TECHNICAL REPORT

https://empoweringknowledge.me/

Briana White
Edwin Silerio
Ojas Patel
Ryan Morris
Travis Langston

Table of Contents

Introduction

Purpose

Our website is a tool for users to find out information about their states, especially in terms of energy consumption and production, representatives in Congress, and different energy-related bills and legislation.

Motivation

Today, more than ever, people need to be aware of where their energy is coming from and how it's being produced. We've found through personal experiences that many people don't know who represents them in government. This means that people don't know who is making decisions on their behalf and, more importantly, what decisions are being made on their behalf. Additionally, it is hard to know the sources from which your energy is coming i.e. solar, nuclear, fossil fuels, etc. Due to the climate-related issues of the modern-day, we wanted to create this website to encourage users to figure out where their energy is coming from and contact their representatives if they disagree with those choices.

User Stories

Phase 1 Stories to Developer

We served as the customers to congressfor.me.

User Story 1:
Currently, there is a lack of variety in color on the website. As a user, I would like to see more color. Please add more images or anything else that will accomplish this.

User Story 2:
As a user, I want to be able to read all of the text on the page. It is currently difficult to read the description because the color is too similar to the background. Change the color to something that contrasts the background.

User Story 3:
As a user, the splash page is not engaging. I would like to see an image that represents the website's idea. Please add an image that will hook the user in and gives them an idea of what your website is about.

User Story 4:
As a user, I want to know who created the website. I want this information to make sure the information and authors are credible. This can be accomplished by adding images and information on the creators.

User Story 5:
As a user, I would like to be able to reach a few significant pages on the site from any page on the site. Home, about, and the model pages are the most important at this stage. The navbar should be concise with clickable buttons that redirect the user to the other pages.

User Story 6:
As a user landing on the splash page initially, I would like to be able to access the rest of the site from the splash page and see the model pages. It would be helpful if I could start from the splash page and move to the rest of the site, and if the splash page was more useful.

Phase 1 Stories from Customer

We served as the developers for Stock Overview.

User Story 1:
Need to add more images and the image size is little big, so making the size smaller will look better

Response: We added an image on our splash page and added images of ourselves in the about page to make it more inviting for users.
Estimated Time: 2 hours
Actual Time: 1.5 hours

User Story 2:
As a user who is researching certain states, I would want to filter/order states based on the relative breakdown of their different types of energy consumption. Ex: high-low states in terms of % renewable energy, or filter states that have > 0% of a certain energy type.

Response: Filtering will be implemented in a future phase
Estimated Time: 6 hours
Actual Time: n/a

User Story 3:
As a user who is viewing a specific piece of legislation, I want to see all the politicians who voted for or against it (as applicable).

Response: Filtering will be implemented in a future phase

Estimated Time: 3 hours
Actual Time: n/a

User Story 4:
As a user, I want to find politicians that are relevant to me by filtering based on political party and location.

Response: Filtering will be implemented in a future phase
Estimated Time: 4 hours
Actual Time: n/a

User Story 5:
As a user who is looking for legislation/policies that may be relevant to me, I want to be able to filter legislation based on certain characteristics or categories (e.g, home, commercial, apartments), since energy regulations might differ for each of these categories.

Response: Filtering will be implemented in a future phase
Estimated Time: 4 hours
Actual Time: n/a

User Story 6:
As a citizen who is interested in a particular candidate, I want to be able to see all the legislation and policies a certain candidate has supported/voted for.

Response: Currently, we have listed different sponsors and cosponsors for legislation and policies.
Estimated Time: 1 hour
Actual Time: 1.5 hours

Phase 2 Stories to Developer

We served as the customers to congressfor.me.

User Story 1:
As a user, I want to ensure that the website is reliable and works as expected. This includes making sure the front end works as expected and that there are no errors in the backend that could lead to failure. This can be accomplished by creating tests for the front end and back end.

User Story 2:
As a user, I don't want to see all of the data for each model on one page. This can be very daunting and make it difficult to focus on the data presented. Instead, limit the number of instances to some number, say 10, per page and allow users to flip through pages.

User Story 3:
As a user, I'd like to have a preview of the model before I click on it. There are thumbnail pictures on the donor model page, but not on the other two. Add a small picture for each district and representative in the tables.

