Modeling the Senedd Election

William S. Conroy
976789

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Bachelor of Science

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

August 18, 2021

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date .....

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Contents

List of Figures

Chapter 1

Introduction

For a voter's interest to be upheld on voting day, an exhausting list of factors have to be considered. A voter first needs to have the enthusiasm to vote to begin with, the system their vote is counted in has to be unbiased, a voter must have a clear understanding of their interests and the interests of any parties. Key to this project is the idea that even if all those factors are correctly taken into account, there is still a piece missing. The voter must also be educated on how the voting system itself functions. This will allow a voter to cast a vote know that it will have a much impact as possible. The dissatisfaction people feel around the impact of their votes is clear, as demonstrated in a Public meeting in Cardiff. Where it was stated by a voter that in 54 years of voting they felt their vote "has not once had any bearing on the result whatsoever" [1]. Substantial effort has been put into several factors so that a voter's interest can be up held on voting day. Non-partisan Get Out the Vote (GOTV) campaigns utilised strategies from television ads to door-to-door calls [2]. In terms of educating voters, Voting Advice Applications (VAAs) have shown"considerable potential to increase political interest, knowledge, and voter turn out" [3]. Over the years multiple countries have looked into reforming the electoral systems such as Botswana's hunt for a replacement to their First Past the Post model [4]. But less has been done to educate voters on how to maximise the impact of their vote. Which this project hopes to address.

1.1 Motivations

More immediately obvious elective systems such as FPT, are intuitive and simple to understand [5]. This seems desirable and in line with our goal for voters to understand the impact of their

votes. However, these systems are not necessarily the most representative [1]. Due to similar concerns in 1999 Wales introduced the Additional Member System (AMS) system to elect its assembly in hope to address the representation issue. The AMS works by having two votes, the first of which is to elect a representative for their constituency, there are 40constituencies out of the assembly's total 60 seats. The constituency vote uses the simple FPTP vote. The second vote is where it changes. The second votes are for their regional representative. Each of the 5 regions has 4 representatives and they are selected using the d'Hondt formula. But once elected the 20 regional assembly members and the 40 constituency assembly members have the same authority. Because they are functionally identical there are huge ramification on who you should vote for. As AMS is incredibly useful for smaller parties, tending to increase the number of seats they get while reducing the number of seats the larger parties attain. Understanding the difference between the vote is crucial for tactical voting.It is also important to note that the Senedd and Election Act of 2020 state that 16 and17-year-old's will now be able to vote in Senedd Elections from 2021. Which will "introduced the largest franchise extension in Wales since 1969" [6].Meaning this coming year is when most people are going to have to be educated on voting since the AMS was introduced.When googling 'How the Senedd is elected?' The first result is senedd.wales [7]. The page clearly and concisely explains how a 'registered voter has two votes'. Going as far as to have several example equations demonstrating the system Figure 1.1.As a resource for my initial document, this couldn't be better and in terms of bare-bones education on how the system works, it is also effective. But this project aims to produce an auxiliary online tool to help voters better grasp the full effect of their vote in the Senedd Assembly Election. The second benefit of voters having a greater knowledge of the power of their vote is to increase voter engagement. Valid vote turnout in 2016 was45.4% which is still less than the 46.45% in 1999, showing it is still an ongoing battle [8].

The calculation for the first seat would be as follows. The total number of votes for each party is divided by their division total:

	Party A	Party B	Party C	Party D
First additional seat	50,000 ÷ 4 = 12,500	62,000 ÷ 4 =15,500	48,000 ÷ 3 =16,000	36,000 ÷ 1 =36,000

Figure 1.1: How the Senedd [7]is Elected Example graphic

1.2 Objectives

The main overarching goal is to educate a user on the difference of vote impact in regional and constituency votes. But to accomplish this, a model that can accurately determine results from a set of votes, will be used. The model needs to be highly interactive; allowing users to see for themselves how changing their votes will affect elections results. For the model to be useful as an educational and scientific tool it must be able to handle all current regions and constituencies, not just modeling one example region. This level of granularity is needed so that user can learn in the context of there own region. It will also mean that the model will be usable to model polling data across the country, as the data might only be there for certain regions. Being able to model polling data is important as it can tangibly showing what is happening in the moment and what role changing there vote can play in the election.The model will also need to produce a brake down for every region and constituency and not produce a country wide result.

But for this model to have any educational use, it must be accessible. Accessible in the sense that users can physically access the tool , but also being intuitively usable once they have that access. Making the model a web application will solve the physical element. In terms of user accessibility, a list of regions and constituencies isn't useful; new voters might not be able to name there region and constituency from off a top their heads. This is why an interactive map of Wales will be used to display the modeled data. The map will be broken down into its 5 electoral regions and further down into its 40 constituencies. So as to reflect the model. When a constituency is selected a user will be able to adjust what percentage of the vote a candidate has. They will also be able to do this for their region. The application will then update showing what effects this has. Another objective is that voters and party members alike should be able to better understand the results of previous elections. This means that the application should come with previous election data built-in. But displaying the results is not enough to properly understand the impact of the system, it is important that that data can be manipulated. To show how much of a change is needed in the percentage of the votes a party gets for there to be a change in the number of seats that that party receives. As noted before in 2021 16-year-old's will be now eligible to vote and this project objective to get that age range actively engaging with the election. This means not just watching passively the Senedd YouTube videos on the topic but actively engaging in every way they can [9]. As the video just gives the theory behind the system and not what that means to their vote. This objective is hard to quantify but it means to have as little information explained through plain text as possible, rather having them learn

through playing with the tool and having it accessible online so that it can be easily shared.

1.3 Listed Aims

- **Accurately model results given voting data**

- **Store of a repository of historical election data**

- **Parties, Constituencies and Regions have dedicated pages displaying the models results**

- **Display the model's election results with color-coded interactive map**

- **Allow users to edit election data that the model can use**

1.4 Overview

Chapter 2 will outline how the Senedd Assembly's seats are assigned, along with discussion about already established modeling systems and educational tools in this space. In chapter 3 will touch on how the development cycle has radically changed from the plan outlined initially. Followed by an in-depth description on the evolution of the database and other fundamental aspects of the web application. The 3rd chapter goes into how the models objectives will be tested, and what the results mean in the context of the discussed motivations. The final chapter will reflect on the project as a whole, while also giving a clear set of steps which should be taken to further the project.

Chapter 2

Background Research

2.1 Similar Products

There are several websites that specialize in the display of election data. The main ones of interest are the Senedd Business [10] pages and BBC's Wales election pages [11] two products that display historical election data.

2.1.1 Senedd

The Senedd Business [10] pages are the Senedd official results pages. Throughout this process I was primarily sourcing my election data from the Senedd Business pages. Senedd Business pages contain a few extra pieces of data that aren't important for modelling elections; such as spoilt ballots. This data isn't displayed in my application. The Senedd Business pages, however are extremely hard to navigate. Navigation is made so arduous due to constituency data being kept so completely isolated from regional data.To access a specific constituency, from that constituency's region page, a user has to navigate to a homepage that lists all election data stored by the Senedd Business site, then to a page that summarises the constituencies results, then to a page listing all constituencies, and then and only then a user can choose their given constituency. Link have a peculiar habit of sending a user to 404 error pages [12]. Candidate links are the most egregious for this. I also discovered in the testing stage of my project data was missing; in 2007 the South Wales West Region, didn't elect any candidate.

2.1.2 BBC

The BBC's [11] 2016 pages fair better than the Senedd Business pages. They are considerably easier to navigate, facilitated by the color-coded interactive map, links that's allow trivial switching between regional and constituency pages. Alongside the raw data they display in myriad of insights on the results, including what parties have lost of gained seats in this election. Similar to the Senedd business pages that have dedicated pages each constituency and region in a list of candidates however the BBC's list far more complete.

2.1.3 Limitation

All websites of this kind, which specialize in the display of election results, will have the same limitation. A limitation that my application hopes to solve; which is that the results are not modeled. They just display results passively. There is no user interaction with the data and the results are recorded rather than calculated. This is perfect for these pages use case; which is to quickly inform a user who wants to briefly check certain result. But does not serve our stated goal for education. another common limitation put these separate regional and constituency data. Both sites have dedicated pages for Regional and constituency results, that not share any data.This is even true Wikipedia [13]. This is where the design of my pages are able to highlight and compare regional and constituency results directly, as shown in Figure C.2. As the results in a constituency have a direct impact on the regional results it is vitally important to highlight this connection. There final limitation is that they all have separate historical results. while the BBC pages do tell you whether or not a party has lost or gained seats. If a user wanted to check this for themselves they would have to leave the BBC's site, where they would be forced use google to navigate to there completely separate results. It would just so happened that those results would also be in a completely different format.

2.2 Related Work

2.2.1 Election Forecasting

Reach into election forecasting is primarily broken down into four groups:Structuralists, Aggregators, Synthesizers, and Judges [14].

2.2.1.1 Structuralists

Structuralists use external factors such as economic indicators. Then with those indicators, they predict support for a party. This estimation is made using ordinary least squares (OLS) on single equations [15]. OLS is also known as linear regression. This statistical modelling method is incredibly popular to the point that it is one of the most frequently used. (Linear regression, 2020) because the data they used is long-term data structuralist models tend to lend themselves to comparing elections against each other [16]. This method had for quite some time been the most popular for forecasting European elections to the point of exclusively [15].

2.2.1.2 Aggregators

As the name just aggregators suggest this model collect sources of data. Mostly collecting and interpreting polling data. [16]. While structuralists are long-term data and create relatively fixed decisions, aggregators are constantly collecting data throughout the election period to update the forecast appropriately. [15]

2.2.1.3 Synthesizers

Is the combination of tools and aspects of both aggregators and structuralists. The process begins with an economic model taken from the structuralists, then throughout the election period, this is supplemented with aggregation. This method unlike both the structural and aggregate model can be implemented for forecasting on a state level, not just a national level. Unlike European forecasting, synthesizer and in fact both structuralist and aggregator models have been used to forecast the United States election for quite some time [15].

