CAB230 – Web Computing

CLIENT-SIDE documentation

n10234756

2019

Contents

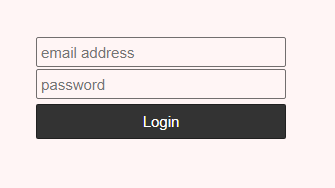
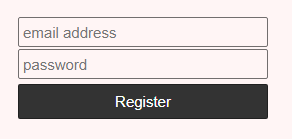
[Intro 2](#_Toc10392849)

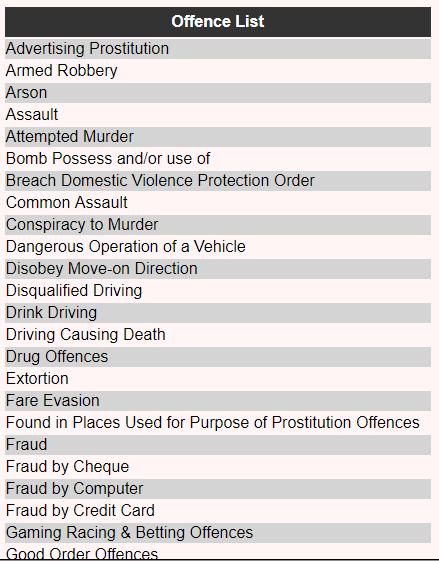
[Technical, testing & limitations 3](#_Toc10392850)

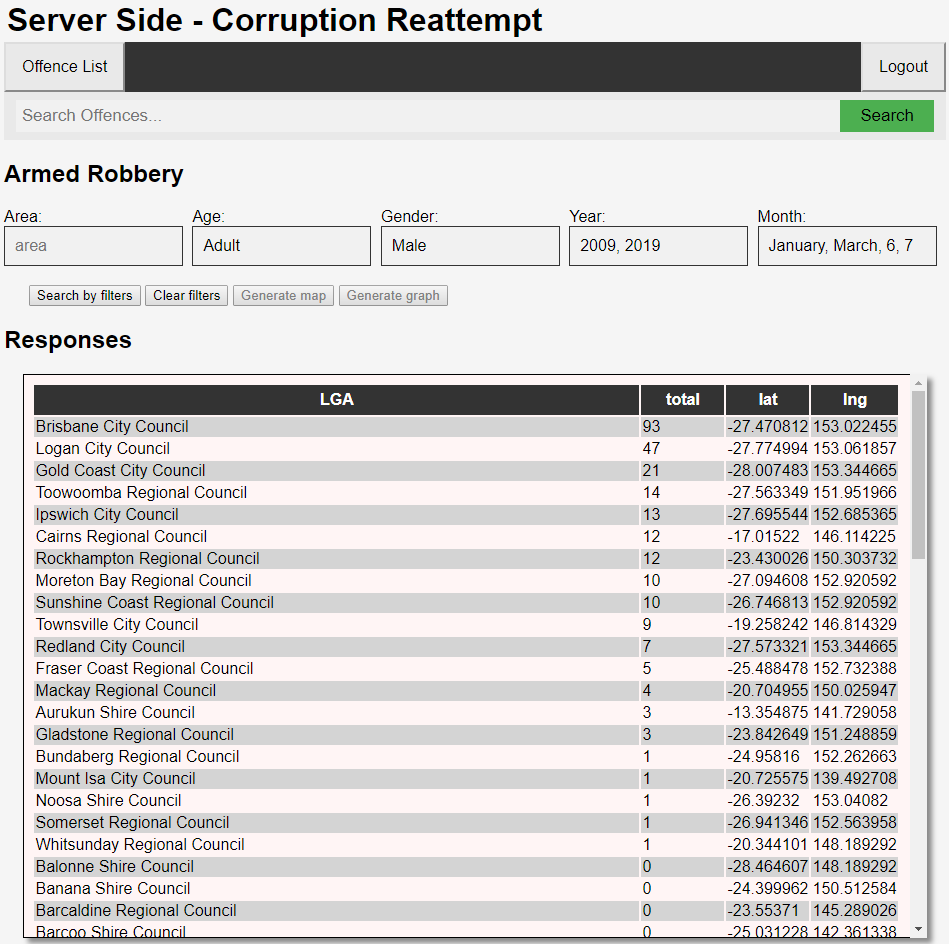
[User Guide 5](#_Toc10392851)

