

YOUR VOTE COUNTS.

YOUR VOICE MATTERS.

A Voter Registration Database Application



Team Members

Abdel Rahman Mansour, Adam Elkhanoft, Gabriel Medina, George Adler Buras,
John Hudnall, Katherine Perez, Kendall Comeaux

Table of Contents

1. Problem Statement	3
2. Methods	4
3. Experiments	6
4. Results	9
5. Discussion	15

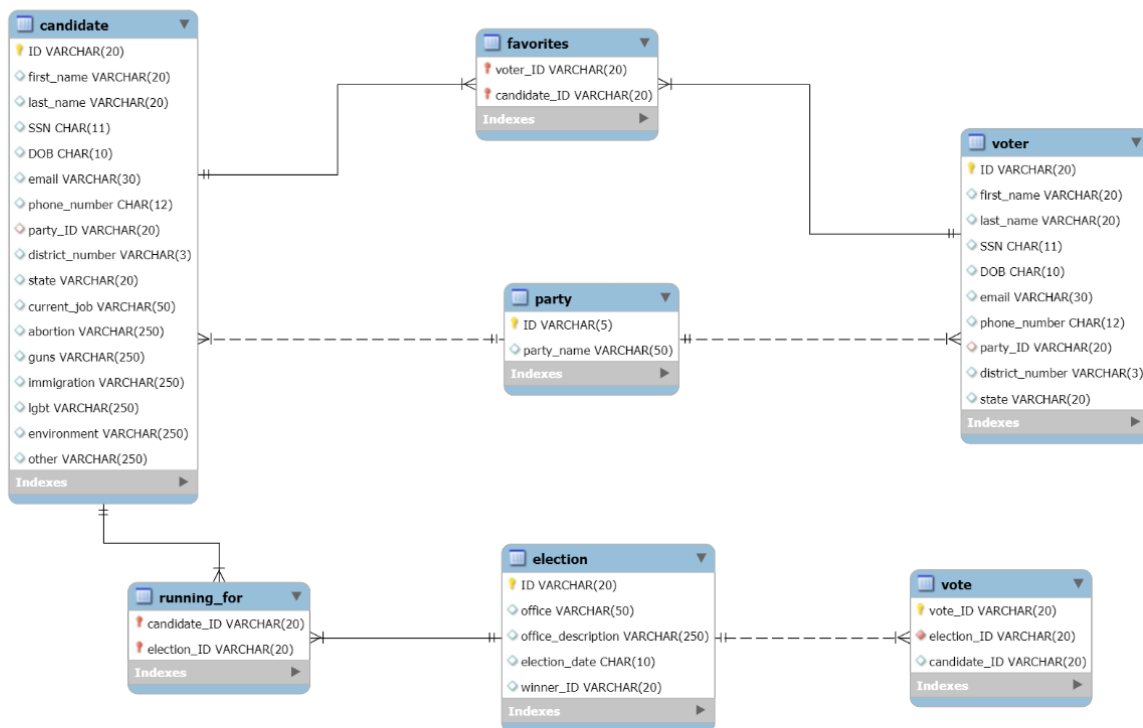
Problem Statement

The goal of this project was to create an improved voter registration database management system (DBMS) that will simplify and streamline the voting process for everyone. While modern-day voter registration DBMSs are efficient and get the job done, they lack many features which would increase the usability of these systems such as providing meaningful information about election candidates and the policies that they support. Our newly proposed voter registration DBMS will allow citizens to easily sign themselves up to vote, view all potential candidates as well as their stances on controversial topics, favorite candidates they would like to view again later, and keep track of ongoing elections. Our group's motivation for choosing this project is to expand and build upon the capabilities of modern-day voter registration DBMSs because of their significance and the large-scale impact that voting as a whole has on our daily lives.

We built a new and improved voter registration DBMS that contains all the basic functionality expected from such a system while simultaneously adding two new main features that will improve the user experience for all voters. The biggest feature that most voter registration DBMSs lack that ours provides is the ability to view candidate's opinions on numerous topics that are relevant to our political climate. This is an essential feature because it allows voters to develop their own individual opinions on the candidates which in turn equips them with the ability to make an educated vote based on what they prioritize in a potential candidate. Another feature that isn't readily available in most voter registration DBMSs that our application provides is the ability to favorite particular candidates. This feature not only grants voters the capability to keep track of how their favorite candidates are doing in a specific election, but also the ability to view all of their favorite candidates on a separate page for easy access. Altogether, the addition of these features will significantly enhance the voting experience and make the process of keeping track of elections a lot less daunting and intimidating to the average citizen that is just trying to vote responsibly without spending weeks researching and keeping track of all the candidates for all the elections.

Methods

Our voter registration DBMS consists of seven major tables that will help describe and show all the necessary relations required for an effective voter registration DBMS. The seven tables being: **a party table** (ID, party_name), **a voter table** (ID, first_name, last_name, SSN, DOB, email, phone_number, party_ID, district_number), **a candidate table** (ID, first_name, last_name, SSN, DOB, email, phone_number, party_ID, district_number, state, current_job, abortion, guns, immigration, lgbt, environment, other), **an election table** (ID, office, office_description, election_date, winner_ID), **a running_for table** (candidate_ID, election_ID), **a vote table** (voter_ID, election_ID, candidate_ID), and **a favorites table** (voter_ID, candidate_ID).



As you can see in the table above, **the party table** is used to define political parties, which voters and candidates can belong to. This allows us to query the database for party membership. **The voter table** is used to define a person who can vote in an election. Voters have attributes that could be used to verify their identity, which would be useful to election

administrators working the registration desk in a real election. This application was created to be used from the perspective of a voter. Like the voter table, **the candidate table** contains identity verifying information to ensure that the people running for office are real. In addition, the candidate table contains attributes where the candidates can provide a brief overview of their positions on controversial topics (abortion, guns, immigration, lgbt, environment, and other). These attributes allow our application to provide a centralized location where voters can learn about the opinions of candidates, which is a feature lacking in existing similar applications. This is an improvement because instead of needing to search for each candidate's website (which may or may not exist), voters will be presented with the information on those sites which they need to become informed about the candidates they are voting for. **The election table** is used to define an election, including relevant information such as what office the election is for, a description of that office, the date the polls will close, and who won the election.