2.2.1.4 Judges

This model is designed initially for state-level forecasting. It uses both qualitative and quantitative analysis combining the two to create the final forecast. On the quantitative side, the analysis will use any data it can get it's hand on and quantify. A list that6 includes and is not limited to: "population demographics, historic and recent voting patterns in states and districts" [17]. When all of the collect data has gone through the analysis, a primary forecast is made. The primary forecast is a stepping stone to which qualitative analysis can be added too. This is the stage where political scandals are taken into account. Whether or not there is an active incumbent and much more. Once this has been processed a "Judge" will continue to

follow the election and stay in contact with associated political groups on either side of the election. As this is mostly a forecast model which is used in the United States. When the Judge has taken all that data under consideration, they then can speculate who might win [17].

Chapter 3

Project Management and Development

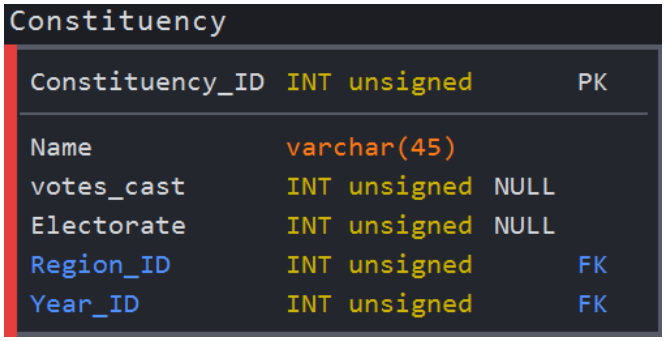
3.1 Tools

3.1.1 Laravel

It is of particular importance to outline how Laravel [18] works, so that my implementation can be understood, along with the changes to my initial development strategy.

3.1.1.1 Migrations and Models

Together migrations and models make up the database structure and implements relationships for that structure. Each table gets it's own migration and model file. Primary and foreign keys are specified in the migration file along with the name of each column and what type of data is stored. Models come in when implementing the relationships in the database. When two tables have a relation, a function needs to be added in each table's model. This function will specify what type of relationship is being used; such as a one to one relationship. Constructing the database in this fashion means that accessing and manipulating the database can be done in a manner that more closely resembles object-oriented coding; a style I am more comfortable with.



A. A diagram of the final Constituency Table

```
public function up()
    {
        Schema::create('constituencies', function (Blueprint $table) {
            $table->id();
            $table->string("name");
            $table->Integer("votes_cast")->unsigned()->nullable();
            $table->Integer("electorate")->unsigned()->nullable();
            $table->BigInteger('region_id')->unsigned();
            $table->BigInteger('year_id')->unsigned();
            $table->timestamps();
            $table->foreign('region_id')->references('id')->on('regions')->
                onDelete('cascade')->onUpdate('cascade');
            $table->foreign('year_id')->references('id')->on('years')->onDelete('
                cascade')->onUpdate('cascade');
        });
    }
```

Listing 3.1: The implementation of the above Constituency Table in PHP

Figure 3.1: Image A shows the desired table layout, who's implement in Laravel with PHP is demonstrate in Listing 3.1 below

3.1.1.2 Views and Controllers

Controllers get past data from a route, such as a specific year in the case of a YearController calling the view function. This functions job is to query the database to get access to the data a view is going to require and then format that data. The data should be formatted in a manner that requires the view receiving to do as minimal processing as possible. As the view's role is to display the data. This is achieved with a mix of HTML, JavaScript and Blade [19], which is a templating engine.

3.1.2 Leaflet

Leaflet is a JavaScript library for interactive map's, which they specifically stat are mobile friendly [20]. Which I was looking for as to help with the accessibility of the application. The

main functionality I used was the Polygons, you can give then a boundary in the form of a latitude and longitude 2D array, and they will display that shape on top of there maps. Whats more you can easily change the colour and opacity. There interactive maps use OpenStreetMap [21] data, they look very similar to Googles maps.

3.1.3 Web Scraping

At the outset of this project I had no intention of using a web scraper, but due to factors outside my control, which will be discussed in section 3.3, I ended up using one. As this web scraper was used for collecting election data and didn't have to directly communicate with my web application, meaning I was free to choose what ever language I deemed most appropriate. This ended up deciding upon Python. I'm reasonably familiar with Python, finding it natural to quickly prototype type with; which was necessary in this use case. Whats more Python comes with a large selection of pre-installed standard libraries, one of which was urllib. This allowed me to get the content of my chosen web-page in a format I was familiar with out install anything extra. Listing 3.2 show just how simple it is to print the content of a page.

```
1 import urllib.request
2 if __name__ == '__main__':
3     page = urllib.request.urlopen("http://seneddvote.test/years/3/constituencies/
          Delyn")
4     for line in page:
5         print (str(line))
```

Listing 3.2: Code in Python to print the contents of one of the web page produced by my model

3.2 Original Plan and Modifications

As seen in Figure 3.2, my original plan was going to use the waterfall model. At the out set this seemed to be the optimal model, due to the fact that Laravel compartmentalizes different aspects of the application very effectively. You could code the migrations and models to create a fully working database, with out touching the front end views once. This compartmentalizing meant that I naturally thought to break down the project into stages based around thoughts sections. Which was to code the database fully, then move to the controllers and finish up with the views. Each of these steps would require a different skill set; allowing me to focus on one discipline and language at a time, not having to flick between JavaScript and database management. But once the database was implemented and I had moved on to implementing the

controller, I discovered it was inefficient and complicated to work out how to access and format the data . The database was made so cumbersome for the controller because the waterfall model dictates that you should plan everything out at the beginning, as I had little experience with Laravel's controllers and even less with database design, it meant the original plan was always going to have to be improved. Whats more I hadn't started work on the view either, so I didn't even have a clear idea on how my data should be best formatted.

Once it became apparent that it would be necessary to update the database design, I decided to switch to an extreme programming framework. Where I would focus on making the year view first. This is because the year view would contain all the information about an election's results, meaning all subsequent views would just contain a smaller aspect of the database, but in more detail. So once the year view was made; it would make the construction of all future controllers and views more convenient as there would be an outline of what would be needed to be produced in the year controller. What is more it would reduce the chance that I would have to go back to already implemented controllers and update them, due to a change in the database.

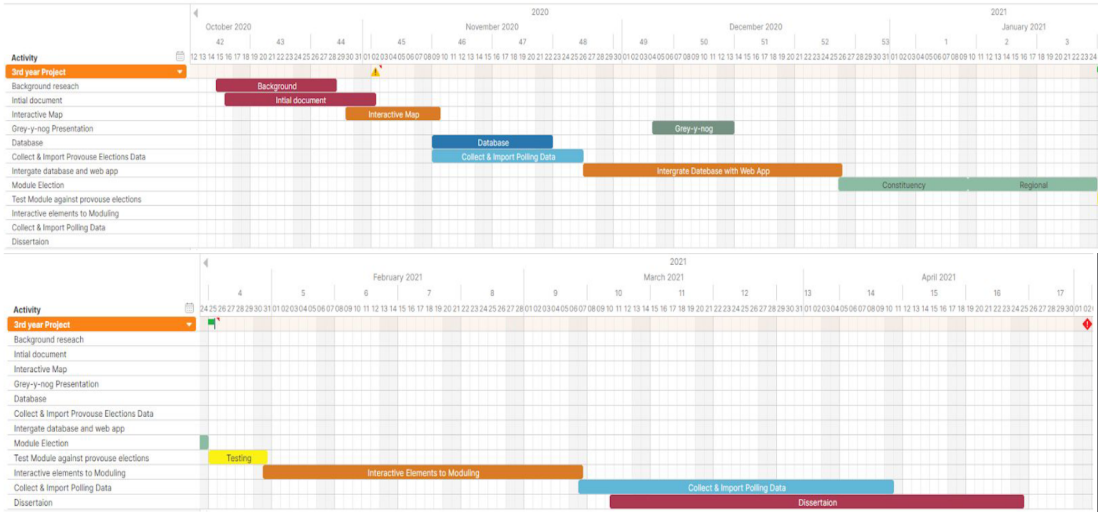


Figure 3.2: A Gantt chart for Terms one and two, as originally outlined in the initial document

3.3 Database Seeding

3.3.1 Seeding Test Data

At the beginning database seeding was just the simple act of making a handful of regions and constituencies. This was done so that I was able to prove all relations functioned properly, and to give the views and controllers data that they could interact and display with. This was achieved with simple loops so that the quantity could be changed at will. In most cases a factory would be use for this type of task, as I have done so in the past, but I have chosen not to use factories here for two reasons. Firstly factories mostly use Faker [22]. Faker will used to create a set amount of pseudo realistic data for a seeder. But because this database is used for the modeling of an election, I thought it wise to use consistent data so that I would reliably know the expected outcome. This allowed me to quickly test my model, every time I seeded my database with out having to check randomly generated results. Secondly Faker is exceptionally adept at generating items such as fake emails, fake passwords and set lengths of pseudo text.These are all useful if you are developing an application, which will have many users who all generate comments who's content is irrelevant. This is far from the case in my implementation of an election model as I only need a handful with fake names for parties and set number of votes all of which easily done by hand.Faker is made further redundant due to my application being focused heavily around the interactive map of Wales whose test boundary data, is far out side it's scope.

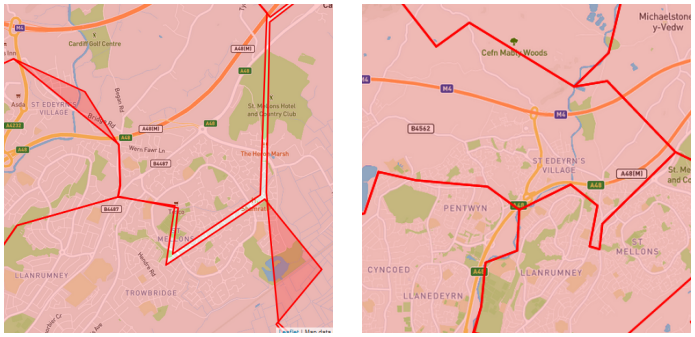
3.3.2 Seeding Historical Data

The two major challenges when seeding historical data. One was locating a source that contained adequate data to produce accurate results, which also provided the supplementary educational information I would need to fully realise the application. This included all candidates running, party images and colours. The other was formatting the raw data that I sourced into functions that the seeder would be able to understand and populate the mySQL database with .