User Story 4:
As a user, I would like the splash page to help me find a few instance pages I might be interested in. Currently, the "Learn more" button does not add functionality, since I can access the representative model page through the nav bar. You could possibly choose a few random pages to link to in a slideshow.

User Story 5:
As a user, I want the data on the website to be recent and dynamically pulled rather than hard-coded. This way, if there is a change in information, such as the district that a member of Congress represents, the data that is displayed on the website is changed. This can be accomplished by making sure the front end consumes a RESTful API that pulls the data.

Phase 2 Stories from Customer

We served as the developers for Stock Overview.

User Story 1:
The number of commits are not synchronized. There are 90 commits listed on the Gitlab, but 77 listed on the website. This needs to be changed so the numbers are synchronized.

Response: After reading this customer story, we realized that there a mismatch in the number of commits. When we looked into it, we noticed that the missing commits on our About page were due to not counting commits from merge requests. As a result, we changed the way we call the GitLab API and parse through the commit history data.
Estimated Time: 1.5 hours
Actual Time: 1.5 hours

User Story 2:
There needs to be more social media or images to each instance. There are currently only one or two and each instance page needs more than that.

Response: After reading this customer story, we realized that our APIs have more social media information that we can add to our database and therefore present to the user of the website. For example, there are Facebook usernames, Twitter handles, and CSPAN IDs for all of the members of Congress and GovTrack URLs for all of the bills which can be presented.

Estimated Time: 2 hours
Actual Time: 1.5 hours (for adding to the database and displaying the links, not for embedding the actual Facebook/Twitter feeds to the website)

User Story 3:
The number of issues on GitLab does not match the number of issues on the website. There are more issues closed on the GitLab than there are on the About page.

Response: We believe that the issue is because of not correctly using the right usernames to pull the Issues data.

Estimated Time: 1 hour
Actual Time: N/A (Still in progress)

User Story 4:
As of now, there are only 3 instances of each model per page. There should be 9 by the end of Phase 2, so you need to expand to 9 instances and make them visible on each model page.

Response: The three instances per page were due to the requirements of Phase 1. We have gone ahead and made it so 9 instances are visible per model as we have now used our REST APIs to gather data and stored said data in MySQL databases.

Estimated Time: 4 hours
Actual Time: 6.5 hours

User Story 5:
In the energy bills model page, there should be a link when I click a sponsor or politician. Now, it is not linking. Please make it linkable to those instances.

Response: We are working on creating a join table and setting up foreign keys so that all politicians can be linked to the energy bills that they have sponsored or have been associated with.

Estimated Time: 5 hours
Actual Time: N/A (Still in progress)

Phase 3 Stories to Developer:

We served as the customers to congressfor.me.

User Story 1:
As a user, I want to see clearer images for all of the districts. If I am interested in learning about a certain district, it is hard for me to see where it exactly is in the state with the current images. Additionally, it means that if I am doing research and want to share/copy/download the image from your site, I won't be able to as it is blurry.

User Story 2:
As a user, it is difficult to navigate over the current state of the pages for the models. It is hard to view three different layers of buttons and to see all 20+ or so pages. Maybe instead have three numbered pages visible and add arrows to navigate over the pages.

User Story 3:
As a user, I am very interested in learning about different donors. As of now, it is cool that I can see which parties they donated money to, but I would like to learn more about the company/organization/group, etc so I know more about why they donated the ways they did.

User Story 4:
As a user, I am very interest in learning about different members of Congress. Currently, when I click on their names, it loads infinitely. However, I would like to see information about them, such as a picture, a bio, how long they served, state, district, age, twitter feed, etc.

User Story 5:
As a user, I am curious about which donors donated in which cycle. I have noticed that while most donors gave money during the 2018 cycle, there were some donors, like Chicago Board Options Exchange, gave money in 2016. With so many results with 2018, it makes it hard to find donors for other cycles, so I would love a way to filter the donors based on cycle to make it easier to find the data I want.

Phase 3 Stories from Customer

We served as the developers for Stock Overview.

User Story 1:
There needs to be a search bar on a splash page. It will let customers be able to search comprehensively. It will show whatever data that is satisfied with a value typed on it.

Response: We went ahead and included a search bar so our users can access the information they want faster. We believe this will significantly improve the user experience.
Estimated Time: 10 hours
Actual Time: 12 hours

User Story 2:
The current cosponsor is shown as floats on the page for Energy Bills. This does not make much sense since you don't have a fraction of a cosponsor. Also, when there is no cosponsor, a more reasonable approach is to set them as 0s when users are viewing instead of "nan" (your database can still have null). As a user, I would want a more clean format on these attributes.