The running_for table is used to record which candidates are running in which elections. We query it to find all the candidates that are running for a particular election. **The vote table** is used to record each vote that a candidate receives in an election. To preserve the privacy of voters, the person who cast the vote is not recorded. If we wanted to keep track of who has voted (to ensure that no one votes twice) then we would implement a voted table with voter_ID and election_ID attributes, but we deemed doing so was unnecessary for the scope of our project. The vote table is queried to determine how many votes a candidate has received in a particular election. **The favorites table** is used to record which candidates a voter likes the most. In our application, voters can press a heart button to favorite a candidate. The favorites table is queried to create a list of the candidates that the voter who is logged in has favorited. This allows our application to present the voter with a list of their favorite candidates, which is a feature that existing similar applications lack.

Experiments

Before getting into the actual details of how we implemented the project, we just want to quickly give some context on the technologies that we used to develop our application. Our application's back-end had two main components, a database and a server. The database that we used was MySQL because it was a requirement for this class and it also allowed us to get more familiar with how SQL can be used to build enterprise-level software. In regards to our server, we utilized Node.js in cohesion with the Express.js framework to build REST API endpoints which allowed the front-end to easily communicate with the database and keep an up-to-date visual for the user. Furthermore, for our front-end we used React because it is the most popular JavaScript framework and it has an enormous amount of libraries and user support which makes the process of developing a user interface fairly straightforward. Finally, we also made use of a couple of miscellaneous technologies, such as Discord for communication, GitHub for version control, and Visual Studio Code as our text editor. Regarding our data collection and how we retrieved our data samples, our tables were filled with made-up data for the purpose of demonstrating the functionality of the application. All of the data was either created with the intent to parody reality or created arbitrarily. Further data can be added to the database through the application (e.g. by registering as a new voter, favoriting a candidate, voting for a candidate, etc.).

Moreover, let us begin talking about the implementation of the actual project and how we hooked up the front-end to the back-end. We will start from the very beginning of the project, the first two pages that a user can view when they initially open up the application are the login and registration pages. These pages behave exactly as you would expect them to, they are simply forms that prompt the user to enter information which will either create a new account or log into an already existing account. However, the underlying mechanism for these pages is that once a user submits a form, the front-end application sends an HTTP request to one of the server's REST API endpoints where the endpoint will do some input validation and then either create a new voter account in the voter table or simply retrieve the data affiliated with the already existing voter account in the voter table.

Once a user creates an account and logs into their account, they will be greeted with the profile page which will display all of the data that is associated with their account. This page is fairly simple and doesn't have much interactivity, but the main thing that a user can do on this page is update their information. The underlying mechanism for this page, and pretty much all the pages of this application, is very similar to the login and registration pages in the sense that once the user submits the updated information, the front-end application sends an HTTP request to one of the server's REST API endpoints where the endpoint will make sure that the data submitted is valid and then update the data in the voter table accordingly.

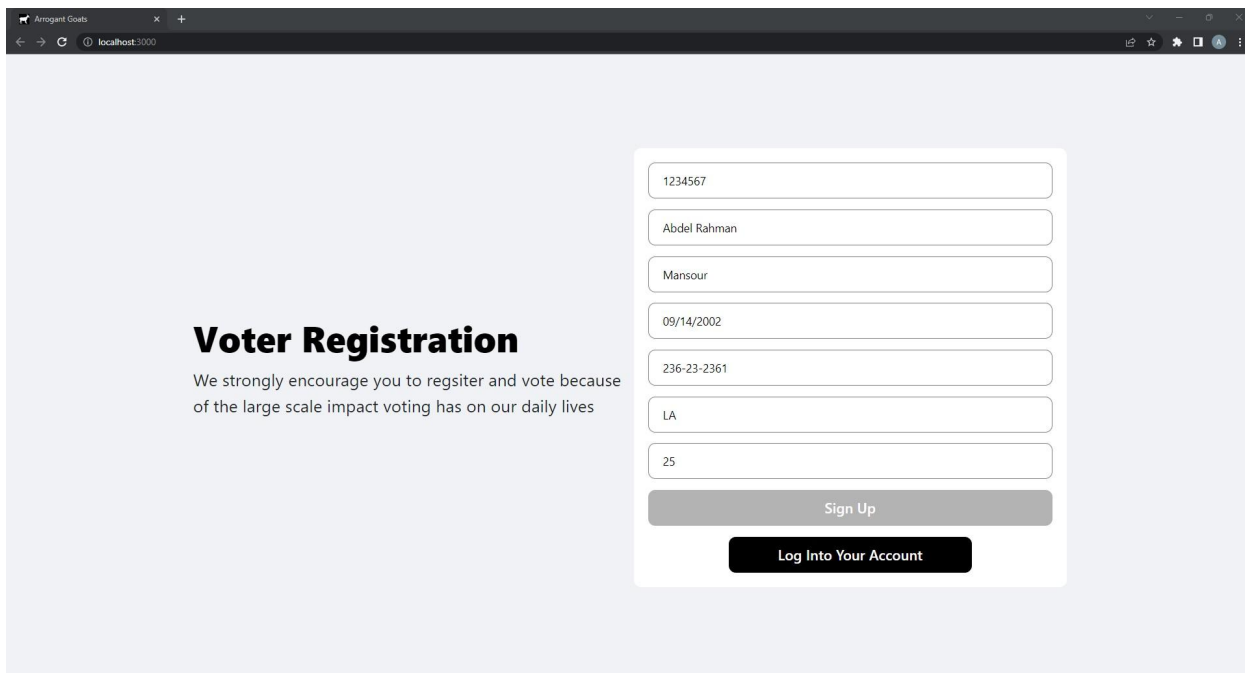
The next two pages that the user can navigate to are the candidates and favorites pages, these pages are heavily intertwined because the actions that the user takes on the candidates page directly impacts what they will view on the candidates page. On the candidates page, the user will be able to view all of the potential candidates for all of the elections in an easy to view and digestible card format. If the user clicks on one of these cards, they will then get a full page focused view of that specific candidate where they can look over that candidate's details and opinions. Nevertheless, the main feature on this page is demonstrated by the favorite icon which is present on all of the candidate cards. When this favorite icon is clicked an HTTP request containing the candidate's ID and the user's voter ID is sent to one of the server's REST API endpoints where the endpoint will make sure that this combination of IDs doesn't already exist and if it doesn't then it will query the MySQL database to add a new entry to the favorites table. Therefore, this is where the connection between the candidates page and favorites page begins to show, because when the favorites page is loaded, a request is sent to retrieve all the candidates tied to the user's voter ID from the favorites table. Then the front-end will render the candidate cards for those specific candidates that were previously added to the favorites list on the candidates page. Finally, if the user decides that they no longer like a candidate they have the option to remove a candidate from their favorites list, to do this they simply have to click on the favorite icon again which submits an HTTP request to the server which will in turn query the MySQL database to delete that entry from the favorites table.

