My project dealt heavily with data manipulation using SQL and PHP. I also worked with a lot of HTML forms and implemented one Javascript function. As a starting place, it used the framework provided by C$50 Finance (ie. `login.php`, CSS Stylesheets, jquery, etc.), as its user interface was very similar to the type of interface required by this project. In addition, this project required some of the same functionality as C$50 Finance, for example printing tables based on user input and allowing the user to update data in the SQL database. Looking at the implementation of C$50 Finance provided a starting place for me to begin my work.

One of the most challenging tasks for me was to implement the `search.php`. It was a particular challenge because users had the ability to enter a number of different data types (`ints`, `varchars`) and some of the data was stored in `Fulltext` columns while other data was not. Additionally, I wanted users to be able to enter any combination of search terms and not be restricted to using only one type of search at a time. Ultimately, I settled on using the `LIKE` operator and concatenating all of the data entered into a single string and enclosing each term using '`%`' to search, which ultimately proved successful. In addition, I had to ensure that the search would run correctly even if the user did not input gender or party, both of which used radio buttons. To solve this issue, I added an additional "default" radio button to each that had a value of "". This default button was set to be hidden using HTML, as to not clutter the interface.

Next I implemented `fundraising.php`. I made the decision to make the Voter Registration Number the primary index and require that the candidate enter it for each person who's information they are editing. This is not the most "user friendly" decision, but I thought it would be useful because the Voter Registration Number is the "primary index" used by the City/Town Clerk, and would avoid all ambiguity in cases where voters have identical or similar names and addresses. (I am a case myself, as both my father and I are registered under the same name at the same address.) I wanted to return a confirmation page to the user confirming the data they had just edited, and I initially struggled with finding a way to return the row that was edited. Ultimately, I decided on adding an additional column to the database called `isedited` of type `bool`. By default, the column is `false`, but when the user edits a row, it is changed to `true`. Using PHP, I construct a table of all rows that have a `true isedited` column.

Immediately after preparing the table, the column is reset to `false` so that that row is not returned the next time the user adds additional fundraising information.

`Generate Reports` is similar to the other pages that return a table, but it contains an `if` statement in the PHP. Because the user can request multiple types of reports from `reports.php`, the controller prepares different information based on the value of `$_POST`. `Election Commission Reporting` has a similar structure, with the addition of the arithmetic done at the top of the page. Of particular interest are `$totalreceipts` and `$underfifty`, which record the sum of all donations received and the sum of all donations that were under $50 each, respectively. For `$totalreceipts`, I was able to use the `SUM(column)` operator to add all the figures in the column together. With `$underfifty`, however, I was unable to do this because `SUM( )` does not support the addition of a subset of the total database (ie. `SELECT x FROM y WHERE z < 50`). To address this issue, I created a `foreach` loop and implemented `$counter` which added each row of `$underfifty`.

Finally, I implemented `import.php`, which was similar in many ways to import from C$50 Finance. First, I implemented a file upload tool using HTML that would allow the user to select the desired file from their computer. I then passed this to `fgetcsv` using `$_POST`. I had to particular difficulties with implementing this feature. First, was an issue with the CSV file. I discovered that PHP can have difficulty reading certain CSV files in that it does not correctly interpret the end of the line. Ultimately, I discovered a line of code on the internet that sets `auto_detect_line_endings` to `true`, which solves the problem. Additionally, for an unknown reason, the page `import.php` did not recognize the superglobal variable `$_SESSION`, which I was using to tie the import of voter entries to the user's User ID. Ultimately, I added a line that starts `$_SESSION` if it is not active when the page loads.

The final feature that I implemented was a javascript script that prints information off the webpage. I decided that the browser's print feature was insufficient because it would be unacceptable for the webpage header/footer to be printed alongside the desired content. Therefore I created a `print_div`, which I used to enclose all the data I wanted to be printed. Finally, I implemented a script that prints the contents of `print_div` when a button is clicked on the screen.