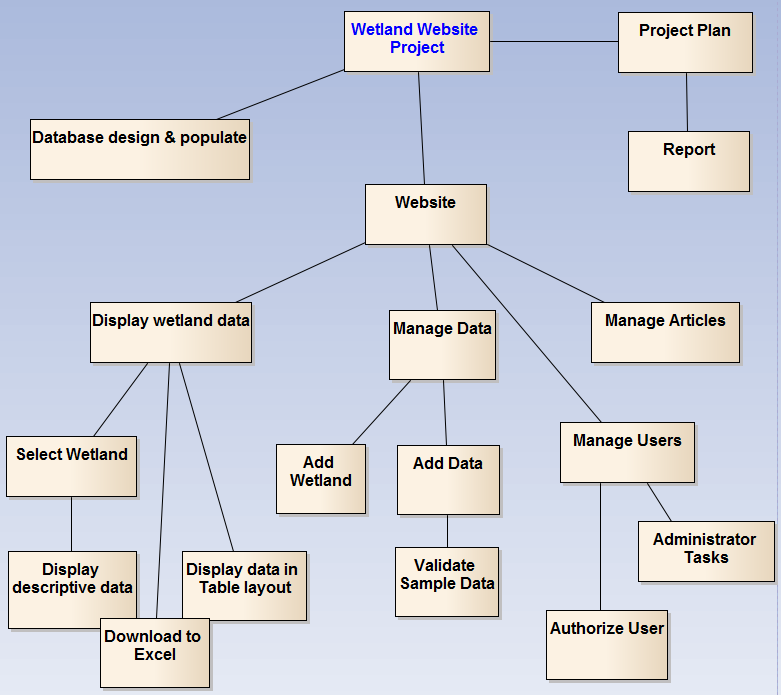


Figure . Product Breakdown Structure

## Risk Analysis

{Todo}

# **System Analysis and Design**

This document

## Research of similar systems

{Todo}

## Requirements

{Todo}

1. **Scope of Functions**
   * 1.1. The system will record ….
   * 1.2 The system will allow ...
   * 1.3 The system will generate ….
   * 1.4 The system will keep track of .. .
   * Design the interface to target people with scientific background.
   * Database can be updated without having specialized skills.
   * Database can only be updated by registered users.
   * Display data on specific wetland sites.
   * Be able to download data to an excel spreadsheet
   * Display links to publications and literature on wetlands sites.
2. **Operational**
   * 2.1 The system will run on any Web browser and on the intranet.
3. **Performance**
   * 3.1 Download speeds.
4. **Security**
   * 4.1 Information