Response: We understand that this can be confusing to the users. To fix it, we changed the data type in our database from Strings to Integers so that it truncates the decimal. Additionally, we set the NaN values to 0.
Estimated Time: 1 hour
Actual Time: 1 hour

User Story 3:
Many sponsor links provided on Energy Bills only lead to blank pages with nothing, even for many sponsors I can see their pictures in the links under the attribute title. As a user, I would want more information about these sponsors and notice when no information is unavailable.

Response: We have fixed our links in the Energy Bills table to link directly to the Congress Members instance pages. This means that the sponsor links shows all relevant information on said Congress member.
Estimated Time: 2 hours

Actual Time: 2 hours

User Story 4:
All states' energy use attribute is "Coal:". This is clearly incorrect and limited. Also for the attribute "Bills Sponsored By State Representatives" is always empty. As a user, I want to access the correct information on these attributes.

Response: We had a lack of information in our database and used Coal as the default value. To fix this, we used one of our data sources to pull data on the top used energy sources for each state and appended 3 columns to our States data table.
Estimated Time: 2 hours
Actual Time: 1.5 hours

User Story 5:
Right now the pagination bar is too short and only allows users to go to one page after the current page. As a user, I would want the ability to skip multiple pages at a time.

Response: After doing research with other popular websites, we found that they also had pagination under a similar model of ours.
Estimated Time: 0 hours
Actual Time: 0 hours

User Story 6:
Right now the first attribute on Energy Bill Page is "#". However, after going through more than 10 pages, that attribute is always empty, which takes extra space and is confusing to users. Consider removing it or actually put useful information on it

Response: We recognize that this is unnecessary and can worsen the user experience. It crams the table shown to users and does not provide useful information so we removed it.
Estimated Time: 1 hour
Actual Time: 1 hour

User Story 7:
Currently, there are many politicians with no image. As a user of the website, I would like to access to images of all politicians. If there is a case when absolutely no image can be found, I would at least need a generic picture instead of a "broken image" icon.

Response: We were having issues setting up a default picture on the front end for the last phase. As a result, we decided the best thing to do was to set the image URLs in our database. When we used the image URL with the congress member’s id appended to it, if it returned a non-200 status code, we went ahead and set the URL as a default image URL.
Estimated Time: 1 hour
Actual Time: .5 hours

APIs

GitLab API

We pulled data on issues closed and commits pushed by our team members from GitLab's API to dynamically update the team member statistics on our About page. This is currently performed by the front-end within the About.js file, but in the future will be implemented as a call to our back-end's API. Preliminary code for the back-end implementation can be found within backend/gitlabapi.py and backend/main.py.

ProPublica API

This API provides us with information on Congress members, as well as bills that may have been passed, defeated by vote, or introduced without a vote yet. We can determine the Congress member responsible for sponsoring a bill, how different Congress members voted on particular bills, party affiliations of cosponsors, social media information for a Congress member, and other information to populate our instance pages with or filter our model pages with.

DataUSA API

DataUSA provides a wide variety of general data on states, from which we will select additional attributes for our state model that allow for new filtering and visualization methods. This includes an extensive selection of economic data and population data, which are intended to add depth to the user's understanding of energy production and consumption within a particular state.

EIA API

These data sets, provided by the U.S. Energy Information Administration, allow us to scrape data on energy production and consumption by state to collect information for our state model. The API provides this information by state and provides a further breakdown by year, allowing us to make connections between yearly data, bills introduced during particular Congress terms, and Congress members that represented the state during a particular year.

EmPoweringKnowledge RESTful API

Our API documentation describes the current implementation of the API. We used Flask-Restless and SQLAlchemy to generate our API and interact with our database. Data for Congress members, congressional energy-related bills, and states are paginated to reflect the front-end's display on the model pages. The caller can send a primary key at the end of a model call to

retrieve data on a specific instance, which populates our instance pages. In some cases, Flask-Restless provides filters for retrieving additional information for our instance pages, such as retrieving all the bills sponsored by a particular Congress member. The documentation contains examples of how we use these filters, where applicable.