# Intro

Basic functionality was implemented due to technical difficulties and limited time. As was observed during 9-11 tutorial by tutor, the assessment (and cloud based backups) had corrupted, thus an attempt was made to recreate a basic client-side interaction, however due to the strictly limited time available, react was not able to be implemented, this also extends to a rather basic design. Due to slack being unable to reliably connect the account used (instead automatically redirecting all traffic to CAB302 Slack channel), acceptable libraries could not be reviewed, thus external libraries are not used.

The register and login endpoints were used to create login forms that allow a user to either login or register. When a user is logged in, they are unable to relog or register until logging out.

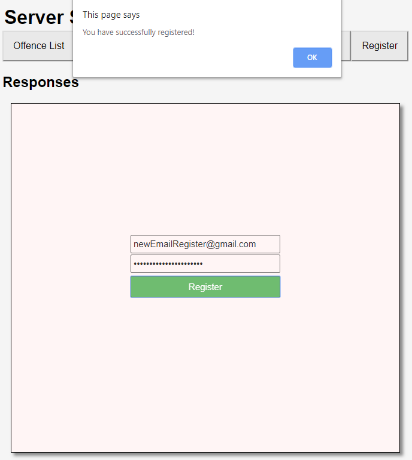
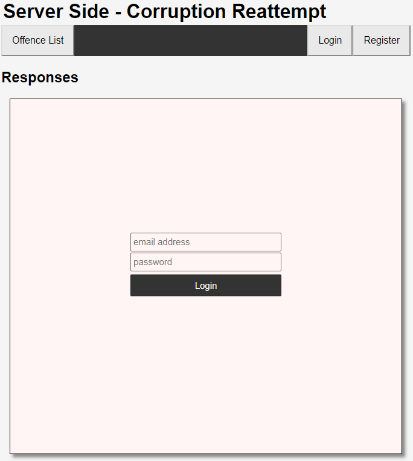
The offence list is available, this serves a table containing all available offences. The user is then greeted with a search form and filters; however, the filters are unavailable until an offence has been searched, this is because the offence parameter is an integral part of the call. As can be seen above from the provided portion of the offence list, the entries are formed into a table which dynamically grows based on the size of the retrieved content using element keys. The offence list can be used as a source of reference for the user as to what offences are available to search.

Users can search an offence in the provided search bar, the process has been simplified by adding an auto complete method to the search bar, allowing a user to easily complete their search queries.

As can be seen multiple different filters can be applied, and multiple of each filter can be applied, as is shown, results are filtered to be Males offenders during 2009 and 2019, during the months January, March, June and July, which can be requested using either the month name, or the index value of the month.

Tables have the ability to be sorted in an alphanumeric way, in both a descending and an ascending order, this is done by clicking on the header of the table, this is demonstrated using the total in descending order.

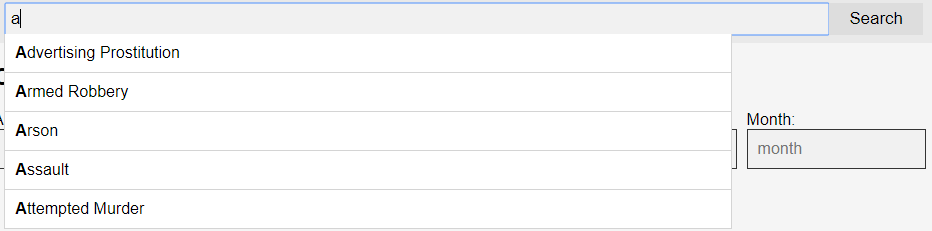
# Technical, testing & limitations

Once registered or logged in the user is greeted with a simple confirmation in the form of an alert. Once logged in, the user is given additional tools to navigate with, such as a search and various filters.