3.3.2.1 Map Data

I Required data for 5 regions in 40 constituencies, so that they could be plotted on the Leaflet Maps. it was vital to set up the map as early as possible in development; being the main tool used to display data and navigate the website. It was challenging to find data to serve this purpose. I tried a myriad of websites, some like Senedd Wales [23] which had wonderful

data, but only in the form of PDF images. A format impossible to export for my purposes. The closest I came was OpenStreetMap [21] data. There boundaries broke down Wales in 22 principal areas which did not match either the regions or constituencies. I did however come across the Office for National statistic's' [24] pages but failed see they had the boundary data I need and not just extraneous data like population size so ignored them. This was a huge mistake.At this point I was worried that not being able to acquire these boundaries would create a bottleneck; creating a knock-on effect on the whole development. I elected to make my own temporary boundary data so that development could continue. To achieve this I made a JavaScript web page that allowed a user to draw a polygon on a Leaflet Map, then click a button which would output the latitude and longitude in a 2D array. A demonstration of this tool can be seen in E.1. This data was never intended for the final product; it was horribly inaccurate, having multiple areas that overlap with each other and large gaps as seen Figure 3.3. Which is not to mention how it could mislead the user into believing they belonged in the wrong constituency. But at the time I thought I had no choice, and fully implemented the model using these boundaries. I eventually discovered the accurate boundary data that the Office for National statistic's' [24] had. Importing the data into the database I found it increased the seeding time for the coordinates from 70 seconds to 771 seconds. This wasn't a concern for the final product as the seeder would only have to be run once when deploying. However using my agile development cycle, I would have to regular reseed the database, creating another bottleneck. As the testing data was accurate enough to tell constituencies apart as seen in Figure C.1. I chose to keep utilising the temporary data until I was convinced the database was finalised, and would need no reseeding. Where I could import the historically accurate boundary data and reseed just once more.

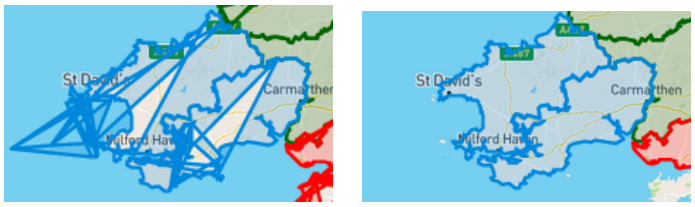


A. Temporary inaccurate map data with gaps and over lapping sections

B. Permanent Historical Data

Figure 3.3: Image A show the temporary map data used during development, short comings, While image B shows the permanent data that replaces it.

Eventually I located the accurate boundary data on the Office for National Statistics pages [25]. Where there API produced GeoJSONs; which was something I was looking for from the beginning. However the GeoJSONs formatting had a few peculiarities, that my test data had an accounted for. The latitude and longitude was stored in a 3D array, not a 2D array due to the fact they had multiple boundaries per constituency/region to account for islands. Something which I had neglected to do. This meant my database and controllers gave each region and constituency only one polygon each.To convert the GeoJSONs data into a format my seeder would understand I wrote a Python tool. That read in GeoJSON files, then output a text document, where each line was the name of constituency/Region and their longitude and latitude in the form of a 2D array. My first attempt to turn the GeoJSON's 3D array into a 2D array, was to concatenate the array into one longer 2d array. this was done when converting the string given by the GeoJSONs to an array.This had the effect of combined all the island boundary data with the main body of the polygon. the results of this plan can be seen in Figure 3.4A. My next change was to just drop the extra islands. Then I manually added one island to the North region, in the correct location within the boundary array so that it would render normally seen in Figure 3.4B..



A. Islands Causing Errors in Map Data

B. Fix Errors in Map Data

Figure 3.4:

Once the historical data was correctly formatted and ported into my seeder, another problem arose. User feedback at this stage came back, stating that the pages where loading too slowly. One user saying it was "epic-ally slow to load". This was valid criticism, with the main year page for 2016 taking over 13 seconds to load.

My first thought was the the region's boundary was needlessly precise, which was creating these load times.So I added a function in my Python GeoJSON converter, which would strip N numbers of latitude longitude pairs out of the data in regular intervals. This meant the general outline of the boundary would stay consistent whilst being a little less precise. The stripper function would also keep the first and last latitude longitude points no matter what. This allowed the polygon to connect up in the same place. The python GeoJSONs converter, would take the GeoJSONs files and then output a RegionSplit.txt file, which had the smaller data set. I then used another simple leaflet test page to display the cut data. Allowing me to see if the data was still usable, with out the need to reseed the whole database.

This fix did have the effect of reducing loading times, but just not significantly enough to justify the loss of data. This meant a better solution had to be generated. The real problem that was causing long loading times, was due to my database structure. I was trying to imitate a 2D array, with relationships inside mySQL. Each boundary would be made up of 100s of co-ordinate entries each having a relationship to a region. The change I implemented was that the latitudes and longitudes would be stored as a single string. Each region would have only 1 co-ordinate linked to. That co-ordinate would have a large string then could converted later. Instead of one pair of latitude and longitude. However once this was implemented a new problem with the data was found. The GeoJSONs stored latitudes and longitudes in the opposite order to Leaflet. Meaning when the data was imported; regions where placed in the wrong location and stretched. I had fix this problem earlier, as I had to split up the latitudes and longitudes up when making the co-ordinates. But this wouldn't work now, because I was using

JavaScript's Eval function to convert the string. I needed to reformat the string before it was imported into the database. So a final function was added to my python tool. Which read in GeoJSON's, then took the latitudes a longitude strings and swapped the latitudes and longitudes order, outputting this into a final text document. Once this change was implemented seeding the co-ordinates which took 771 seconds before now took 11. The year view before and after this fix looked identical yet the loading time dropped by 10 second. At this point the database still has a many-to-many relationship between coordinates and constituencies/regions and it's strictly speaking not needed as each constituency/region only has one polygon attached to it now. But as I had stripped away the island data, I'm keeping this many-to-many relationship so adding the island data back in can be done with ease later on.

3.4 Controllers

3.4.1 Year

3.4.1.1 Objects

Though out making the Year controller several key changes where made to the overall implementation of the controller. First of which was I stopped passing region data as a Region object. When passing a Region object excess data was also passed, what's more , information may have been calculated in controller which needs to be associated with the region. But can't be as it is just an object. To solve both problems at once I used associative PHP array instead. These arrays can be accessed in an almost identical way to the Region object as seen in figure 3.5, but you can added as may element as you want. Using these arrays I would bundle together all the data needed to display certain aspects of the pages, the such as the regional seat as seen in Figure 3.5.

```
1 $region->id;
```

Listing 3.3: Show how to accessing a Region object's ID

```
1 $region['ID'];
```

Listing 3.4: Show how to accessing a Associative Array

Figure 3.5: Demonstrating the similarities between accessing a Region object and similarly formatted Associative Array

```
$seats, ['id' => $seat->id, 'colour' => $colour, 'partyName' => $winner->party->
    name, 'partyImage' => $winner->party->image, 'partyID' => $winner->party->id,
    'repName' => $winner->name, 'repID' => $winner->id]
```

Listing 3.5: Formatting data to displayed a modelled Seat

3.4.1.2 Query

When I first implemented the controllers, the queries that I used were horrible inefficient. As seen in 3.6. what makes queries like this inefficient is that they are querying the entire constituency table, meaning they will be slower the larger the table gets. however using an object's relationship functions and then querying the results is far more efficient. As you are just dealing with that objects relationships. So it doesn't matter how big the rest of the database becomes you are only. They are far easier to read as well.

```
$constituency = Constituency::get()->where('seat_id',%seat->id)->first();
```

Listing 3.6: Inefficient querying of a database

```
$constituency = $seat->constituency;
```

Listing 3.7: Efficient querying of a database

Figure 3.6: Demonstrating how I change the way I queried the database

3.4.1.3 Map Change

The Old implementation of boundaries was slowing down loading, because the controller get boundaries to each region in the form of a 2D array. this meant querying every constituency and region for all of there coordinates , then each access the latitude and longitude. combine them into an array and then add that array to a larger boundary array. On a low average each constituency and region had 500 points to make it's boundary as there are 40 constituencies and 5 regions that is 22500 iterations. what makes it worse is querying , and accessing the database, can be resource-intensive. Meaning this does explain the 13 seconds of load time. when plain simple strings where used instead of, this pseudo 2D array with coordinates tables. the number of iterations needed went form 22500 to 45, and what is more in each iterations it was only accessing one element in the columns not two.

3.5 Views

A view can inherit other views. layout.app is the parent to all other views, and contains the navigation bar, allowing users to login and out at will, as access all historical data. As one of the main objectives is to allow users to directly compare regional and constituency results, most views where going to need to display two maps. So two, two map views where created. One that displayed maps side by side and the other on top with data down the side. Years inherited the side by side view as shown in Figure C.2 and party, regions and constituencies used the side by side view in C.4. Having the double map layout, meant that whenever I was creating a view that needed some maps. That view would just contain the data that need to be displayed, and the JavaScript to display the map its self was kept out of side and mind in the layout it inherited.This also meant all my web pages had a homogeneous feel to them as seen in Appendix C. Displaying most of the data outside the map, I elected to use tables as they were easy to populate and scale.

3.5.1 Region and Constituency

The Region and Constituency views are almost identical, just receiving different data from the controllers and flipping the map order. Which due to the fact they inherited the same map layout, meant just changing the section's names, change the order the data is displayed on the maps. Most of the display code is in the same structure as the Year view mostly filling in tables. However there where some changes. The maps needed to be altered, so they would zoom in to the correct area. This was done by centring both maps around the regional polygon bounds. This meant that both maps where zoomed and centred around the same point and looked consistent.

3.5.2 Party View

The only implementation of note in the party view was that I altered opacity of the polygons, this was so that the winning constituencies stood out.