Postman

Models

Data

State

- Name
- Population
- Median household income
- Primary political party
- Energy consumption over 1993-2017
    - 10 subcategories: petroleum, biomass, coal, geothermal, hydroelectricity, natural gas, nuclear power, solar energy, wind energy, wood and waste

This model represents states in the United States of America. We want to represent each of the 50 states and information about them, such as population, income, political leanings, and energy consumption.

The attributes we have selected for states will allow users to apply filters and sorting in future phases that may reveal – or debunk – patterns in energy consumption as it relates to population size and a metric of state wealth.

Bills

- Title & short title
- Official bill ID
- Bill type (also indicates chamber of origin)
- Committees
- Sponsor’s BioGuide ID
- Party count of cosponsors
- Dates of introduction, HofR passage, and Senate passage
- Summary & short summary
- Congress.gov and GovTrack URLs

This model represents energy related bills that have been introduced in the US Congress.

The attributes we have selected for legislation will allow users to find bills sponsored by the same Congress member or same party and discover patterns in bills sponsored or supported by different parties. Our database stores bills dating back to 1993, which is the earliest available data from ProPublica.

Member of Congress

- First name & last name
- BioGuide ID (member_id)
- Political Party
- State & district
- Percentage of votes along party line
- Facebook account, Twitter account, personal website URL & cspan_id
- Phone number & office information
- Image URL

This model represents members of the United States' Congress. It includes members of the Senate and the House that have served from 1993 to 2017.

The attributes we have selected for Congress members allow users to filter for Congress members from the same party and determine links between partisanship and sponsorship or voting tendencies for energy bills. The user will also be able to filter by state, or by Congress members that served during a particular term. Our database stores Congress members who served as early as 1993, to match our bill and state energy data.

Connections between Data

Congress members represent a particular state; the party of elected Congress members is influenced by the general partisanship of the state. Stances on energy-related issues are sometimes influenced by prevailing party positions on these issues. Their stances may also be influenced by the forms of energy that are predominantly produced or consumed within the state they represent. A Congress member's stance on energy-related issues has an impact on legislation considered and passed by Congress. A particular Congress member, representing a particular state, sponsors a bill, which other Congress members may cosponsor, and eventually all vote on. Patterns in legislation sponsored by Congress members from a particular state may also emerge.

To implement the linking and relationships between the data, we are using the Congress member ID as a primary key for the CongressMember table and are using the state they represent as a foreign key into the States table. Additionally, the States table have the state name as a primary key, but also use it as a foreign key into the tables for each of the energy sources. Similarly, in

the Bills table, the Bill ID is the primary key and the sponsors have the same ID as the Congress Member IDs, so we use the sponsor ID as a foreign key into the CongressMember table.

To further our data connections, we had some complications inserting the same Congress member into the CongressMember table for two different terms, so we created a Join Table so that given an ID, it will return a list of chambers and session numbers.

Tools

React
React is a JavaScript library that was used to build out the frontend of our website.

Reactstrap
Reactstrap is a tool that has easy to use React4 Bootstrap components, helping to make our website beautiful, clean, and organized.

Slack
Slack is a tool that allows for easy team collaboration and communication. It was essential for our team's success by allowing communication to be organized into tasks and groups. Slack made it easy to share files and ideas without being physically next to each other.

Postman
Postman was used for two reasons. It helped to generate code needed for pulling data from our model data source APIs. Additionally, it was used to create the schema and documentation for our API.

GitLab
GitLab is the version control system we used.
Our GitLab repository URL is https://gitlab.com/patelojasv/cs373-web

Flask
Flask helped create our RESTful API so users could access our data.

Flask-Restless
Flask-Restless helped create our API by interacting with our database using SQLAlchemy and generated our "sends" and "requests".

Grammarly
Grammarly was used to ensure that there were no grammatical errors in any of the text on our website.

Namecheap
Namecheap was used to acquire a free URL.

Amazon Web Services (AWS)
AWS is a cloud computing platform used to host our website. We also utilized AWS EC2 to host our backend services, namely our API, and AWS RDS to manage our MySQL database.

SQLAlchemy
SQLAlchemy was used in our Python scripts for easy table creation, insertion, and management of our MySQL database.

MySQLWorkbench
MySQLWorkbench was a key tool that helped in testing our databases. It provides visual representation of our MySQL database and helped create easy SQL queries and commands. For example, it helped when determining whether data was being entered into tables correctly and made it simple to drop tables or delete rows when needed.