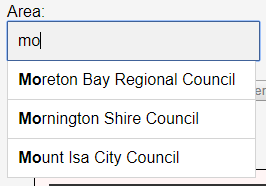
The login and register forms share a method to help prevent duplication, instead using different parameters to help differentiate which form should be served on request. As is common with modern websites, a user is also logged in upon registration.

To attempt to minimise repeating DOM using Document Fragments was originally intended, however due to the nature of some of the information being used, would prove to be as impractical as direct DOM changes, however, due to time frame the simplest method would be to modify some elements indirectly without using the DOM, as to avoid the time it would take to constantly rewrite and delete these elements. However, as the project was unable to be written in React.js, it does not use a virtual DOM such as would be found in a react module, as such is not as fast and uses more memory.

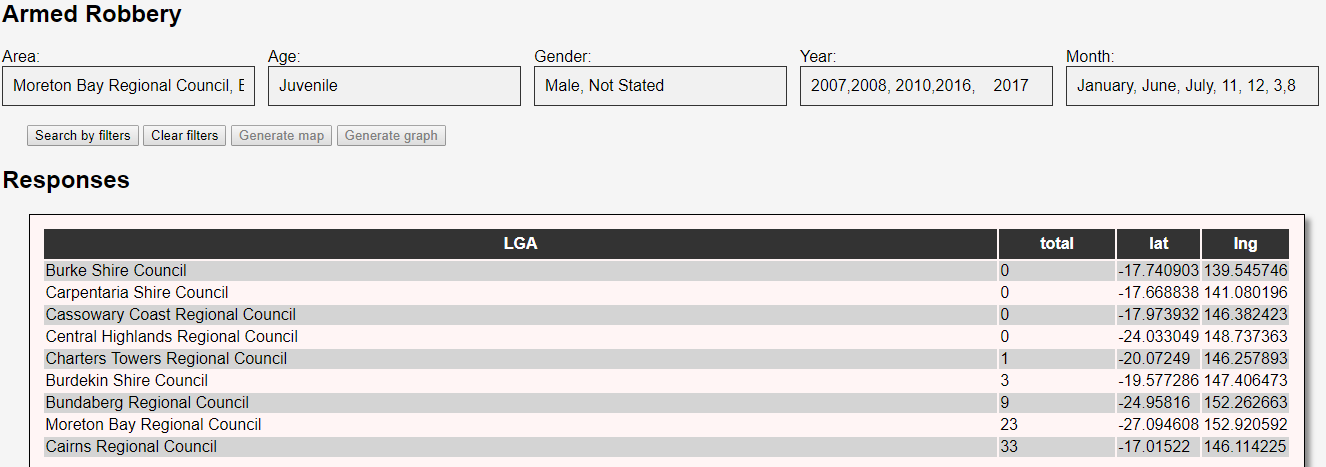
All tables are dynamically generated and can be sorted alphanumerically through the header. An attempt was made to combine the numeric and alphabetical ordering into a singular function due to their innate similarities, however Booleans and value matching proved in effective, often resulting a large memory leak which would otherwise crash the webpage, as such this not was implemented and the separate methods remain.



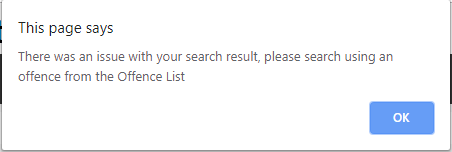
An automatic word match was also implemented, this is to assist users in searching for valid offences and make the selection process far easier as users can simply select a value from the dropdown list. This value is then retrieved in order to process the fetch call. To account for all possible instances of offences, the method is populated using an initial fetch to offences.

Results from the initial search can then be further filtered, similarly to the initial autocomplete, each filter has its own variant, this was not initially possible and required modification and reallocation or variables, however, eventually became a possibility. Following the same rules as the offence search, users can click or select the dropdown elements using the arrow keys. Doing so will automatically fill out the field.