3.6 Routes

I haven't touched on routes too much before now, as the implementation of routes with out the concept of a user with in the web applications is relatively straight froward. I have yet to

implement users beyond what is achievable using Boostrap [26]. Where using the command shown in 3.8, will install basic views, migrations and JavaScript, that will allow a user to create an account, that is registered with an appropriate email and password. All the basics of logging in and out is completed and the framework for password resetting is installed. But as I am yet to have any level of user hierarchy; allowing any user to access any page the routing remains simple. However there are some small additions I felt necessary.

```
1 php artisan ui:auth
```

Listing 3.8: Bash command to install Boostrap's authentication framework

The first of which is the indexByYear route added to both regions and constituencies. Using the address in Figure 3.7, will retrieve a view listing of all constituencies for the given year. This index view can be far more useful than the normal constituency index view as the normal index view will display all constituencies in the database.

http://seneddvote.test/years/3/constituencies

Figure 3.7: Web address that displays a page listing all constituencies for the year with ID 3.

The second more complex addition was the showByYearName route I also added for regions and constituencies. Which will allow a user to use an address like this:

http://seneddvote.test/years/3/constituencies/Delyn

A. A diagram of the final Constituency Table

```
1 Route::get('years/{year}/constituencies/{name}','
      ConstituencyController@showByYearName')->name('constituency.showByYearName');
```

Listing 3.9: The implementation of the above Constituency Table in PHP

Where for a given year id they can search for a region/constituency by name. This means they would not have to memorise the IDs for constituencies or regions. This would facilitate much faster navigation by an advanced user. It would also make automatic testing of the website easier as you don't need to keep track of each constituency region ID across the year but just a name.

The new route would give the region/constituency controller' s showByYearName function, a year object and a string. This function would then query the year object, for the desired region/constituency using the string. Where it passed that object to the standard region/constituency show function. this meant no repeat code had to be written. As it just utilised the existing code, this principle was also applied to the partyByName route.

3.7 Database

The final iteration of the database is reflected in figure 3.8 diagram. Throughout the project all the catalogued changes to the database , so that it could better accommodate the views, controllers and new data, is displayed in appendix D. This figure will give you a excellentover view on how the project has evolved.

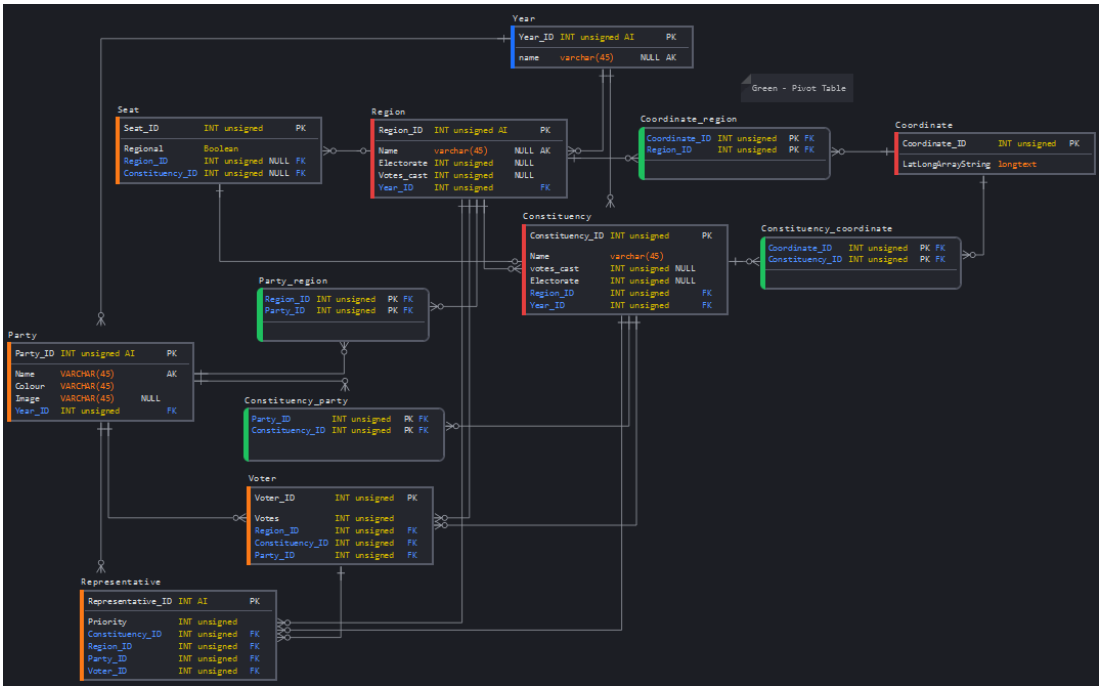


Figure 3.8: Shows the final iteration of the mySQL database user to record historical data [27]

Chapter 4

Testing and Analyze

4.1 Testing

Section 4.1.1 looks at the collection of test data for the subjective elements of the web application. This includes items such as ease of use and readability. While Section 4.1.2 looks at testing the objective components of my project, which is the accuracy of the model.

4.1.1 Subjective Testing

Before I could get any feed back on the web application, a user would have to be able to access it. Due to the fact that I have developed the application on a Homestead [28] oracle Virtual Machine [29], the application, was only accessibly from my own machine. My first idea was to make it accessible form any where on my local network. This was in part because I knew how to implement this. But there are two problems with this approach. One is that having it accessible only on a local network is fine in an Covid free environment, as I don't have that luxury the site needs to be accessible form anywhere. The other problem was using this method doesn't allow mobile devices to access the sight due to a the phone try to access port 8000 on it's self [30]. There are fixes to this but I had trouble with them. Whats more is that I felt that it was vital to test the usability of the application of on a mobile device. As 50.88% of web traffic is on mobile device [31], a number that I have assumed is higher in young adult ages 16, however I have been unable able to confirm this, which is a key demographic that needs educating in the 2021 election; being the first year they are able to vote.

I didn't choose to buy a web domain and host the site, due to security , and financial concerns. As in it's current state I have not specifically taken into account security concerns,

though I have of course been coding with it in mind at all times. The other problem with having it hosted online is the modelling and data has yet to be tested, and I do not wish to miss inform the public. This left me with NGrok, which comes as standard with Homestead. NGork an create a secure introspectable tunnels to localhost [32], in just one single command listed in 4.1 once set up. This allows me to 'host' the website quickly for short period of time within unique address. My first tested NGork I found none of the links or images where functional. this was due to the way links are made with Laravel, as the base of the link as always sennedvote.test. this needed to be replaced with the custom address given my NGork, this was done with this line of code.

```
1 ngrok http -host-header=rewrite seneddvote.test:80
```

Listing 4.1: Creates Tunnel to my localHost that as the model stored on.

[language=php, label=lst:ngorkRouteFix, caption=Allow links and Images work when tunneling] ./listings/ngorkngorkRouteFix.php

Once the tunnel was set up I used it consistently, throughout agile development regularly testing the site on mobile devices, but also have selected users play around the site alongside me. So that I could see how they use the site and could give me immediate feedback back. I was always looking for specific feedback in terms of the models implementation. Seeing if the data was conveyed correctly, I did not use a questionnaire as my test group was so small, one-on-one chat sufficed in most cases. I also made sure the range of applicants was diverse form 20-50+

4.1.2 Objective Testing

4.1.2.1 Historical Testing

Initial Test I found my self in the unique position of having a large number of real life results. So I chose to test my whole site against the real life historical results. Getting historical result however proved a problem. I modified my scraper, which I used on the Senedd Business pages, to get the Regional winners in each year, and exported them into text files. I found that the results where incomplete as in 2007 South Wales West elected no one. As this wasn't an issue with my scraper but the site data it self, I increased my testing scope so that it would also include checking if the Vote count was accurate with other sources. In case all the data from that site was corrupted. Due to it's formatting I used the data from Wiki's pages on Senedd Elections [13] but who's result I crossed referenced with the BBC's website [33]. Using the

template shown in Figure 4.1 I tested 2007, 2011 and 2016 as these where the only years with the additional member system. Full results can be seen in appendix B. As the model failed so early on in the testing I stopped after testing the year page. Looking at this test, it showed that my implementing of the D'Hondt formula was inaccurate there were discrepancies don't count for some business pages. however these discrepancies would quite small so I ignored them at present.

2016												
	Historical Results				Model Data				Test			
Totals:	1903274	20	40	60	1903271	20	40	60	FAil	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	673062	2	27	29	673061	6	27	33	FAil	FAIL	PASS	FAIL
Plaid Cymru	420924	6	6	12	420922	4	6	10	FAil	FAIL	PASS	FAIL
Conservative	406443	5	6	11	406443	4	6	10	PASS	FAIL	PASS	FAIL
Liberal Dem	143669	0	1	1	143669	1	1	2	PASS	FAIL	PASS	FAIL
UKIP	259,176	7	0	7	259176	5	0	5	PASS	FAIL	PASS	FAIL

Figure 4.1: Showing the first Testing done on 2016 result

Fixes Looking at my implementation of the D'Hondt formula I found that it had a logic error. The model would in the first round of the D'Hondt formula give out a seat when ever a party with a higher vote was found. Not calculating who has the most votes in the round and giving them the seat. Once that was fixed and another round of testing was done, I found my model still failed. Doing additional research found that the Senedd election use a modified D'Hondt formula. In this formula the constituency seats already awarded where taken into account.

In the modified D'Hondt formula the seat count for each party is waited at the beginning equal to number of seats already rewarded in the constituency race. Then using this weighted seat number the D'Hondt is run normally. You can see the implication of this new formula use in the region controller in Listing A.1.

But has the constituency results need to be calculated before the D'Hondt formula can be run; changes to all other controllers needed to be made. The regional controller was a simple fix as it already calculated the constituency results, as it had to displayed as supplementary information. However ordering had to change and the calculated constituency results had to be passed into the new the D'Hondt Formula. The Party controller was the most complicated. As in it old implementation it only kept track those constituencies the party we running in. But for each region the party ran in, all constituencies in that region need there results calculated then fed into the model. however those calculated constituency results should not get past into the view as the view only wanted the results for constituencies they were running in. For the

constituency view, I chose not to calculate the regional results. This was due to the feed back from users. The only impact regional results had on the constituency view was the regional polygon but as you said said colour which hard to understand I just not calculate the results at all. this increased the loading speed whilst not reducing any usability of the site.