Finally, the last and most important page that the user has access to is the elections page where they are able to view all of the elections that are happening right now with a short

description of the responsibilities these offices entail. If the user clicks on one of these elections, they will then be presented with a page that shows all of the potential candidates that are running for that specific office. Then similar to the candidates page, if the user clicks on a candidate card on this page they will get a full page focused view of that specific candidate where they can look over that candidate's details and opinions. However the biggest difference between this page and the candidates page, is that when the user clicks on a candidate card they are also shown a vote button which, as you would expect, allows them to vote for that candidate. If the user clicks on the vote button, the front-end application sends an HTTP request containing the election's ID, the candidate's ID, and the user's voter ID to one of the server's REST API endpoints and then this endpoint will make sure that the IDs provided are valid in the MySQL database and if they are valid it will submit a query to the database which will create a new entry in the vote table. This new entry in the vote table will also be reflected on the front-end by incrementing the vote counter for that candidate in that specific election.

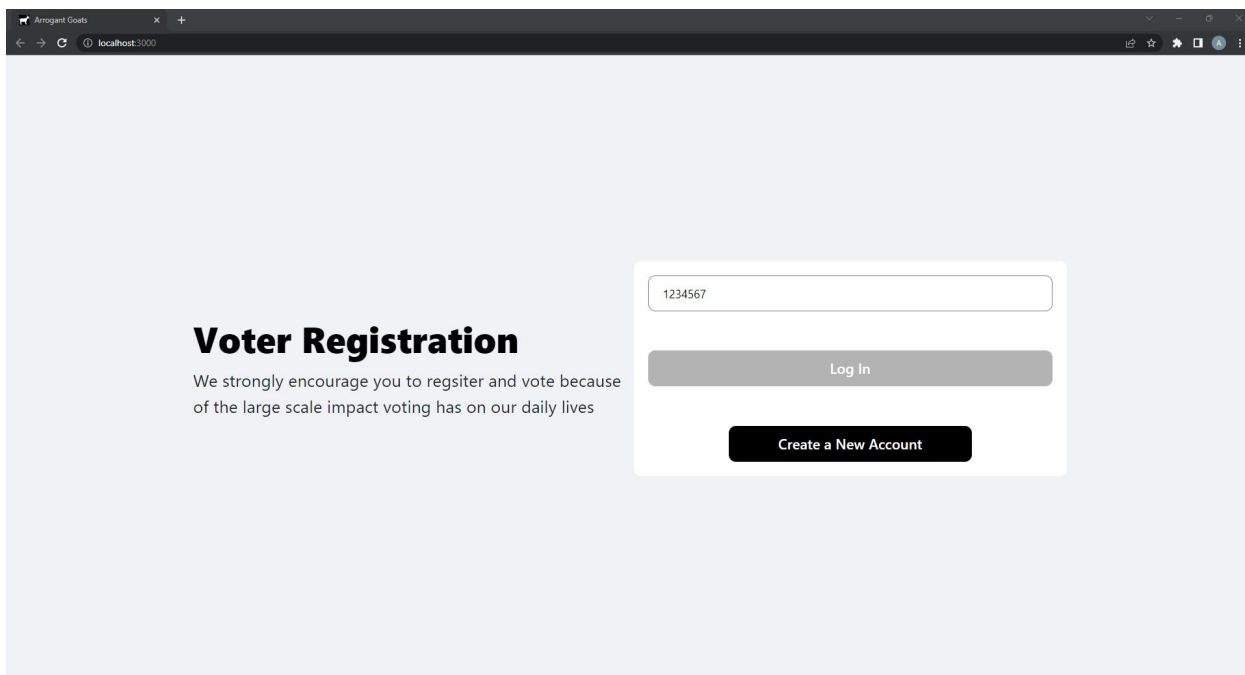
Results

The application opens up on the registration page, where a user can create a new voter account.