Multiple elements can be concatenated in each filter area. Additional filters can be concatenated using commas. As many people instinctively add a space after a comma, the inputs are handled using a regular expression to be placed into the correct format to be used in a fetch call. Area, Age, Gender and Year use this method, however Month uses a slightly different variation, instead splitting and mapping the user input to an array, before returning these values as a usable string.



As can be seen above, multiple filters can be used, initial order does not matter and months can be input through either their name or their ordinal number, the table is also alphanumerically ordered to be descending, all fluently interacting with each other, many elements need to be indirectly handled for this to occur, as directly handling all elements at once often resulted in freezing and memory leaks.

Catches have been implemented to prevent issues such as false entries and alerts issued to notify the user, for example searing “Derby Racing” in the offence search will return an alert notifying the user this is not a valid entry. Invalid filters use a similar notification. A filter clear button is created using a simple value check which resets the initial state of the filters, allowing users to reset and restate their filter parameters rather than individually resetting each filter. Filters are disabled when an invalid offence is used.

All elements attempt to scale with screen size. The autocomplete originally had the ability to query multiple objects in a concatenated state, however this occasionally returned either erroneous results or did not correctly display results, this could not be resolved in the short time frame, as such was removed.

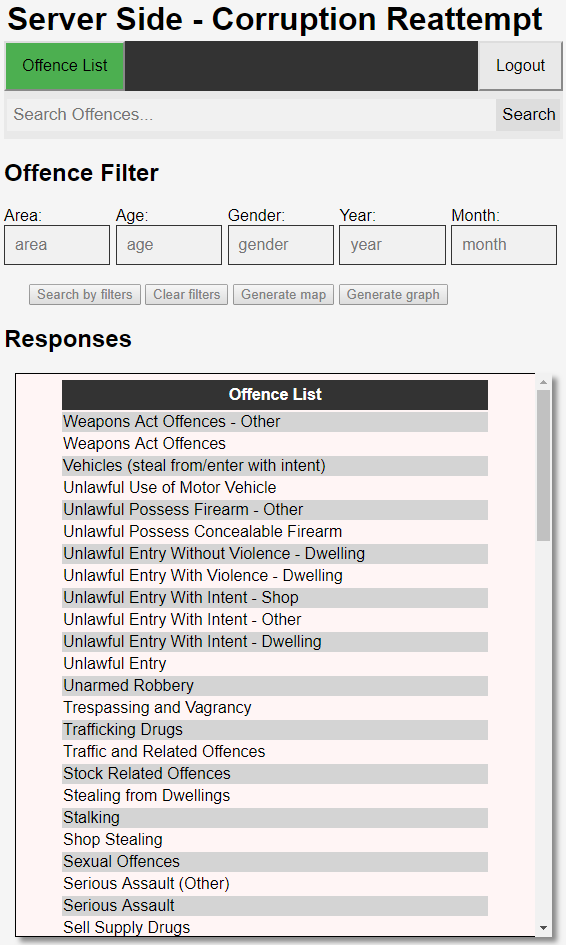
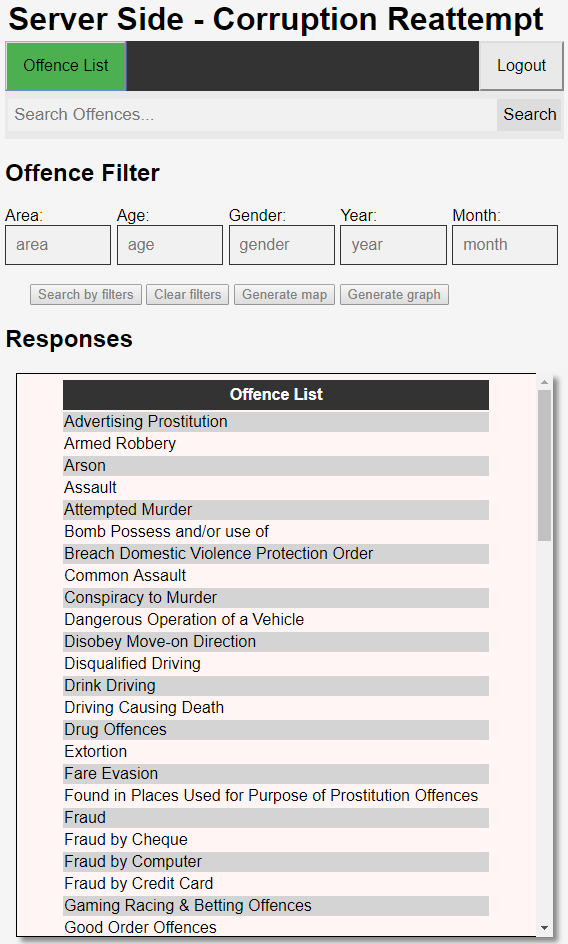
The biggest technical issue encountered was the multiple unexpected instances of the corruption of both current and backup documents, the cause of this is still unknown, this issue resulted in the complete loss of all relevant files, as such a new work environment had to also be setup. This issue resulted in the need to recreate the client side, due to the need to recreate the client side and the very limited amount of time to do so, along side the issues encountered when accessing resources, API’s and libraries for mapping and graphing could not be identified as acceptable (as the previously used libraries/APIs were react native only), alongside the short time frame, prevented the creation of a personal representation of these.