2016												
	Historical Results				Model Data				Test			
Totals:	1903274	20	40	60	1903271	20	40	60	FAil	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	673062	2	27	29	673061	2	27	29	FAil	PASS	PASS	PASS
Plaid Cymru	420924	6	6	12	420922	6	6	12	FAil	PASS	PASS	PASS
Conservative	406443	5	6	11	406443	5	6	11	PASS	PASS	PASS	PASS
Liberal Dem	143669	0	1	1	143669	0	1	1	PASS	PASS	PASS	PASS
UKIP	259,176	7	0	7	259176	7	0	7	PASS	PASS	PASS	PASS

Figure 4.2

Robustness and Page Testing

As the implementation of the formula facilitated changes in not only the year controller but the regional and party controllers as well. I needed to updated the testing to assure robustness. I needed to check if model was uniform across the site. however is regions and parties all use the same controller, I didn't feel it was necessary test every individual region and party. o for each year two parties in regions where chosen to be tested. I make sure that chosen test cases would cover as many regions as possible whilst testing outlines in the data set. Figure B.1 shows my tests results. For testing the regions the easiest way was to compress the results into a string that could be compared directly. Where each letter would represent a party. Once is these pages with tested that I was confident that my data accurate across the site. I only did one robustness test and that was for a party with no votes, the model worked. However I was unable to test for a tie as in the state that a tie happens a candidate is chosen randomly, this mean tie must be recorded and not modeled as a new winner could be chosen every time a vote is called.

PARTY PAGE TEST

2016			
	Historical	Model	
Party	Region Seat(s)	Region Seat(s)	Test
UKIP	7	7	PASS
Labour	2	2	PASS

2011			
	Historical	Model	
Party	Region Seat(s)	Region Seat(s)	Test
Green	0	0	PASS
Cymru	6	6	PASS

2007			
	Historical	Model	
Party	Region Seat(s)	Region Seat(s)	Test
Democrats	3	3	PASS
Conservative	7	7	PASS

A. All Party Page Testing

REGION PAGE TEST

2016			
	Historical	Model	
Region	Region Seat(s)	Region Seat(s)	Test
Mid & West	L,L,U,P	L,L,U,P	PASS
South Central	C,C,P,U	C,C,P,U	PASS

2011			
	Historical	Model	
Region	Region Seat(s)	Region Seat(s)	Test
North Wales	C,C,D,P	C,C,D,P	PASS
South Wales East	P,P,C,C	P,P,C,C	PASS

2007			
	Historical	Model	
Region	Region Seat(s)	Region Seat(s)	Test
South Wales West	P,P,C,D	P,P,C,D	PASS
Mid & West	L,L,C,P	L,L,C,P	PASS

B. All Region Page Testing

Figure 4.3

Chapter 5

Conclusions and Future Work

5.1 Conclusion

The project has fulfilled the majority of its aims. Testing has proven that given voting data it will accurately model all regional elections. Clearly fulfilled the first stated aim layout at the beginning. However users are unable to enter there own data in it's current implementation. as the database has been fully implemented can interact directly with the model a base of functionality has made which will allow using implementation user editing. it would just be a case of letting the user edit the database candle modelling with when the page is displayed In terms of the web pages themselves, all of the aims have been achieved; the map is fully implemented, the pages are natural to navigate and have a homogeneous feel and all of the data stored in the database is efficiently displayed. Ever since the quality of life fix, which drastically increased the map loading speed, navigation using the map has felt great. This was a high priority in my implementation, as once the user is given the power to edit the database, the modelled results will be calculated instantly. A crucial step that makes the educational tool viable. As it will allowing users to play around with the data as much as they want not getting discouraged with long loading times. However the pages are far from perfect. Feedback obtained during the agile development, showed clarification is needed in areas, such as a key for the map. But this alludes to a larger problem; although the main the aims of the web application have been achieved aspects of the overarching objective have been neglected. The data which is generated and then displayed, is it's self a fantastic tool and prop for a user to wield when teaching them selves, but does minimal amounts of teaching one its own. Apart from the raw data which is neatly formatted, the pages themselves are quite barren of insights. A significant

amount more can be done to guide and educate a user within the pages themselves. In the form of extra data generated outside what is needed for the model, and small facts and incites can be easily written about for the historical election. However this minimal approach has one benefit the application displays wonderfully on mobile devices, which contributes to the accessibility of the application which is a major objective for this project.

A significant reason that one of this project's major aims, user database interaction, went unrealized was due in part to significant wastes. Born from poor project research and planning form the outset. Selecting the waterfall development strategy at the beginning, leaded to a liberal chunk of time being spent on fully implementing a database that needed multiple complete remakes. however this is expected in development. The more significant waste of time was due to neglecting proper research into the tools and data that where available to me at the beginning. As the Senedd business [10] pages had a laundry list of floors and I should have used the Office for National Statistic's [25] API form the very beginning. Which would have drastically reduced the time spent formatting and importing data allow me to spend more time on other aspects of project

As it stands the modelling portion of this project is complete, and able to interact completely with the finished database and views. Meaning it is fully functional as a repository of historical Senedd elections. The application is also far more convenient to navigate, and importantly more accurate than the Senedd business pages [10]. due to the interactive map and functioning modelling system, which displays the results for all regions. A case could be made that my application does a better job than the BBC pages, but it wouldn't be an overwhelming argument. The BBC's pages also have an interactive map, and has far more insights. However this is only due to the unfinished nature of this project. As a whole the project is a solid foundation, that has been designed so that a large amount of extra functionality can be build from it.

5.2 Contributions

The three main contributions of this project can be summarized as:

- **An Educational Tool**

  This project functions as an online educational tool, designed to specifically examined the impact of the additional member system on regional and constituency votes

- **Election data Repository**

  The application functions as a repository for every Senedd vote that used the additional member system. Providing all the same data that the Senedd business pages do, but in a easier to navigate format, and with correctly modelled election results

- **Election Model**

  It provides a scientific tool, which can model the results for the Senedd election, given voting data. Which can be in the form of polling votes, historical votes or custom data votes.

5.3 Future Work

- **User Database Interaction**

  This is the most important and also the hardest out of the next steps. Luckily it is not too hard to accomplish due to the current implementation. The basic idea of what needs to be added is that; a user should be given a page where they select a election year as a basis. Once selected they should be able to change the votes for any party in any constituency and region. Then that data should be imported to the database. Then a user can simply go to the view for that fake year, and be able to see the newly modelled results.

- **Display Island Polygons**

  As stated before I am not using the full boundary data, taken from Office for National statistic's [24], as I'm neglected to use the data for islands. This needs to be fixed, and the steps to do so are simple. As there is still a many-to-many relationship between coordinate and the regions/constituencies. So I just need to seed the islands as a new coordinate that is related to it's given regions/constituencies, and change the controller so instead of taking the first instant of an coordinates it takes them all and iterates though them.

- **Extra Information**

  This is a mix of quality of life improvements, such as a key for the map and little piece of text added to the page to guide the user. But it also entails a larger undertaking, that is making the pages educational on there own.This would include a homepage that would outline the difference between the two votes and why that matters in plain text. The final touch would be adding concise insights into the historical data being displayed. This

would include , explanations as to why a given party only ran in the regional election and not the constituency election, and why a party benefited from the additional member system and others suffered.

Bibliography

[1] R. Commission. (2004) Commission on the powers and electoral arrangements of the national assembly for wales. Accessed: 17/10/2020. [Online]. Available: https://webarchive.nationalarchives.gov.uk/20090807222148/http://www.richardcommission.gov.uk/content/template.asp?ID=/content/finalreport/index-e.asp

[2] D. Green and A. Gerber, "Get out the vote: How to increase voter turnout." Brookings Institution Press, 2019, pp. 182–183.

[3] J. Fivaz and G. Nadig, "To keep or to change first past the post?: the politics of electoral reform." Policy Internet, 2010, pp. 162–195.

[4] M. Molomo, "in search of an alternative electoral system for botswana," 2000.

[5] A. Blais, "To keep or to change first past the post?: the politics of electoral reform." OUP Oxford, 2008, pp. 1–1.

[6] V. A. 16. (2020) Votes at 16. Accessed: 27/10/2020. [Online]. Available: https://senedd.wales/en/abthome/role-of-assembly-how-it-works/Pages/Voting-Age.aspx

[7] J. Whittle. (2020) How the senedd is elected. Accessed: 27/10/2020. [Online]. Available: https://senedd.wales/en/gethome/elections-referenda/Pages/abt-nafw-how-assembly-elected.aspx

[8] C. Rallings and M. Thrasher. (2016) Elections in wales may 2016 –aspects of participationand administration. Accessed: 27/10/2020. [Online]. Available: https://www.electoralcommission.org.uk/sites/default/files/pdf_file/Report-Electoral-Data-National-Assembly-for-Wales-May-2016.pdf

[9] Senedd. (2020) Senedd. Accessed: 29/10/2020. [Online]. Available: https://www.youtube.com/channel/UCfFbE37IX0a-XAKE9yw-CPQ>

[10] S. Business. (2021) Election results. Accessed: 24/04/2021. [Online]. Available: https://business.senedd.wales/mgManageElectionResults.aspx

[11] BBC. (2021) Wales election 2016. Accessed: 24/04/2021. [Online]. Available: https://www.bbc.co.uk/news/election/2016/wales/results

[12] S. Business. (2021) Oops, something has gone wrong. Accessed: 24/04/2021. [Online]. Available: https://senedd.wales/error.aspx?aspxerrorpath=/people/184&RPID=1526367105

[13] Wikipedia. (2007) 2007 national assembly for wales election. Accessed: 24/04/2021. [Online]. Available: https://en.wikipedia.org/wiki/2007_National_Assembly_for_Wales_election

[14] M. Lewis-Beck and Stegmaier, "Us presidential election forecasting." PS: Political Science Politics, 2015, pp. 284–288.

[15] ——, "Forecasting elections in europe: Synthetic models." PS: Political Science Politics, 2015, p. 205316801456512.