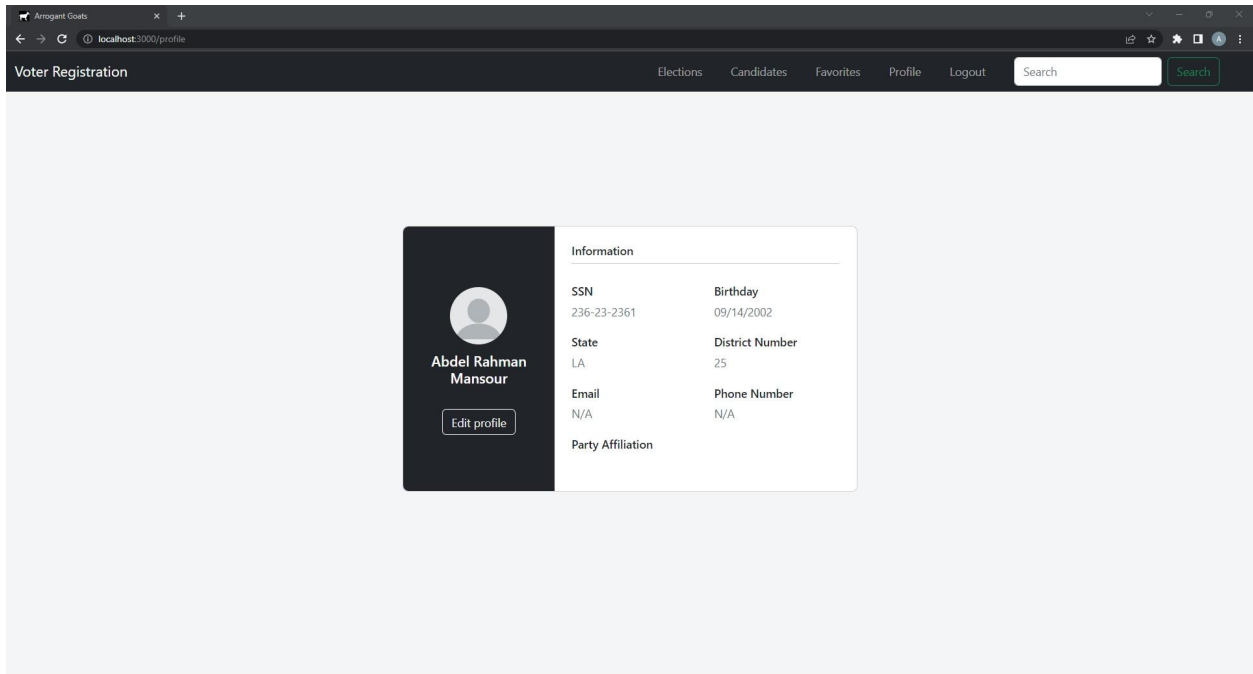
The screenshot shows a web browser window with the title "Arrogant Goats" and the address bar displaying "localhost:3000". The page has a light blue background. On the left, the text "Voter Registration" is displayed in bold, followed by the message "We strongly encourage you to regsiter and vote because of the large scale impact voting has on our daily lives". On the right, there is a white registration form with the following fields: "1234567", "Abdel Rahman", "Mansour", "09/14/2002", "236-23-2361", "LA", and "25". Below the form are two buttons: a grey "Sign Up" button and a black "Log Into Your Account" button.

Once they create a voter account, it redirects them to the login page where they can input their voter ID.

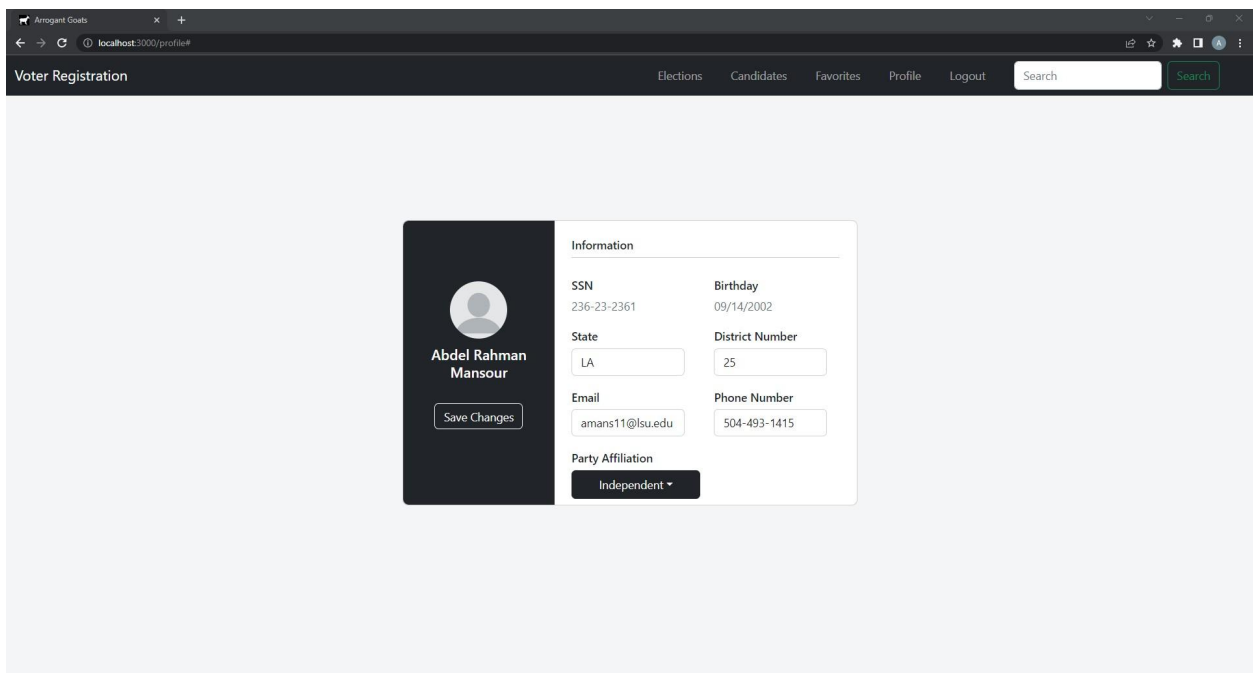


The screenshot shows the same web browser window as before. The page content is identical, but the registration form on the right now only contains a single input field with the value "1234567". Below this field are two buttons: a grey "Log In" button and a black "Create a New Account" button.

