**CS39930 – Assignment**

**Technical Design**

**150071107**

Contents

[Introduction 3](#_Toc509085878)

[Page Information 4](#_Toc509085879)

[Public 4](#_Toc509085880)

[All Pages 4](#_Toc509085881)

[Login 4](#_Toc509085882)

[Leave of Absence 4](#_Toc509085883)

[Join/Contact 4](#_Toc509085884)

[Private 5](#_Toc509085885)

[All Pages 5](#_Toc509085886)

[Dash 5](#_Toc509085887)

[Leave of Absence 5](#_Toc509085888)

[Progress Tracker 5](#_Toc509085889)

[Website Editor 5](#_Toc509085890)

[User Accounts 5](#_Toc509085891)

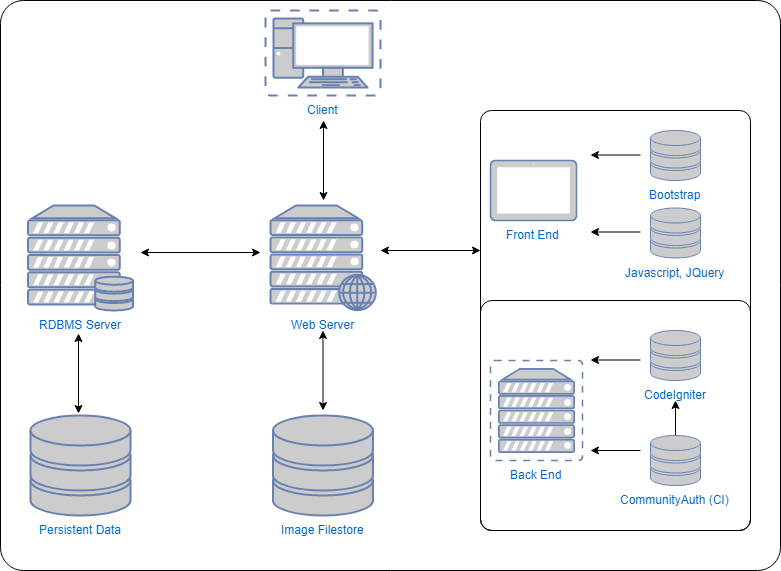
[DB Design 6](#_Toc509085892)

[Sample Data 7](#_Toc509085893)

# Introduction

This document will be providing all the information in relation to what the users won’t specifically see in a literal sense, however will detail all the back-end structure and technologies that will be used to achieve the set out use cases – the technical design. It will cover the website structure and some navigational aspects, the specific technologies to be used in regards to the website overall. Each page will again be detailed like the User Interface, however the focus will be on any specific technologies in use on that page and any special work that would be undertaken for that page only. Finally, the database design will be presented and discussed alongside some sample data.

**Web structure hierarchy and navigation**



**Figure 1- Website Structure**

The structure of the webserver and site is all to be contained primarily within one web server, however it will split appropriately with software or into various virtual servers that are able to talk to each other.

The client will access the web server in the diagram only via the web link – all technologies and features are then accessed from there via the code. Primarily, this comes from the php programming language and its added framework – CodeIgniter. CI is an MVC-style php framework that has multiple features added for ease of use in website creation. CommunityAuth is a library created by a member of the CI community whom has made it open source for easy creation of a secure login system.

This backend framework is supported by the addition of bootstrap and its associated libraries for the front end, visual side. Bootstrap enables an easy method of creating a good looking responsive design with extra features to make the best looking website possible. Javascript and JQuery are also added as part of bootstraps functionality and make performing certain functions easier.

The image filestore is local to the web server and simply a directory where images can be uploaded via a secure page on the private administration area of the website. All images are checked via codeigniters upload library to ensure there is no malicious uploads.

The RDBMS is used to store the persistent data, with its design and usage described later on in this document. The database language to be used is MySQL, owing to its widespread usage, reliability and my own proficiency with it.

# Page Information

The following section will be describing the various technologies that will be used on each page. Some pages are omitted from this as all features are described in the ‘all pages’ section. It will also state what calls will take place in regards to the database and any other specific functions required.