## Business Rules

{Todo}

## Use cases

{Describe}

Register User

Authorize User

Download Data

Query Data

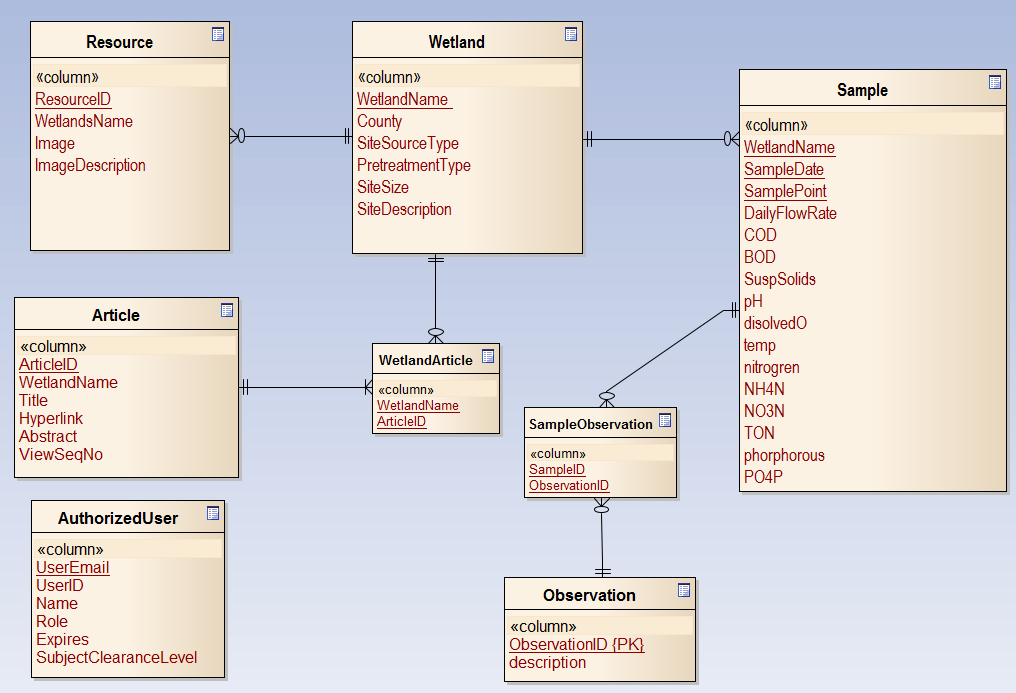
Submit For Review

Upload Data



## Data Modelling

{Todo}



## Interaction Modelling

{Todo}



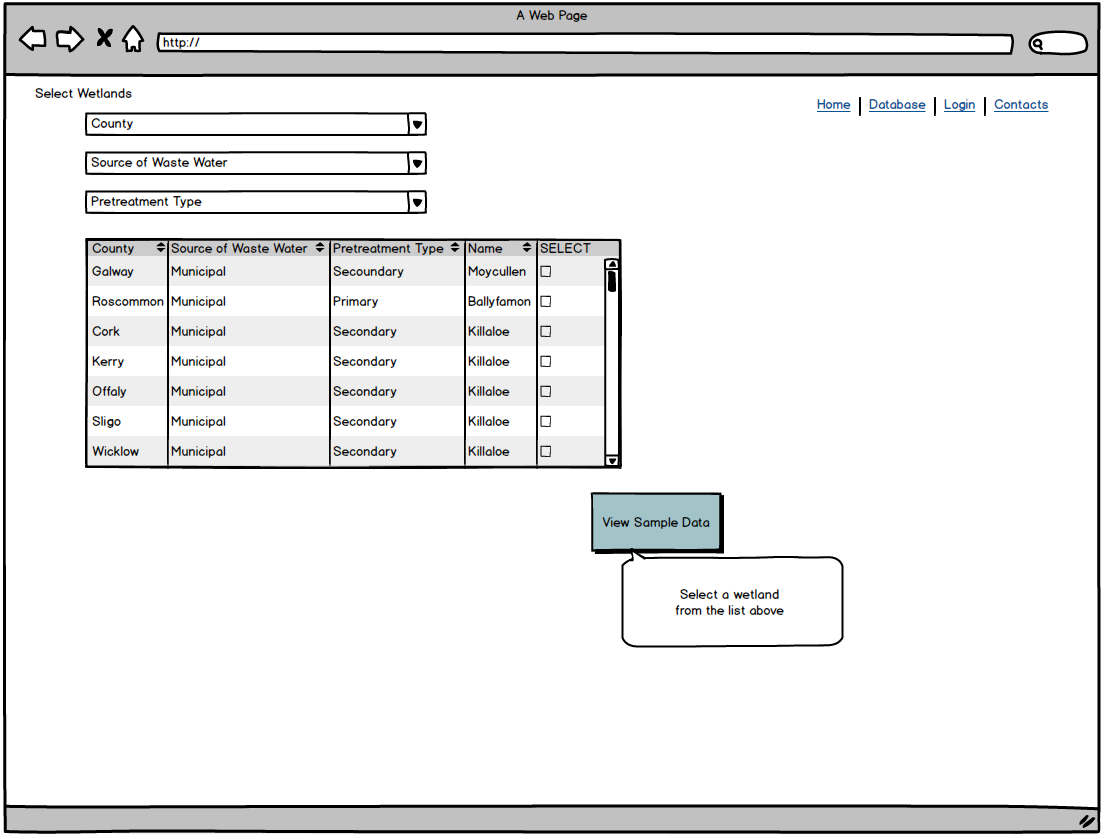
## Project Architecture

{ Discuss }

## Selecting Technologies and Tools

{Discuss}

* Twitter bootstrap
* My SQL
* PHP
* Java script



# **Implementation**

This document

## Database Structure

Todo

## Code Structure

Object oriented PHP Classes  
We decided to use an object oriented approach to developing the website. The main advantages of using the OOP method are the reusability of code and a clearer structure of code. As part of this method several classes were built that have functionality that can be used in different parts of the website.

|  |
| --- |
|  |

Figure . Classes in website directory

The classes were a mix of main classes and helper classes. For example the “DB” class is a main class with a lot of functionality and the “Session” class is a helper class that made session configuration more efficient.

