Sudo rm -rf / Project 3 Writeup

Team Writeup

Overview

The politifind project aims to make political data more accessible to the average person. The project aims to be easy searchable by items such as politicians, bills, and committees. Other data such as how politicians vote on bills and actions on bills will also be available. In addition, users will eventually be able to provide their own input on certain items, such as the popularity of a bill, and this will be available to all viewers.

Members: Andrew Bass, Matthew Bissaillon, Matthew Gramigna, Justin Kennedy

Github: `https://github.com/arbass22/politifind`

Design Overview:

NOTE: We have created a user for whoever is grading this. You can login with username "grader" and password "compsci326"

We started our login/logout functionality with a basic login page that allowed users to be authenticated through the app. Since our users have more information for their profile such as political party, subscriptions, profile picture, etc., we needed to extend Django's built in user model. We did this by making a one-to-one mapping from Our Profile model to Django's user model. After we had this functionality, we templated the UI to ensure that logged in users could not view the profile page without logging in, and we made the navbar show different buttons based on whether or not the user is authenticated.

Next, we gave our users the ability to vote on various bills, including a comment on why they voted that way, for each bill. This is a user creating a new data entity through a form, and the comments they put could be aggregated in automated emails in our application in the future. Lastly, we allowed users to subscribe to different bills/politicians/committees. The data is modified through an POST endpoint we added created to receive an AJAX request from the frontend. A user's subscriptions will appear on the profile page.

Problems/Successes

One problem we encountered was that initially it was quite difficult to get all accurate data in our database from the API we used. There is a lot of political data regarding these bills, so those scripts took a long time. Once we got past that, it wasn't bad to hook up all the authentication. Another problem we ran into was due to the complexity of our models. As we implemented the authentication, some of the models began to change, and it got hard to coordinate everything with the database across all four of us.

We did well at setting up our templates and views such that it was pretty seamless to add the authentication pieces into the templates. We also had success bringing together all of the pieces we worked on individually. Everything from subscriptions and voting to logging in and profiles integrated pretty easily, and we were happy with that.

Team Choice

For our team choice, we are hoping to integrate the Algolia search API (`https://www.algolia.com/`) to make a fast and scalable search functionality that will allow for very efficient searching of the data in our application. Politifind has a ton of data, from thousands of bills to hundreds of politicians and their various voting data, and more. A quick and easy searching function would contribute well to our overall goal of Politifind, which is to make this political data more intuitively and easily accessible to the average person.

Andrew Bass Individual Writeup

For this portion of the assignment I worked on adding the ability to subscribe to bills, politicians, and committees. The subscribe button is the bell icon that shows up near the top of the page. The subscribe buttons will only appear if the user is authenticated, since logged out users will be unable to subscribe. When clicked it sends an AJAX post request to the /subscribe endpoint I created. The request body contains the information needed to add the correct database entries for the subscription. When clicked the button turns yellow and becomes the unsubscribe button, and the tooltip and click action update accordingly.

I chose to use an AJAX request rather than a full form because I did not want the action of subscribing to reload the whole page. It was a bit tricky to get the post request working properly since I need to get the CSRF token into the request. I ended up moving the javascript into a script tag in the template of the subscribe button. This had the added benefit of centralizing the functionality of subscribing and generating the view for the button.

I spent time generalizing the design of the subscription flow so that it would be easily adaptable to the several types of subscriptions we have. This is also the only element of the project so far that used javascript or the bootstrap tooltip class.

Contribution: 25%

Matthew Bissaillon Individual Writeup

Contribution: 25%

Matthew Gramigna Individual Writeup

I initially worked on transitioning all of our mock data to real data. Now, every bill, politician, vote, committee, etc. information is accurate instead of random. This allows our Poltifind users to subscribe to real information now instead of random information. Next, I modified our models to include a one-to-one mapping from a Django auth user to our Politifind user profile, so they could enter additional information about their political views, and have their own subscriptions. Lastly, I templated the navbar and the profile page such that a user can only get to the profile page if they are logged in, and once they are logged in they can either logout or view/edit their profile. I also hooked up the user's subscriptions into the template for the profile.

Contribution: 25%

Justin Kennedy Individual Writeup

In this project I was solely responsible for all voting functionality with the user. This includes templating the bill page, creating the vote endpoint, and modifying the bill view with the appropriate contexts.

Contribution: 25%