## Public

### All Pages

All public pages will be performing at least one call to the database – this is to access the pages table to grab the data of the page (as it is stored persistently so it’s editable) and then display it appropriately. All pages use the same header and footer – the same buttons as per the navigational diagrams in the UI design. The primary technologies in use is CodeIgniter to deliver the page and provide uri routing. Bootstrap is also in use on the front-end for styling and responsive design.

### Login

There will be a second database call from performing the login action of using the CommunityAuth library (from CodeIgniter) appropriately to determine a successful login or not. Upon success, the user will be redirected to the private area dashboard. Failed logins will remain on the current page using CommunityAuths error handling. This will use some of bootstraps functionality to provide a soft window – a modal.

### Leave of Absence

An extra call is made to the database here, however it is a push as opposed to the others. After codeigniters form validation library is used to sanitise the input, it will then be submitted to the relevant table in the database for storage.

### Join/Contact

This page will also have additional functionality using the form validation and handling libraries of Codeigniter. This will then, once processed, be passed to email handling and be sent onward to squadron email address.

## Private

### All Pages

All pages within the private are will be using the majority of the outlined technologies in the website structure. They will taking full advantage of the libraries and features provided by codeigniter, especially in regards to database calls and form handling. Bootstrap is again in use for styling and to easily create responsive design.

### Dash

This page is an informative page that you are directed to upon successful login. Attempting to get to any of the other pages via the URL will redirect to this page if logged in due to permissions handling. If not logged in, the user will be directed to the login page first. There are a number of extra database calls occurring here – the first two are simple pulls from the database of some latest information, so as to keep the user up to date. The final call is in regards to permissions and that of which the user holds. While they are unable to see this, it will then determine which pages they are able to access in the private area.

### Leave of Absence

This page is performing a call to the loa’s table of the database before presenting it to the user in a table – those that are marked as true on the IsArchived Boolean stored in the database are not shown in this table. Those archived (via a button after being actioned on systems not made by myself) are able to be viewed by clicking an appropriately titled button.

### Progress Tracker

The most complex area, this consists of numerous calls to and fro the database. This follows the design presented in the UI documentation and holds the most complex database design points. It enables the tracking of cadets progress in regards to what badges they have achieved, split into levels. The badges must first be added to the system along with cadets names before the tracker can be fully utilised. These are all done via simplistic menu buttons in an area appropriate to the design – all functions are making database calls of both push and pull to deal with the information appropriately. There will also be some JQuery in use on this page to handle information easier and present it in easy formats.

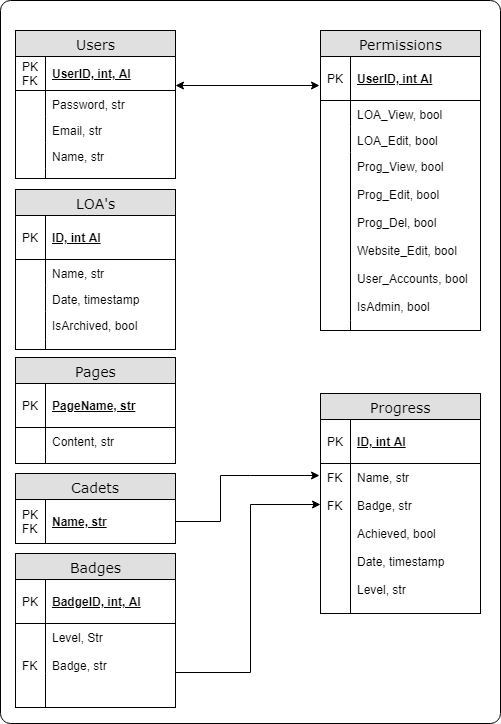
### Website Editor

An important page yet one that is relatively simple in concept. There is three actions being performed here – a call to the database to the Pages table to present the editable pages in a menu and then a form containing the primary website content. Once edited, this can then be saved which is pushed appropriately into the database.