##### classes folder : helper functions

**Config.php:** provides a static getter function for global variables which can traverse a path through nested array elements :  **Cookie.php** uses static functions for getters / setters on cookies.  
**DB.php** is a wrapper *class* for PDO access to database. It provides helper methods to simplify queries and I/O on the data. Uses the *singleton pattern* to return a instance of database that only connects once on a page load ‘round trip’.  
**Hash.php** uses a secure hash algorithm to protect passwords and cookies.

**Input.php** for accessing super global GET / POST values, useful for doing form validation.  
**Redirect.php:** provides a static function which wraps the php header redirect command.  
**Session.php:** uses static functions for getters / setters on sessions.

**Token.php**: provides *cross site request forgery protection*, matches the current uses session token when a form is submitted.

**User.php:** controller class for **User** entity, logging in and checking permissions.

**Validate.php**: provides user input validation, boundary class.

##### core Folder : autoload classes

**init.php**: functionality to include on each page: start session so people can log in, access global configuration settings for the database, cookies and session token, and autoload classes so that classes are loaded efficiently (when they are required) offering *dependency* *injection* with php function spl\_autoload\_register. This will replace the list of all the required\_once functions on each page for a single point of reference.

##### functions Folder : sanitize functions

**Sanitize.php**: function that uses html encoding to sanitize input / output.

##### includes Folder: Error Handling 404

errors folder : **404**.**php** standard message

##### Static Helper Files

Validation Helpers : Input is automatically sanitized  
config.php : Single location for configuration values

The “init” class is required on every web page. This class includes the “Sql\_autoload\_register()” function. The “Sql\_autoload\_register()” is a quick and efficient way of loading in classes when they are required rather than creating and maintain a list of requires. The function is called, the parameter $class represents DB and the DB class is required in. This eliminates unnecessary requiring.

|  |
| --- |
|  |

Figure . “Init” class being required

|  |
| --- |
|  |

Figure . Auto loading classes

##### Login / Permissions

Functionality: User Validation, User Session (Remember me option), Manage profiles, Update passwords, Ability to Register

User\_Session : Holds session hashes of people who have asked to be remembered when they logged in. Stores a hash that corresponds to a user ID and if this matches a cookie hash then that user will be logged in.

## Components

### Bootstrap

We decided to use Bootstrap on the website as it an efficient and easy to use HTML, CSS, and JavaScript framework for developing web sites.

Bootstrap provides a set of style sheets that provide basic style definitions for all key HTML components. These provide a uniform, modern appearance for formatting text, tables and form elements. It also has cross-browser compatibility and CSS-Less functionality. This ensures uniformity across platforms so output remains the same whether you’re using Firefox, Chrome or Internet Explorer.

Bootstrap uses a responsive grid system that re-arrange automatically depending on the screen size. Whether you are using mobile phones, tablets and PCs each variation adjusts the width of the columns.

The “viewport meta tag” ensures proper rendering and touch zooming on mobile devices. The width property controls the width of the device. Setting it to device-width ensure that that the web page is rendered across various devices properly. The “initial-scale=1.0” property ensures that when loaded the web page will be rendered at 1:1 scale, and no zooming will be applied out of the box.

|  |
| --- |
|  |

Figure . Bootstrap CSS in the header

|  |
| --- |
|  |

Figure . Bootstrap grid system

|  |
| --- |
|  |

Figure . Bootstrap web page

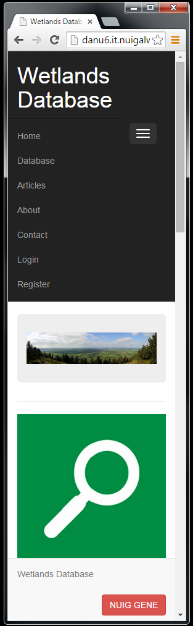


Figure . Bootstrap webpage resizing

|  |
| --- |
|  |

Figure . Bootstrap HTML example

## Data Persistence

Here is an example of the OO functionality of the classes.

The “DB” class is a database wrapper class. Its purpose is to have an abstracted way to work with the database. It contains various functions that can be used in different parts of the website to perform different functions.