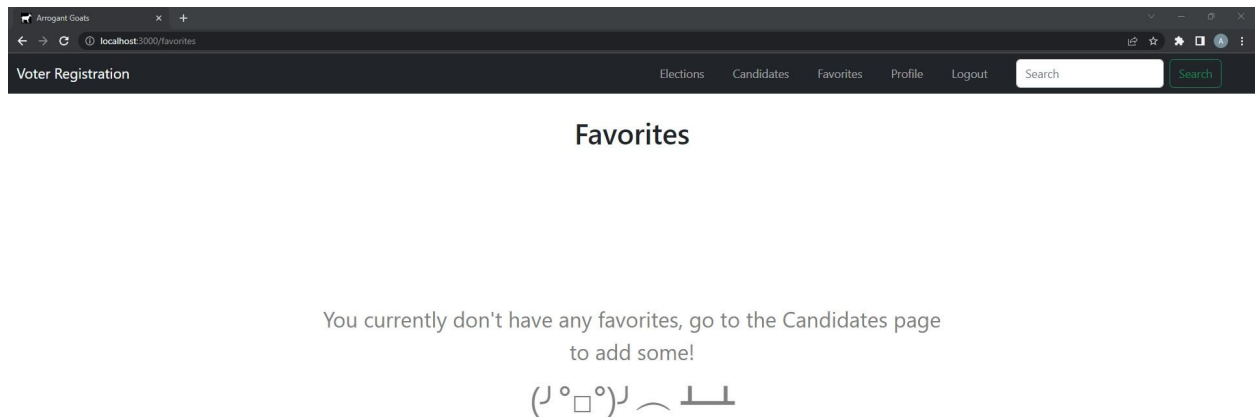
If the user inputs a valid voter ID, the application will then retrieve all their data from the database and display it in the profile page.



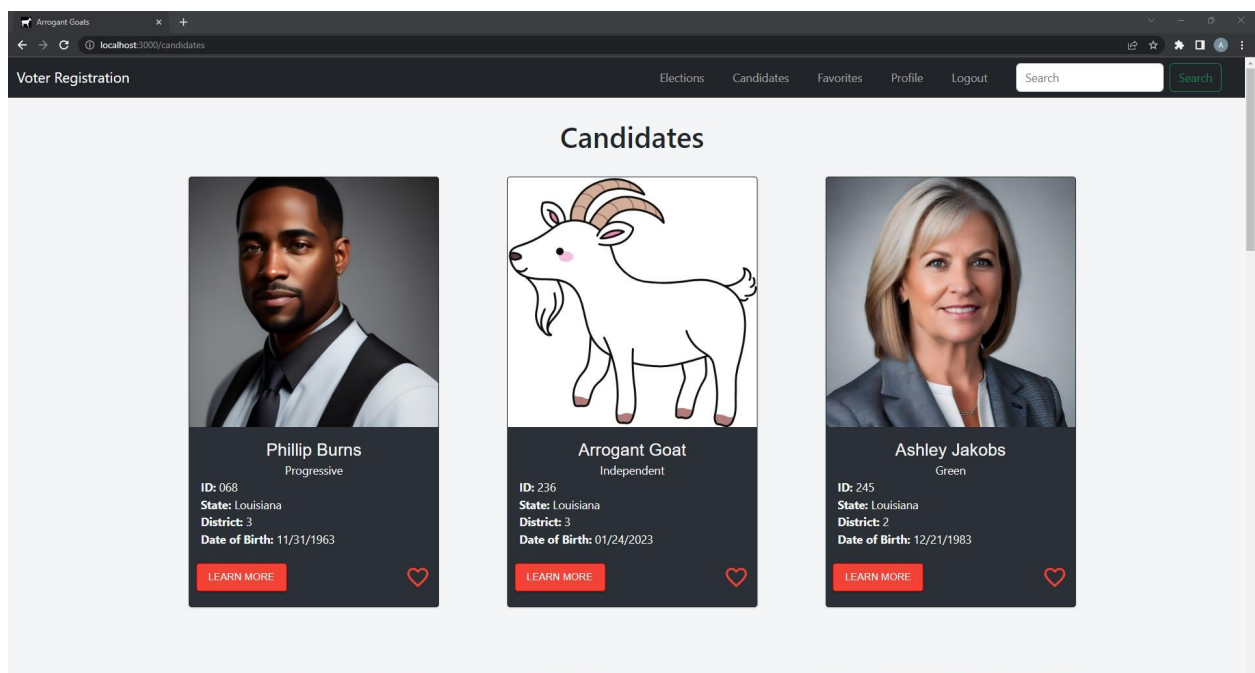
The user can then edit their profile page and all of these updates will be reflected in the database accordingly.