### User Accounts

Another relatively simple concept yet important page, this provides the site manager the ability to change user accounts and associate permissions, which fall into the ability to view the features of the private area and then the ability to actually edit or use them. This is again a pull and push from the database – taking the appropriate information to display to the user before pushing it to the database to save once the user has finished and presses the appropriate submit button.

# DB Design



**Figure 2- First Database Design**

The figure seen above is the initial design of the database. The main change that is most likely to occur is in regards to the Users and Permissions tables – CommunityAuth may require it to be setup slightly differently in requirements. There may also be a slight variation in the Progress table – while this should currently hold all required information, the development process may reveal a slightly easier method of performing this – a comparison will be made from the initial design now to the final version. IsAdmin is a Boolean on the users table to determine access rights – it will mean that the account cannot be deleted by other accounts with the ‘user accounts’ permission.

The Cadets table is on its own currently due to the design – future development has this setup to hold more information if required – however the only requirement to be held now is a Cadets Name as use in the progress tracker.

All parts of the design are set to Not Null, as all information will be required for usage of the website in its current format. The data types are all sensible, logical choices with the majority either being strings (varchar) or booleans to denote true or false. There are a few dates, using the timestamp option for a precise time for later usage. There are a few auto incrementing (denoted by AI) options that are uniqueID’s to enable easy tracking and usage of the records.

All pages will draw from the database in some way. The public area will be drawing from the pages table to grab the editable content and display it, with the leave of absence page inputting data into the LOA’s table.

All pages of the public area will have the option of going to the login page, which will also perform a RESTful request to the users table to determine login rights. All pages of the private area will perform calls to the Permissions table to determine access rights. They will also call and push to the table related to their page to both obtain the information to present and then perform appropriate calls to input data when required. All parts of these will use CodeIgniters validation library to prevent XSS and code injection.

# Sample Data

The below data is a set of varied data to be inserted prior to commencing testing that provides all available data sets to be fully tested. In regards to user accounts, the “admin” user is a permanent one intended to be used in times of website setup or to solve issues; it will create another user account that can then be used from there. The majority of this data should be submitted in the process of testing using the forms and menus as opposed to direct DB insertion.

**Table 1 - User Testing Table**

|  |  |  |
| --- | --- | --- |
| **Password** | **Email** | **Name** |
| Test1 | 561@aircadets.org | Andrew Mann |
| **Enabled (Permissions to select):**  LOAView, LOAEdit, ProgView, ProgEdit, ProgDel, Website, User Accounts | | |
| password | [561@aircadets.org](mailto:561@aircadets.org) | Andrew Mann |
| Test1 | I\_amwrong.com | Andrew Mann |
| Test1 | [561@aircadets.org](mailto:561@aircadets.org) | andytheman |
| Test1 | [561@aircadets.org](mailto:561@aircadets.org) | Jessica Jones |
| **Enabled (Permissions to select):**  LOAView, ProgView | | |

**Table 2- Cadets Testing Table**

|  |
| --- |
| **Name** |
| Evans0 |
| Evans( |
| Evans; |
| Evans. |
| Evans |
| Jones |
| Parry |
| Lloyd Jones |

**Table 3- Progress Testing Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Badge** | **Level** | **Date** |
| Evans | Trained Shot | Blue | 01/01/18 |
| Evans | Trad shog | Blue | 01/01/18 |
| Evy boy | Trained Shot | Blue | 01/01/18 |
| Evans | Trained Shot | Bl | 01/01/18 |
| Evans | Trained Shot | Blue | 31/02/18 |
| Evans | Trained Shot | Blue | 01/01/25 |
| Jones | DofE | Bronze | 01/01/17 |
| Lloyd Jones | First Aid | Silver | 15/03/17 |
| Parry | Leadership | Gold | 15/03/18 |

**Table 4- LOA Testing Table**

|  |  |  |
| --- | --- | --- |
| **Name** | **Date** | **Reason** |
| Evans | 06/06/2018 | Don’t want to go |
| Eval | 06/06/2018 | Don’t want to go |
| Evans | 05/06/2018 | Don’t want to go |
| Evans | 06/06/2018 | No thanks”’DROP TABLE loa;’ |