The “\_construct” function connects to the database. A global configuration is set up in the “init” class. This contains the information needed to connect to the database. This allows for more efficient management. For example, the database information only needs to be updated once and not in ever place that the database is used.

|  |
| --- |
|  |

Figure . database private constructor, singleton pattern

|  |
| --- |
|  |

Figure . getInstance() function of singleton pattern

The “getInstance” function gets an instance of our database if it’s already been instantiated. This means we don’t have to keep connecting to the database again and again on each page. We can use the database on the fly.

|  |
| --- |
|  |

## Security

### Cross-Site Request Forgery (CSRF)

Cross-Site Request Forgery (CSRF) is a type attack where an attacker will use the identity and privileges of the victim and impersonate them in order to perform any actions desired by the attacker, such as change form submission details.

A token generation system was implemented to protect against this kind of attack. We generate a token that only allows data from the form to interpret. This makes sure that the requests are coming from the actual users of the site. A token is generated for the forms and must be tied to the user's sessions. It is used to send requests to the server, in which the token validates them.

|  |
| --- |
|  |

Figure Token in the session array

|  |
| --- |
|  |

Figure Token generation in the form

|  |
| --- |
|  |

Figure Token validation

### Password Protection

To protect passwords salted password hashing was employed.

Hashing a password will take a clear text string and perform an algorithm on it to get a completely different value. This prevents you from storing the clear text passwords in the database. The Hash function “SHA256” was used in this website.

Instead of storing a user's password, you can store the hash of the password. When a user logs in again, instead of checking the password they type in against the one you have stored, you calculate the hash of the password they type in and compare that to the stored hash.

However, hashing alone will not provide enough protection. There are many ways to recover passwords from plain hashes very quickly such as dictionary and brute force attacks. An effective way for ensuring that passwords are securely protected is by adding a salt.

A salt improves the security of a password hash because it adds a randomly generated secure string of data onto the end of a data that can then be added to. The hashed password and the salt are stored in the database. In figure 111, the users have the same password but there hashed password is different because of the salt. This ensures secure password protection.

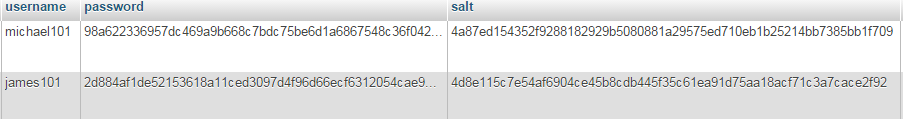
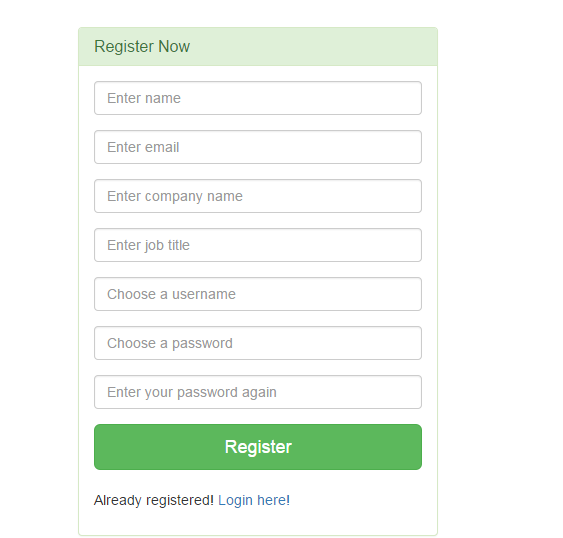


Figure Example of a salted password hash on the same password for two different users

The website required the ability to register a user, log in, query a database, have a permissions system, upload files, download files\*

### Registering

To register an account a user must fill out the “Register Now” form and click the register button. There details are stored in the database once registered and can then log in.



### Validation

The input from the user must pass validation conditions. For example, the “username” chosen must be unique from any other username in the database and the “Enter your password again” input must match the input on “Choose a password”.

The “check” function in the “validate” class outlines the validation cases. The conditions of these cases are defined when the “validate” class is instantiated. For example, the case “min” calls the function “strlen” on the posted data. The “strlen” checks the length of the string on the data. If the “$rule\_value” is set to “2” the input from the user must contain at least two characters or else an error message is outputted.

|  |
| --- |
|  |

|  |
| --- |
|  |

#### Logging in

To login a registered user must enter their chosen username and password into the login area. If an incorrect username and/or password is/are entered an error message is outputted.