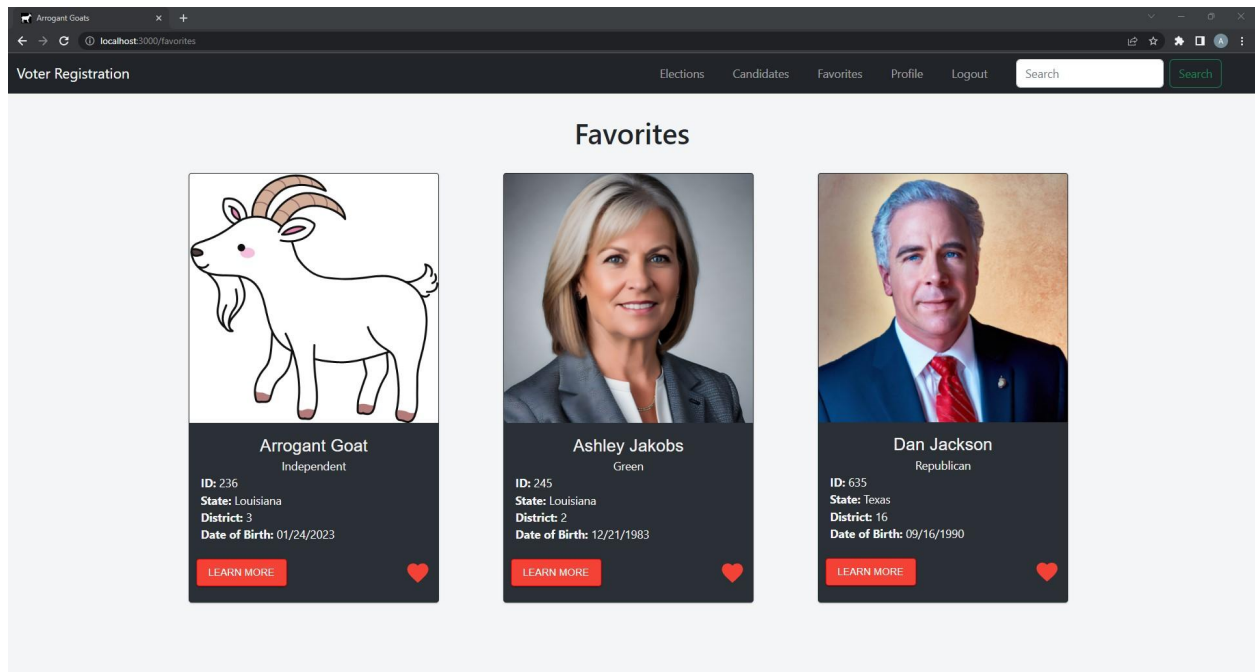
The favorites page is initially empty when a user creates an account because they don't have any favorites associated with them in the database.



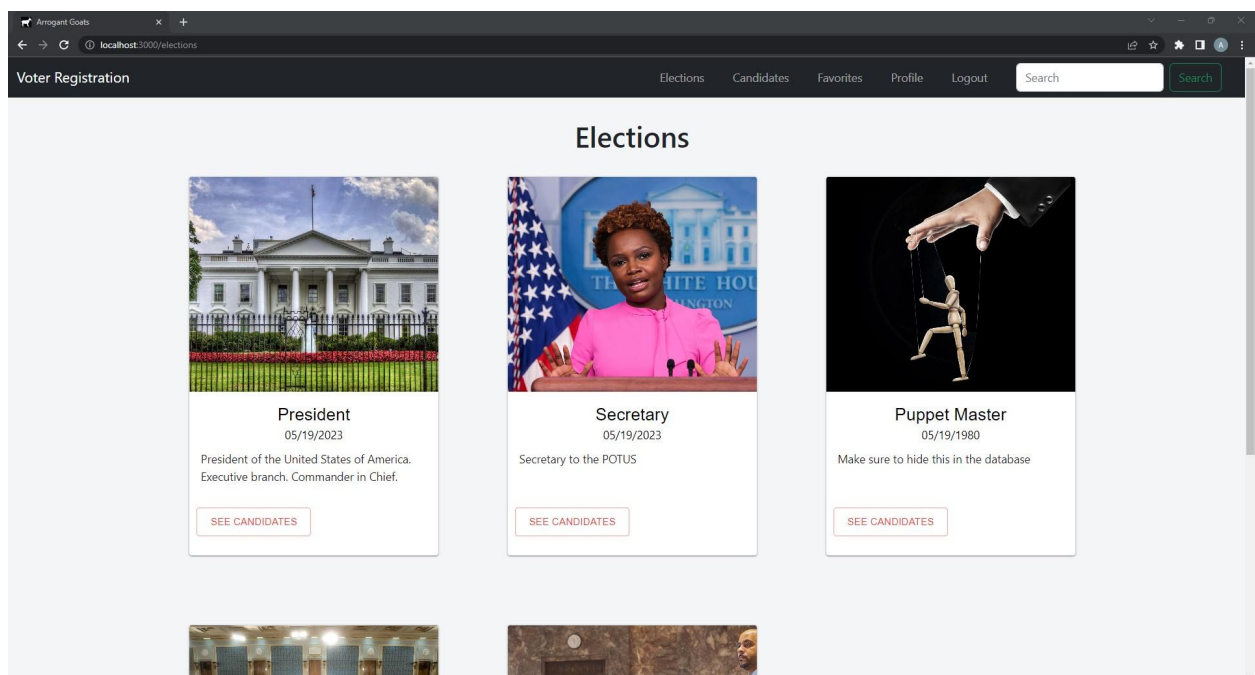
The user can navigate to the candidates page, where they can view all of the potential candidates for all the elections. They can then add to their favorites list by pressing on the heart button.