[16] S. Kennedy, R. Wojcik and D. Lazer, "Improving election prediction internationally." Science, 355(6324), 2017, pp. 515–520.

[17] C. Cook and D. Wasserman, "Recalibrating ratings for a new normal." PS: Political Science Politics, 2014, pp. 304–308.

[18] T. P. S. Foundation. (2021) Laravel. Accessed: 24/04/2021. [Online]. Available: https://docs.python.org/3/library/urllib.html#module-urllib

[19] Laravel. (2021) Blade templates. Accessed: 24/04/2021. [Online]. Available: https://laravel.com/docs/8.x/blade

[20] leaflet. (2021) leaflet. Accessed: 28/04/2021. [Online]. Available: https://leafletjs.com/

[21] OpenStreetMap. (2020) Relation: Ynys môn / isle of anglesey (298793). Accessed 1 November 2020. [Online]. Available: https://www.openstreetmap.org/relation/298793#map=9/52.7355/-2.6532

[22] F. Zaninotto. (2021) Faker. Accessed: 23/04/2021. [Online]. Available: https://github.com/fzaninotto/Faker

[23] S. Research. (2021) Maps of senedd constituencies and regions. Accessed: 27/04/2021. [Online]. Available: https://research.senedd.wales/maps/

[24] O. for National Statistics. (2017) National assembly for wales constituencies (december 2017) wa bgc. Accessed: 01/04/2021. [Online]. Available: https://geoportal.statistics.gov.uk/datasets/national-assembly-for-wales-constituencies-december-2017-wa-bgc

[25] O. for Nathional Statics. (2017) National assembly for wales constituencies. Accessed: 24/04/2021. [Online]. Available: https://geoportal.statistics.gov.uk/datasets/7a4dc40d6c9e4abf8d2fa3b369395d93_3?geometry=-14.610%2C51.216%2C6.297%2C53.562

[26] Free and open-source software. (2021) Bootstrap. Accessed: 24/04/2021. [Online]. Available: https://getbootstrap.com/

[27] SqlDBM. (2021) Sqldbm. Accessed: 23/04/2021. [Online]. Available: https://sqldbm.com

[28] Laravel. (2021) Laravel homestead. Accessed: 27/10/2020. [Online]. Available: https://laravel.com/docs/8.x/homestead

[29] Oracle. (2021) Oracle vm virtualbox. Accessed: 27/10/2020. [Online]. Available: https://www.oracle.com/uk/virtualization/technologies/vm/downloads/virtualbox-downloads.html

[30] A. Skirius. (2019) Testing laravel websites over local network. Accessed: 07/11/2020. [Online]. Available: https://arunas.dev/testing-laravel-websites-over-local-network/

[31] Telemedia. (2021) Market snapshot: Desktop and mobile internet usage in 2020. Accessed: 24/04/2021. [Online]. Available: https://www.telemediaonline.co.uk/market-snapshot-desktop-and-mobile-internet-usage-in-2020

[32] NGork. (2021) Ngork. Accessed: 27/04/2021. [Online]. Available: https://ngrok.com/

[33] BBC. (2021) Welsh assembly election 2007. Accessed: 24/04/2021. [Online]. Available: http://news.bbc.co.uk/1/shared/vote2007/welshasssembly_english/html/region_99999.stm

Appendix A

d'Hondt

```
static public function get_region_results(Region $region, $constituencyWinners){
        $votes = [];
        $seatsGiven = [];
        $voters = $region->voters;
        $seats = array();
        $totalVotes =[];
        //Getting votes for each party and formating for use in the
        foreach($voters as $voter){
            //Seats that each party has won + 1
            $seatsGiven += [$voter->party->id => 1];
            //The votes that the D'Hondt formula modifies
            $votes += [$voter->party->id => 0];
            //A recored of there orignal votes is need
            $totalVotes += [$voter->party->id => $voter->votes];
        }

        //Caluating how many seats each Party already  one in the constituency
            race
        foreach($constituencyWinners as $con){
            if($con['regionID'] ==  $region->id)
                $seatsGiven[$con['seat']['partyID']] += 1;
        }

        $regionColours = array();
        $parties = $region->parties;
        $regionSeats = $region->seats;
        //the D'Hondt is run in stage give one seat away each irration
        foreach($regionSeats as $seat){
            $winningCount = 0;
            $winnerId = 0;
            //finding the party with the most votes
```

```
31              foreach($parties as $party){
32                  //weighting the partys total votes with the seats they have won
33                  $votes[$party->id] = $totalVotes[$party->id]/$seatsGiven[$party->
                        id];
34                  if($votes[$party->id] > $winningCount){
35                      //recording the highest vote count
36                      $winningCount = $votes[$party->id];
37                      $winnerId = $party->id;
38                  }
39              }
40              //Recording that the winner of this round has gained a seat
41              $seatsGiven[$winnerId ] += 1;
42              //this added the formated winner to the winner array
43              $representatives = $region->representatives;
44
45              foreach($representatives as $representative)
46                  if($representative->party->id ==$winnerId)
47                      $winner = $representative;
48
49              $colour = $winner->party->colour;
50              array_push($regionColours, $colour);
51              array_push($seats, ['id' => $seat->id, 'colour' => $colour,
52              'partyName' => $winner->party->name, 'partyImage' => $winner->party->
                    image,
53              'partyID' => $winner->party->id, 'repName' => $winner->name, 'repID'
                    => $winner->id]);
54          }
55          $colourOfRegion = YearController::get_region_colour($regionColours);
56          $seatCount = $region->seats->count();
57          $constituencyCount = $region->constituencies->count();
58          return ['name' => $region->name, 'id' => $region->id, 'electorate' =>
                $region->electorate, 'votes_cast'=> $region->votes_cast, 'seatCount'
                => $seatCount, 'constituencyCount' => $constituencyCount 'colour' =>
                $colourOfRegion, 'seats' => $seats];
59      }
60  }
```

Listing A.1: Region Controller Function for modeling regional winners

Appendix B

Test Results

PARTY PAGE TEST

2016			
	Historical	Model	
Party	Region Seat(s)	Region Seat(s)	Test
UKIP	7	7	PASS
Labour	2	2	PASS

2011			
	Historical	Model	
Party	Region Seat(s)	Region Seat(s)	Test
Green	0	0	PASS
Cymru	6	6	PASS

2007			
	Historical	Model	
Party	Region Seat(s)	Region Seat(s)	Test
Democrats	3	3	PASS
Conservative	7	7	PASS

A. All Party Page Testing

REGION PAGE TEST

2016			
	Historical	Model	
Region	Region Seat(s)	Region Seat(s)	Test
Mid & West	L,L,U,P	L,L,U,P	PASS
South Central	C,C,P,U	C,C,P,U	PASS

2011			
	Historical	Model	
Region	Region Seat(s)	Region Seat(s)	Test
North Wales	C,C,D,P	C,C,D,P	PASS
South Wales East	P,P,C,C	P,P,C,C	PASS

2007			
	Historical	Model	
Region	Region Seat(s)	Region Seat(s)	Test
South Wales West	P,P,C,D	P,P,C,D	PASS
Mid & West	L,L,C,P	L,L,C,P	PASS

B. All Region Page Testing

Figure B.1

2007												
	Historical Results				Model Data				Test			
Totals:	1753639	20	40	60	1763386	20	40	60	FAil	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	603,879	2	24	26	609325	5	24	29	FAil	FAIL	PASS	FAIL
Plaid Cymru	423,878	8	7	15	414888	5	7	12	FAil	FAIL	PASS	FAIL
Conservative	427,883	7	5	12	427884	6	5	11	FAil	FAIL	PASS	FAIL
Liberal Dem	258,950	3	3	6	253505	4	3	7	FAil	FAIL	PASS	FAIL
Independent	39,049	0	1	1	57784	0	1	1	FAil	PASS	PASS	PASS

2011												
	Historical Results				Model Data				Test			
Totals:	1732088	20	40	60	1732088	20	40	60	PASS	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	751612	2	28	30	751612	6	28	34	PASS	FAIL	PASS	FAIL
Plaid Cymru	352706	6	5	11	352706	2	5	7	PASS	FAIL	PASS	FAIL
Conservative	451162	8	6	14	451162	9	6	15	PASS	FAIL	PASS	FAIL
Liberal Dem	176608	4	1	5	176608	3	1	4	PASS	FAIL	PASS	FAIL

2016												
	Historical Results				Model Data				Test			
Totals:	1903274	20	40	60	1903271	20	40	60	FAil	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	673062	2	27	29	673061	6	27	33	FAil	FAIL	PASS	FAIL
Plaid Cymru	420924	6	6	12	420922	4	6	10	FAil	FAIL	PASS	FAIL
Conservative	406443	5	6	11	406443	4	6	10	PASS	FAIL	PASS	FAIL
Liberal Dem	143669	0	1	1	143669	1	1	2	PASS	FAIL	PASS	FAIL
UKIP	259,176	7	0	7	259176	5	0	5	PASS	FAIL	PASS	FAIL

A. Year Test d'Hondt

2007												
	Historical Results				Model Data				Test			
Totals:	1753639	20	40	60	1763386	20	40	60	FAil	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	603,879	2	24	26	609325	2	24	26	FAil	PASS	PASS	PASS
Plaid Cymru	423,878	8	7	15	414888	8	7	15	FAil	PASS	PASS	PASS
Conservative	427,883	7	5	12	427884	7	5	12	FAil	PASS	PASS	PASS
Liberal Dem	258,950	3	3	6	253505	3	3	6	FAil	PASS	PASS	PASS
Independent	39,049	0	1	1	57784	0	1	1	FAil	PASS	PASS	PASS

2011												
	Historical Results				Model Data				Test			
Totals:	1732088	20	40	60	1732088	20	40	60	PASS	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	751612	2	28	30	751612	2	28	30	PASS	PASS	PASS	PASS
Plaid Cymru	352706	6	5	11	352706	6	5	11	PASS	PASS	PASS	PASS
Conservative	451162	8	6	14	451162	8	6	14	PASS	PASS	PASS	PASS
Liberal Dem	176608	4	1	5	176608	4	1	5	PASS	PASS	PASS	PASS