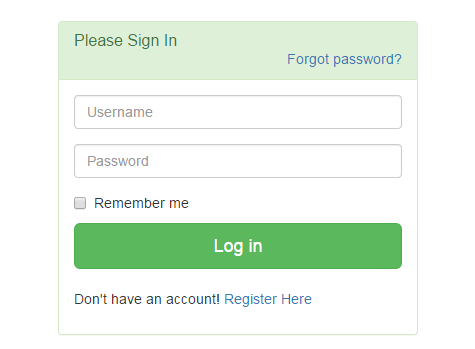


Figure . Log in area

When a user logs in there “userid” is assign to the “Session” which essentially logs the user in.



Figure . Assigning the user id to the session

When a user is logged in, different content will be available for them to see. For example, the navigation menu will contain different links for a logged in user and a user that is not logged in. This is achieved by the functionality in the “User” class which checks if the “userid” is assigned to the “Session” and if so the “\_isLogginIn” condition is “true”. An example of this can be seen in figures 12 and 3.

|  |
| --- |
|  |

Figure . Assigning the user id to the session



Figure . Assigning the user id to the session



Figure . Assigning the user id to the session

#### Remember me functionality

The user has the option of being remembered as they log in. Cookie functionally is used to carry out this functionality. If a user chooses to be remembered, a cookie with a hash value is stored on the user’s computer and in the database. The cookie is in the database with the “userid”. When the user visits the website again on the same computer “userid” is assigned into the session and the user will be automatically logged in.

|  |
| --- |
|  |

Figure . Assigning the user id to the session

#### Logging out

Logging out is essentially the act of deleting a particular session. The “logout” function in the “User” class uses the “Session helper” to delete the session and the cookie assigned to the user. The cookie hash is also deleted from the user session. Without deleting the cookie hash the user would stay logged in.

|  |
| --- |
|  |

Figure . Assigning the user id to the session

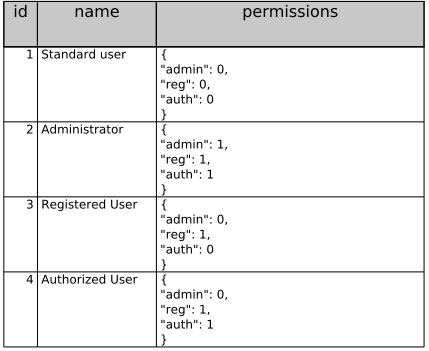


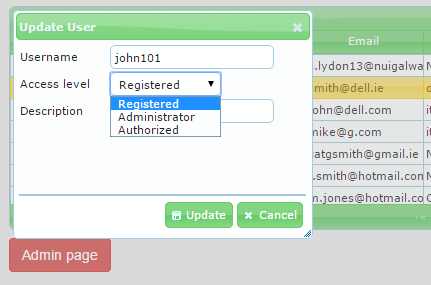
Figure . Table Groups, showing different roles

Groups : permissions holds a JSON string of different permissions a user can have.



The website required a permission system as there would be different users with different access levels on the website. For example, an administrator would have different permissions to a registered user. There are three permissions levels; registered, administrator and authorized. A registered user cannot upload any data, only an authorized or an administrator can. An administrator has the ability to change user’s access level.

When a user registers there access value is defaulted to “1” which is a registered user. It is done using a “jqgrid”. A user is selected in the “jqgrid” and their access level can be changed to a different permission level.



The other important thing for uploading files is you must have “enctype= multipart/form-data”. This tells the browser to expect files to be uploaded.

|  |
| --- |
|  |

|  |
| --- |
|  |

### SQL Injection

A SQL injection attack is where harmful/malicious SQL code is into injected into the database, and the database is forced to run his SQL. An example of SQL injection create an SQL statement to select a user with a given user id

## Website Structure

### Sample page:

The sample page for the constructed wetland database displays the data on the individual wetland. This page contains four tabs: sample data, wetland details, publications and observations.

Each of these four tabs illustrate different pieces of information about the wetland, three of which will be accessible by the unregistered user. The sample data tab contains all of the information gathered from the wetland over time by sample measurements on specific dates. This includes measurements of certain chemicals such as phosphorous as well as results from chemical oxygen demand tests.

The wetland details tab contains images relevant to the wetland as well as a site description giving an overview of the constructed wetland details.