# User Guide

1. Login or Register

The user must login or register to use authenticated endpoints, this can be done though submitting a form containing their authentication credentials. Login and Register should feel familiar to a user. These can be accessed by clicking **Login** or **Register.** Logging in will remove the ability for a new user to login or register, and instead provide a method to log out. A new user can login or register when the previous user has logged out.

*Login Form Example*

1. Offences Table

The tabled of offences can be accessed by simply clicking the **Offence List** button. Clicking the Header of the offence list table will sort the table from A-Z to Z-A, and vice-versa. This is a list of all available offences.

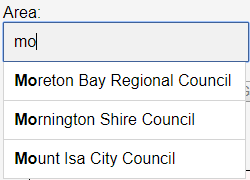
1. Offence Search

Users can search the various Offences available from the **Offence List**, this will create a table relating to the offence queried.

* 1. Auto complete

*A-Z Sorting*

*Z-A Sorting*

All input fields have the ability to auto complete user queries based on available searches. The **Offence Search** shows available offences as the user types, giving the option to select one. The **Area** filter provides the ability to select a region. The **Age** filter provides the ability to select an age group. The **Gender** filter provides the ability to autocomplete whether the convicted party is a Male, Female or Not Stated. The **Year** filter provides the ability select years based on the sample space of the results. The **Month** filter provides the ability to select the months of the year.

*Example of auto complete*

1. Filters

Users can filter for specific results based on the Offence returned from the **Offence Search**. These are all combined, and multiple values for each filter can be concatenated. Clicking on **Search by filters** will return the filtered table. Users can clear their current filters by clicking **Clear Filters**. The **Generate map** and **Generate graph** filters do not work due to time restrains, however can possibly be implemented at a later date.

1. Responses

The responses widget displays all elements returned by the client. All the generated tables (and intended to be generated maps & graphs) are placed here, this is a universal centre point which does not warp or hide any elements, instead becoming its own interactive window.

* 1. Forms

Forms are placed in this window. These forms are like other login and register forms on the internet today, providing a familiar login/register environment.

* 1. Tables

All tables are viewed from within the Responses widget. These tables are all interactive, all able to be ordered based on the headers, from A-Z and Z-A to lowest-highest and highest-lowest number. The tables are sorted based on the header clicked and the resulting sorting is alphanumeric. These tables scale to fit within the viewable widget, however if the table is long enough, the option to scroll through the table becomes available.

1. Logout

Users can **Logout** when finished with the client, a successful logout is confirmed with a system message, the user is then returned to the initial state, where a new user can either **Login** or **Register**.

*Scrollable Table sorted Z-A*