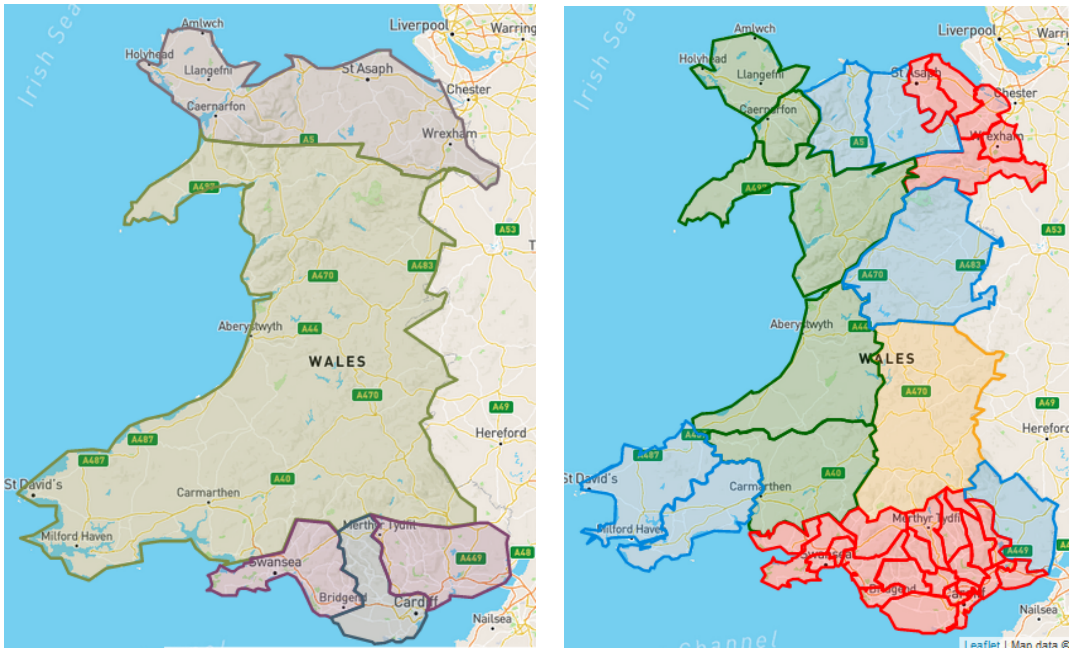
2016												
	Historical Results				Model Data				Test			
Totals:	1903274	20	40	60	1903271	20	40	60	FAil	PASS	PASS	PASS
Party	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total	Vote	Region	Consituency	Total
Labour	673062	2	27	29	673061	2	27	29	FAil	PASS	PASS	PASS
Plaid Cymru	420924	6	6	12	420922	6	6	12	FAil	PASS	PASS	PASS
Conservative	406443	5	6	11	406443	5	6	11	PASS	PASS	PASS	PASS
Liberal Dem	143669	0	1	1	143669	0	1	1	PASS	PASS	PASS	PASS
UKIP	259,176	7	0	7	259176	7	0	7	PASS	PASS	PASS	PASS

B. Year Test modified d'Hondt

Figure B.2: Figure B.2A. show the 2016 results using just the d'Hondt methods. While Figure B.2B. show testing the modified d'Hondt methods

Appendix C

Produced Web Pages



A. Temporary Region Data B. Temporary Constituency Data

Figure C.1:

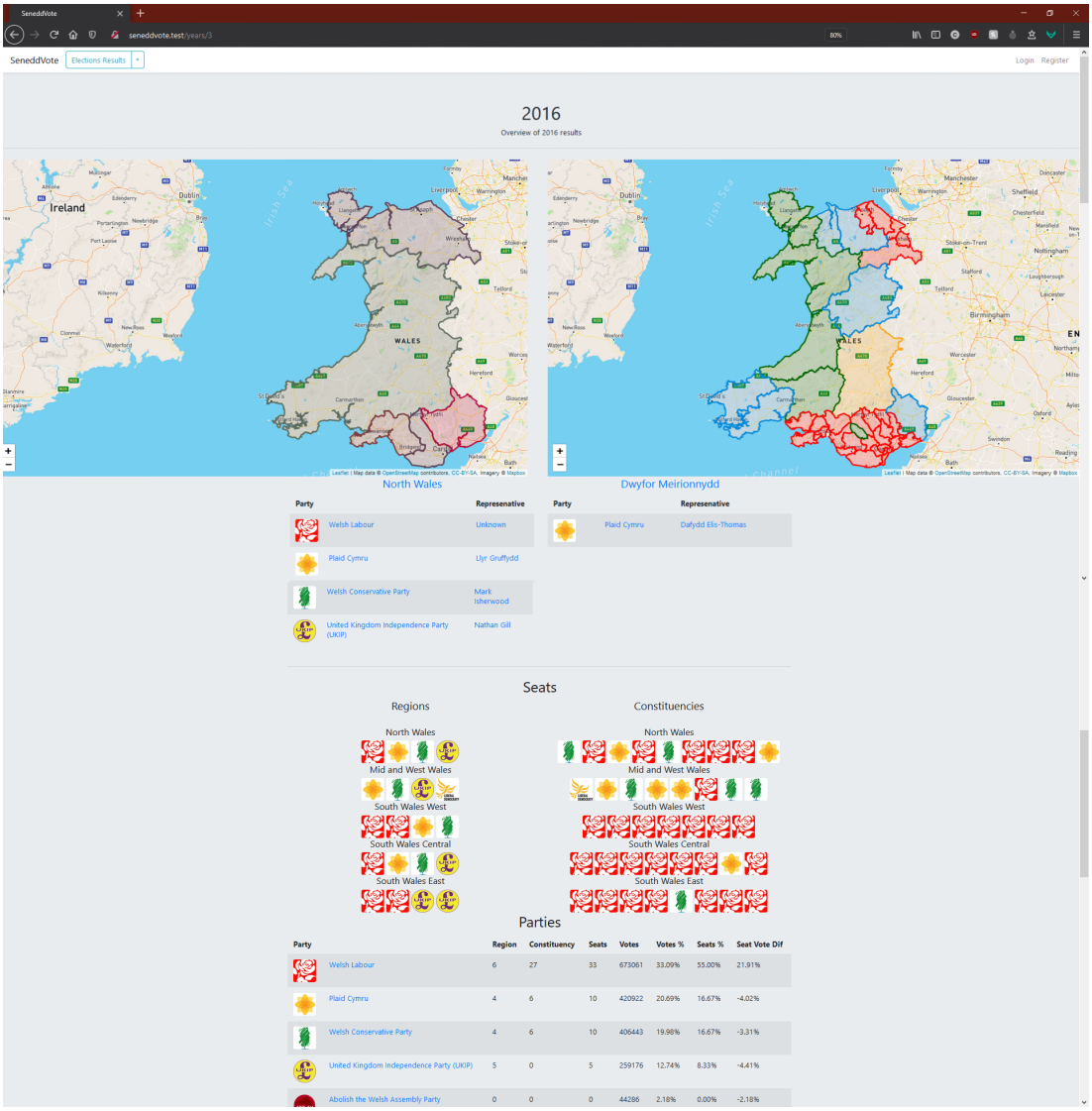


Figure C.2: Year View

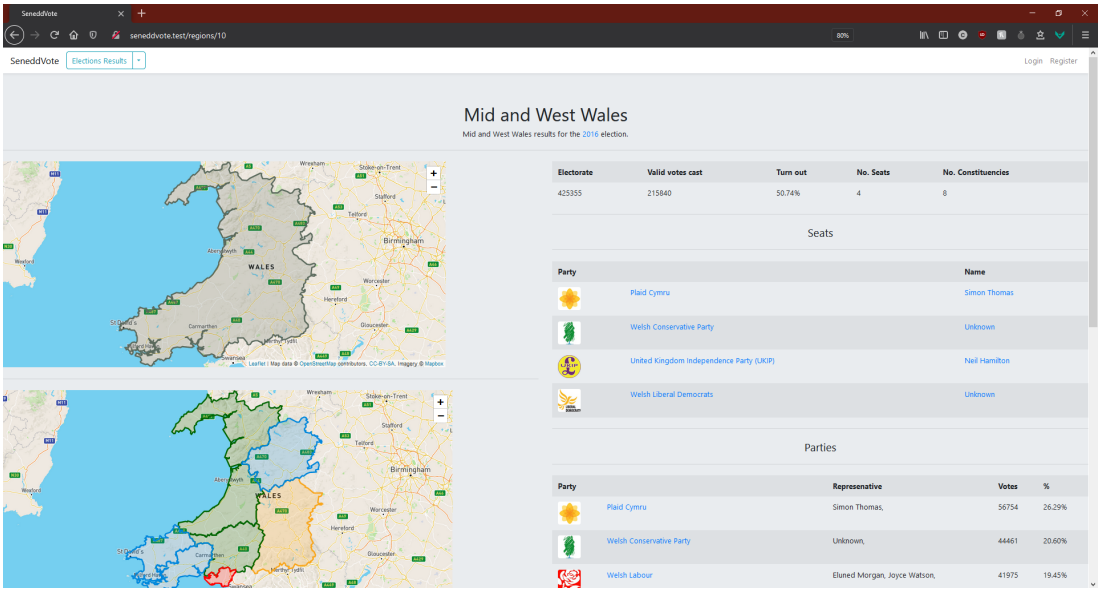


Figure C.3: Region View

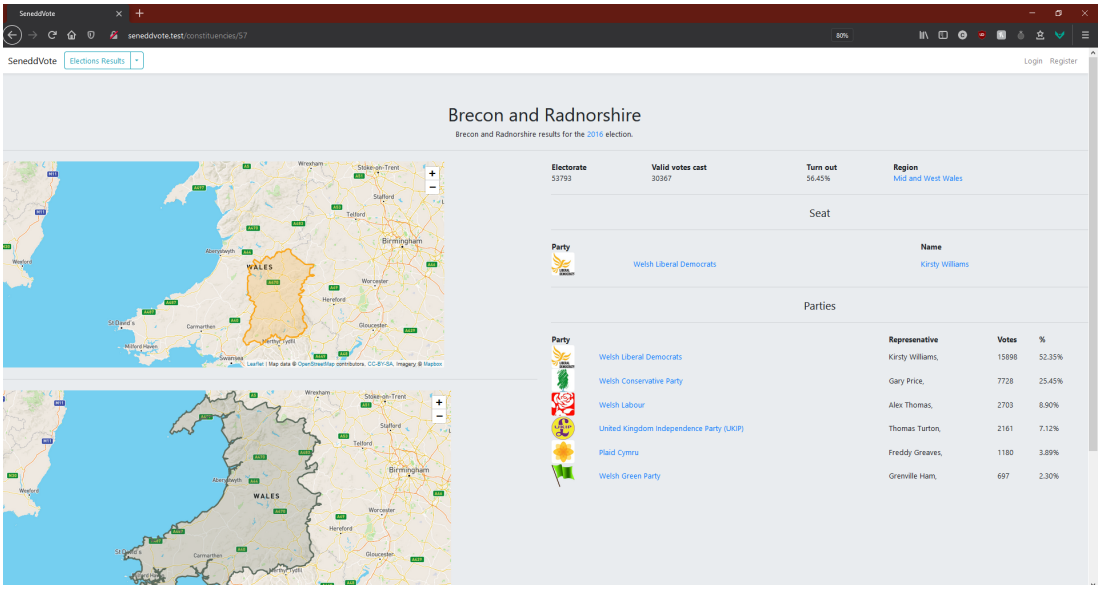


Figure C.4: Constituency View

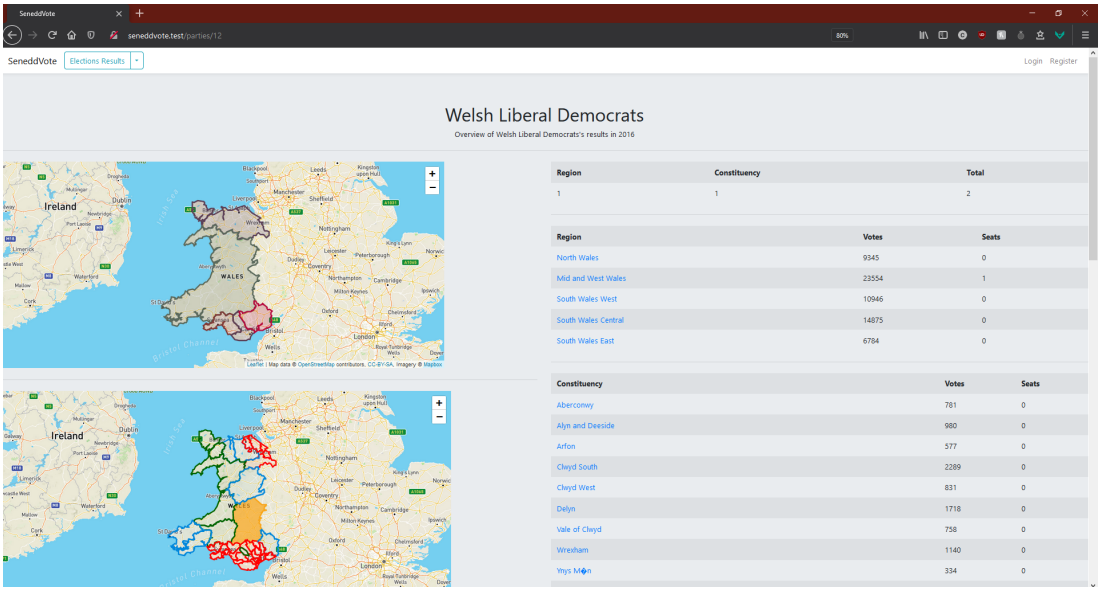
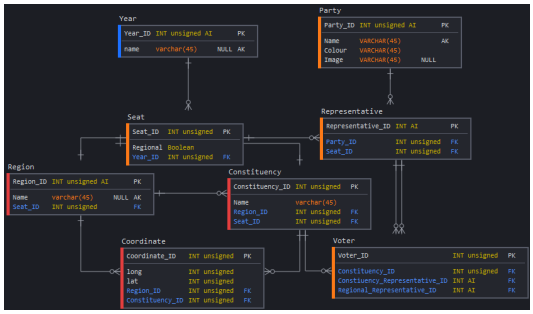


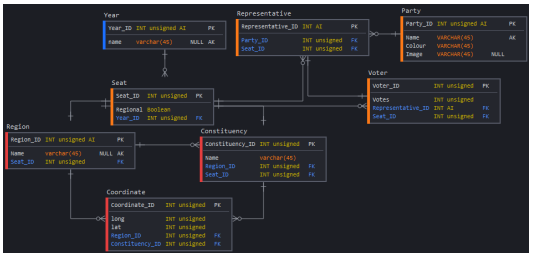
Figure C.5: Party View

Appendix D

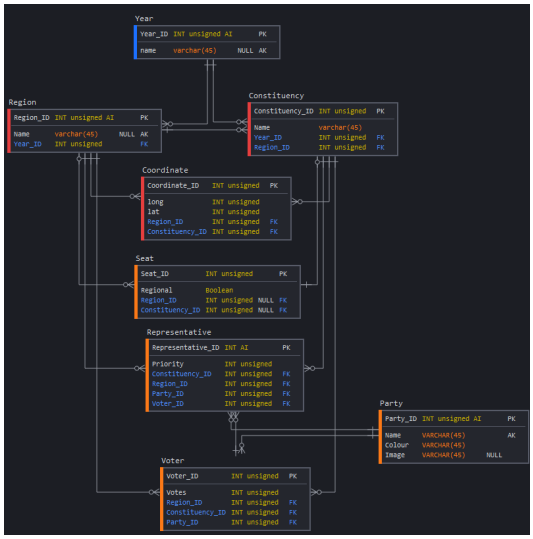
Database Evolution



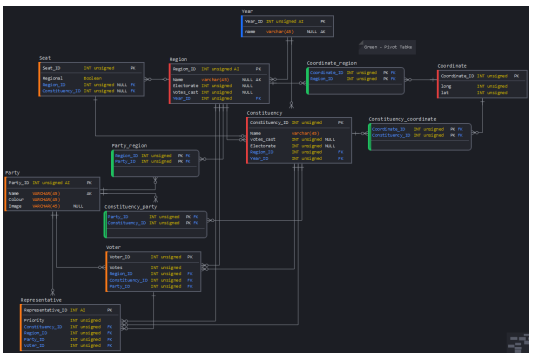
A. Database 0: Original Plan



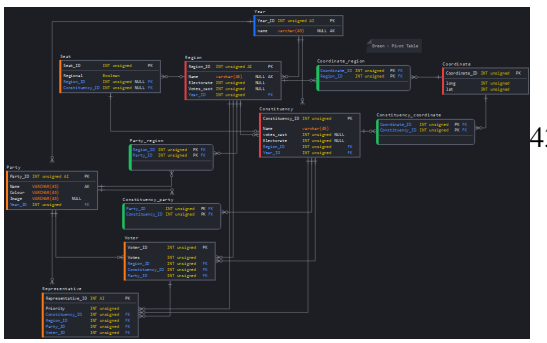
B. Database 1: Single Vote



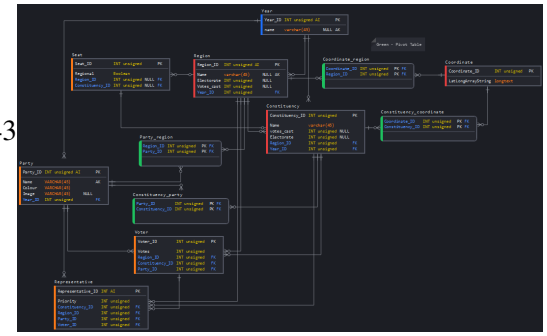
C. Database 2: Region and Constituency Focused



D. Database 3: Removed Redundant Coordinates



E. Database 4: Party Changes

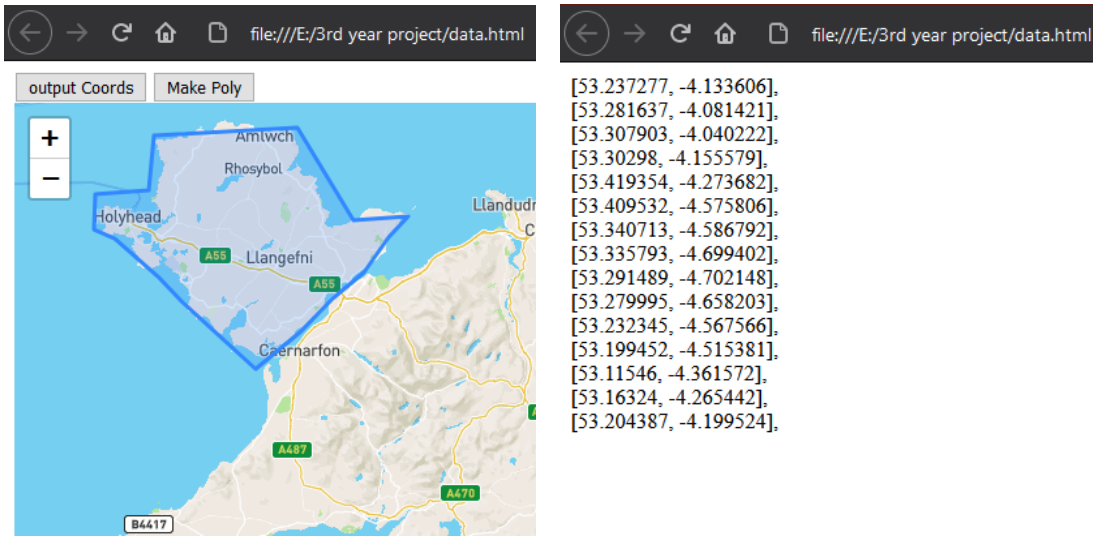


F. Database 5: Final

Appendix E

Data Formatting Tools

E.1 JavaScript Formatting Tools



A. Drawing Polygon B. Outputting Polygon

Figure E.1: Shows a JavaScript Polygon maker, that lets you draw a polygon and then output it's latitude and longitude