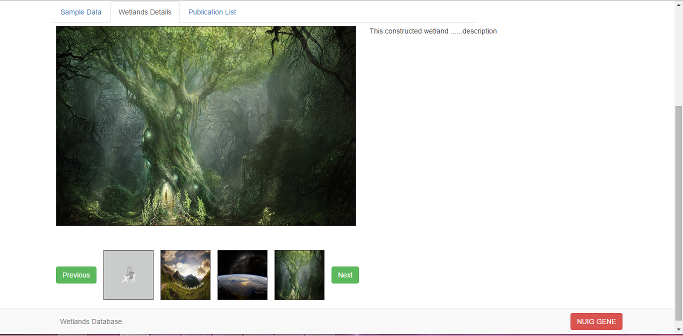


Figure . Wetlands Detail

This tab relies heavily on javascript functions to operate traversing through the images, and allows for the user to scroll iterate through the image selection for a wetland, provided there is one. The smaller images below are thumbnails and on mouse over will change the larger picture above them to become the same picture. There is a slight delay effect added here to add a pseudo-animated style to the gallery.

|  |
| --- |
|  |

Figure 25. JavaScript for scrolling sequence.

Images are uploaded to the site via a form which validates server side the format of the file through php. The user must be logged in in order to upload images. If it passes validation, it is uploaded to the server and a database record is created for the image. The database record then contains a link to the image on the server for use in the html table on the wetland details tab. However, before it is shown in the gallery for a specific wetland, it must be validated by the administrator, who verifies that it is suitable. If the image is appropriate, there is a checkbox on the record which will signal that the image is acceptable for viewing. The SQL query on the tab will only link images which have passed this validation. If the administrator does not think that the image is appropriate, then it will be deleted from the database and the server via the same admin page that would allow it. The images are stored on the server because unlike the sample data images sent by researchers, these images will be available to the public and so storing tem in the database through binary encoding or otherwise would result in a sizeable reduction in the databases speed and ease of use. Many popular photo sharing websites store publicly viewable images online using a file system rather than in a database so this seemed like a good idea to prevent too much delay when trying to view images for a wetland. The site description to the right of the images is an overall description for the wetland. This description is non-editable by anyone but the administrator. The administrator may update this whenever, and as it is unique to each wetland, it is simply a field in the Wetland table in the database.

The next tab on the page is the publications listing. This is a table containing the details of all of the relevant articles for the specified wetland.

|  |
| --- |
|  |

This tab contains the php which retrieves the literature from the database and outputs it row by row in a table format, which is outputted in order of priority. This arbitrary level of priority is decided by the administrator, who as the client specified, is more likely to hold articles written by the NUI Galway research group in a higher precedence than articles written about the wetland by other researchers. It is allowed by a field in the database record which indicates its importance. It is to encourage traffic to NUI Galway papers. The table uses some JQuery to allow highlighting of the rows on hovering over a row, and also allows sorting of the articles by the year by which they were written. Note that sorting the publications like this takes away the priority that NUIG papers had on initial loading of the tab.

|  |
| --- |
| Figure 26. simple JQuery for highlighting the row which is being hovered over. |

The articles are not stored on the database or website itself, but rather a DOI is given for each publication, enabling the user to read a publication as long as the paper remains available to the public.

The observations tab is a tab available to view only by registered members of the website. The data from the wetland on the previous three tabs has been purely objective data, which has been observed and researched. This tab is more concerned with incidents that may have happened to the wetland over time, and possible conclusions drawn from that.

The objective data is open to interpretation, but it will only be those who have an interest in the field who will have an understanding for what each field in the sample grid means. Therefore it is unlikely to be misinterpreted by the public. However, observations and incidents which may be perfectly fine in the case of a specific wetland, may be misconstrued by the user who does not know much about the operation of constructed wetlands.

|  |
| --- |
|  |

As such, it is important that the database is not used against NUIG or anyone involved with the maintenance and/or construction of a wetland, so this data must be hidden from unregistered members. This tab is generated in a very similar format to the publications tab, in that it queries the database and retrieves records specific to the wetland ID gained from the user searching the query page. The main difference is that on the sample page, the tab is only shown if the user has logged in through a simple <?php (if($isLoggedIn) ?> verification.

This verification minimises the risk that a user will misunderstand the data and lead to unnecessary difficulties with any parties involved with the wetland.