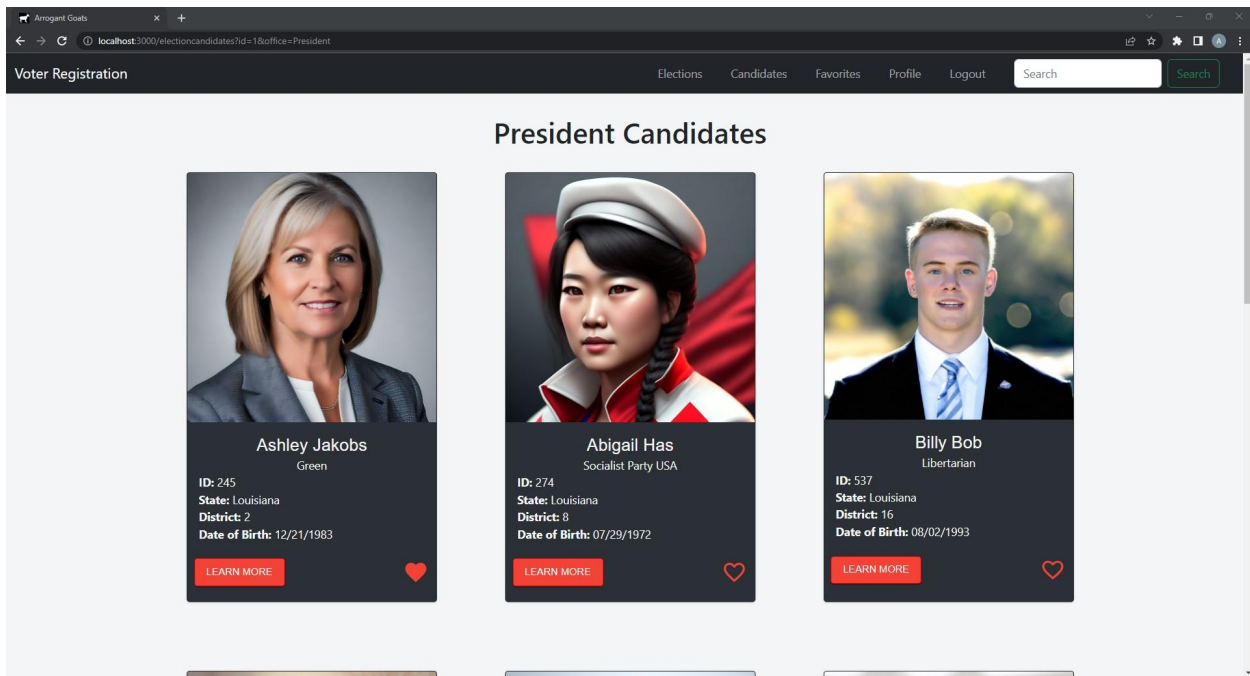
After the user adds a couple of favorites from the candidates page, this is what the favorites page will look like.



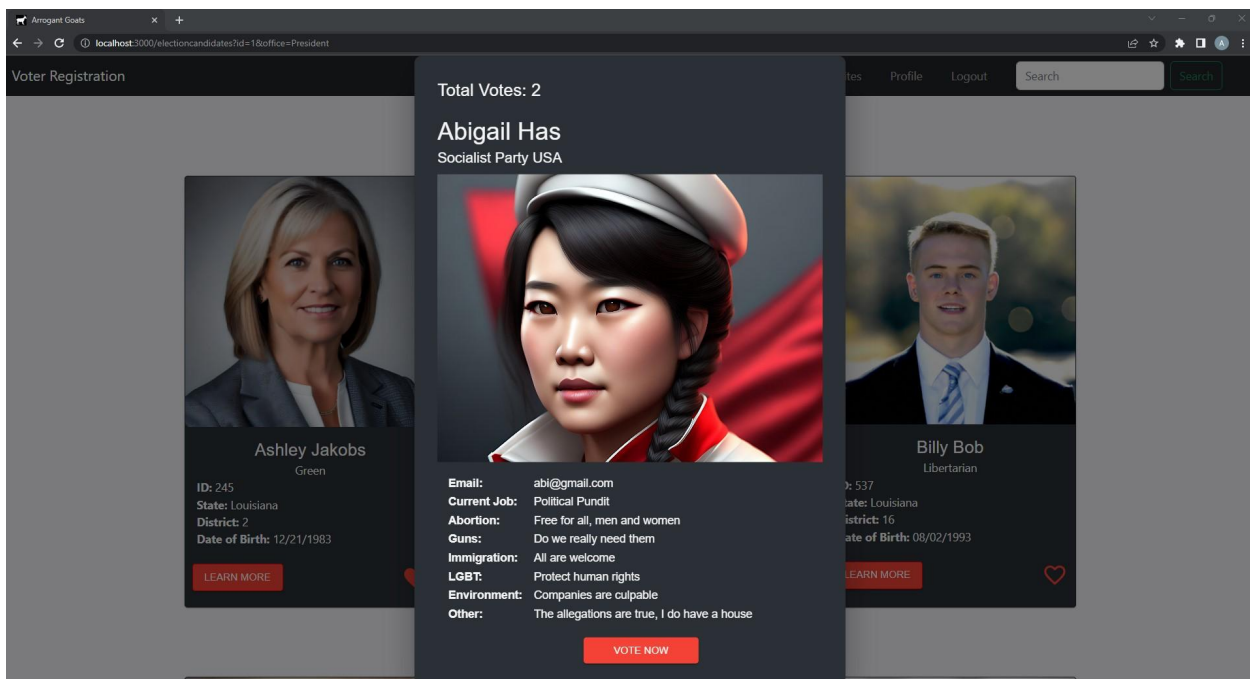
When the user navigates to the elections page, they are shown all of the current elections with a description of them to start with.