### Browse Literature:

This page operates in much the same manner as the publications tab for the wetland page, querying the database through php and then echoing out rows of articles, if there are any. In this case though, the query is not tied to any specific wetland, it reads and writes all the literature contained in the database. Like the publications tab, it contains row highlighting and sorting by year, as well as a scroll bar for navigating through the body of the table so the user doesn’t have to scroll down the page in order to view all of the literature in the database.

The styling for the table is done through bootstraps css and the colour was chosen by creating an image of a colour gradient and using it as an image for the table’s header. It only takes up the top half of the table header columns height wise and so can be positioned on top of the background colour of the row, so that it stands out. The row’s background colours are decided through a php condition which designates the class based on the number of the row.

### Forgotten Password:

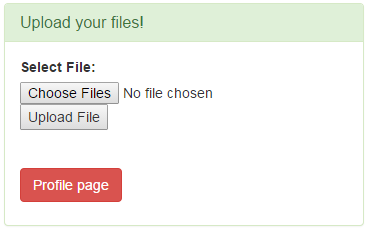
If the user has forgotten their password to the system, they will be able to enter this page and input their e-mail address into a form. This form will be passed to a page which sees whether there is an account with the specified email address, and if there is, creates new password for the user. This is a randomly generated string which uses numbers and letters but does not use letters or numbers which are likely to create confusion, such as the number ‘1’ and lower case ‘l’. The password is encoded and given a salt, and replaces the old password in the database. It is sent to the specified email address, along with a message encouraging the user to change their password, to prevent against security leaks. In addition, the person who fills out the form is not notified as to whether an e-mail has been sent or not, as this facilitates potentially unwanted situations where someone mass-inputs email addresses in order to determine the addresses linked to an account. Although this is unlikely to be an issue with a website for a constructed wetland database, it is nonetheless a good practice to take when designing a system, to be safe.

### Uploading files:

Users that have the suitable permissions have the option to upload files. The “UploadFile” class handles the file uploads.––––

Uploads can be restricted to a predefined range of MIME type. The class will automatically append a suffix to the names of files that could cause damage to the website. This includes files such as PHP and JavaScript files. It automatically handles both single, and multiple file uploads. Duplicate files are automatically renamed by inserting a number before the file name extension.

|  |
| --- |
|  |



## Phased development, next step

Todo

# **Testing**

This document

## Unit Testing

Todo

## System Testing

Todo

## Interface User Experience Testing

Todo

# **Conclusion**

This document

## Unit Testing

Todo

## System Testing

Todo

## Interface User Experience Testing

Todo

=============================================================================

## References

***Environmental Science****.* (2014, 11 30). Retrieved from Dolores Gende's AP Sciences Website: http://apesnature.homestead.com/

***Geo-Environmental Engineering Research Group*** <http://www.nuigalway.ie/gene/>

# APPENDIX A Preparing for Interviews

Questions Interview 1

22 October 2014

11:52

|  |  |
| --- | --- |
| Describe the Users | County Council technicians |
| Describe the key concerns that the system should address for each user | Who will be using the web site ?   What types of user would be accessing and using this system ?  For each type of user, what aspects of the data would they be concerned with? |
| Describe how the data is collected - samples onsite, sent to labs |  |
|  | What information would be most important to the users? |
| Understand the Data involved | Sources/ Sinks of information ?    What external systems need to be interfaced to? |
| Submit Data | What is the process for submitting data?    How is it collected? By whom?  What form does the data come in? |
| Wetland Performance | What are the performance indicators that need to be calculated? |
| Understand the processes involved | What typical searches would be essential to be made available on the website?  What services does the application need to provide |

Questions for interview 2

05 November 2014

11:56

|  |  |
| --- | --- |
| Concern with size of spreadsheet - how it will grow and how that can be uploaded | There will be approx 140 sheets - how will that be uploaded  How much would that grow with samples for each month |
| Identify stakeholders / roles | ACTORs 1 - general public ACTORs 2 - Engineer , Environmental Scientist,   Students, Local Interest Groups |
| How users will use the website | If they saw the tabulated form would that make sense to them?    What sort of questions would they have in mind when they come to the website    Would they be comparing wetlands?  What would they expect to see?   Can you describe the scenario where actor 1 / actor 2 visits the web site. |
| GET TERMINOLOGY - glossary | Clarify the 8 important properties …. Would we be inputing the same statistics - or different element(sample components) |
| Graphical Representation | What graphs do you have on the spreadsheet?  Is that something that would be useful to show on line? |