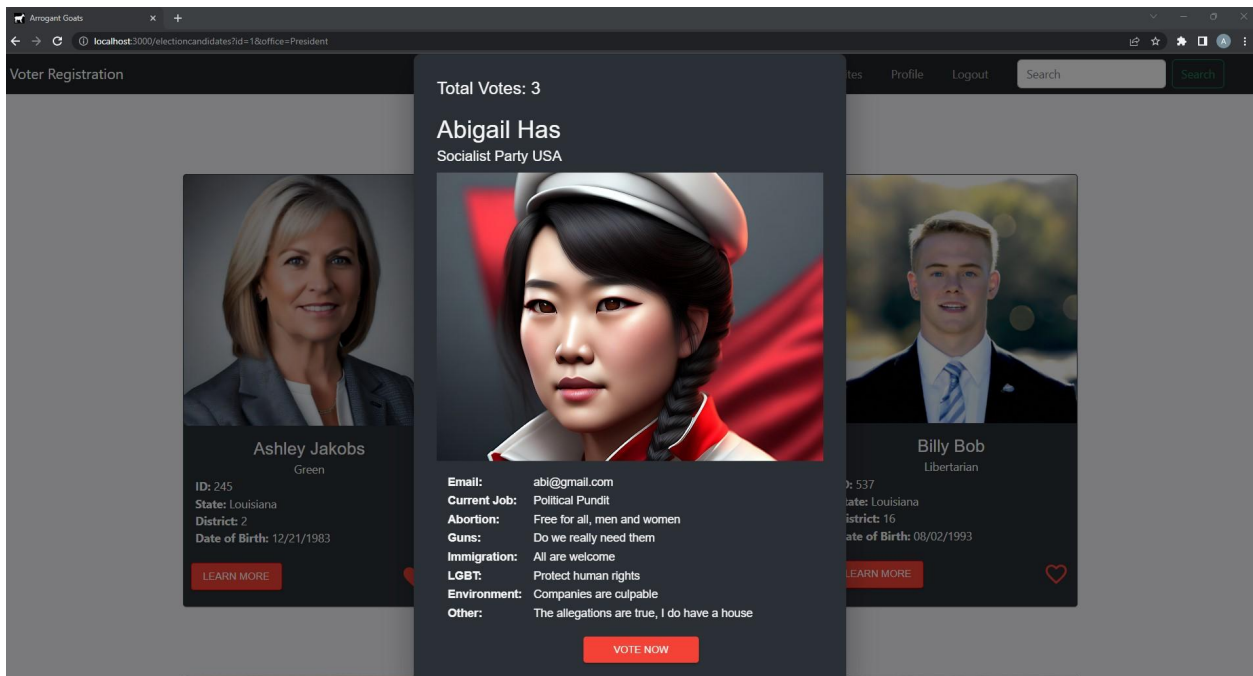
Once they click on the button of the specific election they are interested in, the application grabs and displays all the candidates that are running for that given election.



The user can then click on a specific candidate to view all of their information and opinions on different topics that are essential for most voters to make an educated vote.



If the user is impressed by the candidate's stances on these topics, they can then click on the vote button which will increase the candidates total vote count in the database.



Discussion

In conclusion, we implemented all of the features that we planned to develop at the beginning of the project while also making an aesthetically pleasing frontend that most voters would love to use. Our application isn't perfect by any means and there are many features that need to be improved upon, but we built a very solid foundation that can someday be expanded upon to create a fully fledged voter registration DBMS. Overall, this project was very enjoyable to work on and developing the application went by very smoothly because we split up the work very evenly amongst ourselves which allowed all of us to focus on refining our assigned part of the project.

To quickly give some context on how the work was split up between us all and who worked on what: we had Abdel Rahman Mansour primarily working on developing the server and creating REST API endpoints depending on what was needed for the frontend, Adam Elkhannoufi worked on the candidates page and styling the cards that were used all throughout the frontend, George Adler Buras worked on designing and implementing the MySQL database, John Hudnall worked on the profile page and implementing all of the functionality that is essential to it, Katherine Perez worked on the favorites page and making sure the database is updated whenever a favorite is either added or removed, and lastly Kendall Comeaux worked on the elections page and retrieving all of the candidates that are associated with a given election when it is clicked.

Throughout the process of creating this database management system we were able to achieve many of our goals, but we were also faced with several problems and limitations. Initially we were struggling with getting certain frontend components to communicate properly with our backend (update/post/get), but after working with the backend developers we polished our code to work correctly. We also faced a problem with user data persisting through the user's session in the browser because after page refreshes the entire site forgets the user. To fix this we changed our backend to use `sessionStorage` and `sessionID` to ensure the user is not forgotten until they log out. Lastly, we had an issue when normalizing our database. We determined that in order to put our database in Boyce Codd normal form, we would need to create a new person

table with the attributes first name, last name, SSN, DOB, email, phone number, party ID, district number, and state. The SSN would be the primary key for the person table and would be present as a foreign key in the voter and candidate table. However, we decided that this decomposition would be better left undone, because the overlap between the voter and candidate tables is minimal and doing the decomposition would complicate all of our queries.

Although we completed this project to the best of our ability, we did recognize some features that would be beneficial to add in the future. One of these features incorporated an ``<iframe>`` of Google Maps using Google's API, to display pins where your favorite candidates will be having a Campaign event. This would need a new table using candidateID as a foreign key and campaignID as the primary so that we can display a legend with each favorite candidate having a unique color pin on the map. After selecting a pin it will bring you to that specific event's page which has all the pertinent information and images needed by the user. We also would like to make the vote count for each candidate for each election to be displayed in a progress bar fashion with a percentage bar and their party's color to visualize who is leading an election without actually having to click on the specific election. We could also improve the favorites page's functionality by notifying voters when a candidate they have favorited begins running for a new election. Further, we would like to spend more effort validating input to protect the system against SQL injection attacks and hiding sensitive information. Lastly, we could create a more complete election database by creating different experiences for those who are not voters. For example, we could create a different type of profile for election administrators who would have the power to record the tallied votes directly into the database (which would be a more realistic use case for the database than voters directly voting in the application). If we chose to continue allowing voters to directly vote in the application, we would design a much more sophisticated security system to provide end-to-end verifiability, ensuring that the voter is allowed to vote, that their vote is cast-as-intended, collected-as-cast, and tallied-as-collected. In that case, we would also need to implement a voting system that defines how votes can be formed. Can voters vote for multiple candidates? Are voters selecting candidates, ranking candidates, or giving candidates a score? Further, the voting system would need to define how the votes are used to determine a winner. Some voting systems we might consider implementing would include plurality, ranked-choice voting, and STAR (score then automatic runoff) voting.

Thank you so much for teaching this course Dr. Tisha and the Arrogant Goats team wishes you all the best in Florida and your future endeavors as a whole!!