Docker
Docker was used development so that we can run our code in the Docker image and have all of our dependencies and libraries pre-installed. This makes it very easy to run tests and migrate from machine to machine without having the lengthy set up process.

Black
Black was used to format our Python code.

Pandas
Pandas was useful in helping us analyze our datasets and trim them so we could create useful subsets to push into our MySQL database.

Jupyter Notebook
Jupyter Notebook was used to test snippets of Python code and to make it easy to visualize and interact with our dataset.

Features

Pagination
In order to implement pagination for each of our model pages, we had to add support from both the frontend and backend. When the page loads, it fetches data from page one of our API that

returns nine elements at a time. The front end displays the nine elements returned and uses reactstrap pagination components for the buttons to click through each page. The current, last, next, previous, and first pages are stored and set according on a button click. When a button for a new page is clicked, another API call is made to fetch and display the next page of data in the backend. This allows the user to click through multiple pages of data for the politicians, states, and energy bills.

The backend of pagination utilized the Flask Restless API Manager. The API Manager helps to create our desired API calls and handles routing with collection names. It ensures that the API returns 9 results per week with the "max_results_per_page" parameter, which we set to 9. A sample of such a line can be seen below:

```
manager.create_api(CongressMember, methods=['GET'], max_results_per_page=9, collection_name='congressmembers')
```

Filtering

To filter by different attributes in each of our model pages, we utilized Flask-restless' "filters" object feature. We build a filter string that is appended to our API call. The user can apply multiple filters to the data since our code keeps track of the parameters; Each filter chosen adds a "filters" object to our filter string. Only data that fits the parameters specified by the user, are displayed. We have included filtering in all of our models. For our Politicians model, we can filter by title, so as to exclude either Representatives or Senators. We can also filter by party based on Democrats and Republicans. Lastly, we included filtering based on state. In terms of our States model, we included filtering over the majority political party, by population ranges, by income ranges, and by primary energy source. For our Energy Bills model, we included filtering for committee and year.

Searching

To implement site wide searching, we utilized the Flask Restless' "filters" function as a query parameter in our API call. For each term the user inputs to the site wide search bar, a query string is built to filter the data returned from the backend. An API call is made for each of the model pages and is rendered in each of the respective tabs. The models are only returned if any of the attributes contain any of the search terms. The user can click through the State, Politician, or Bill tabs to see the search result for each model. As for highlighting, we imported the react-highlight-words component that highlights any of the given search terms in the text. Searching in our model pages is done the same way, but it only fetches the data for the specific model.

Sorting

Our sort feature also makes use of Flask-restless' API call features. In searching and filtering, we used "filters" object in our filter string. For sorting, we used the "order_by" object. The user specifies the order they want the data presented, either Ascending or Descending. The order chosen is add to our "order_by" object and is appended to our API call. The data returned is sorted either ascending or descending on first name for politicians, name for states and title for energy bills. We wish to have been able to include sorting as arrow keys directly into the table but struggled to get that functionality implemented. Additionally, we wanted to have been able to

sort by ascending or descending order for every element in the table, rather than just the primary key, but again, were not able to get that functionality implemented.

Testing

To run tests, we set up a CI/CD pipeline with our .gitlab-ci.yml file. This file allows the pipeline to run different testing stages and emails/notifies us when tests fails so we can adapt our code and website accordingly.

Selenium

We used Selenium as well as ChromeDriver for testing the front-end of the site. ChromeDriver allowed Selenium to open up a new Chrome browser to check for front-end functionality. To use ChromeDriver, it had to be installed and placed in the system's path. Selenium allows for testing different buttons on the website, such as those in the navigation bar, on instances, and links in the About page. Additionally, it allows for testing the pagination.

Mocha

We used Mocha and Enzyme to test our javascript code for our front-end. Mocha creates a test suite that runs all the tests in our App.test.js file. Enzyme allows us to render shallow copies of our components so we can test them.

Postman

We used Postman to test our RESTful API. Currently, the test suite checks for a 200 status code, present Content-Type header, correct pagination, and the presence of all expected attributes in the response.

Python

We used Python's unittest module to test our backend code and the API. These tests can be found in the file backend/tests.py. The API tests try certain requests and make sure they return without error and contain the expected attributes. The backend code tests check to make sure the backend functions for getting model data from the restful APIs are working correctly and returning the correct type.

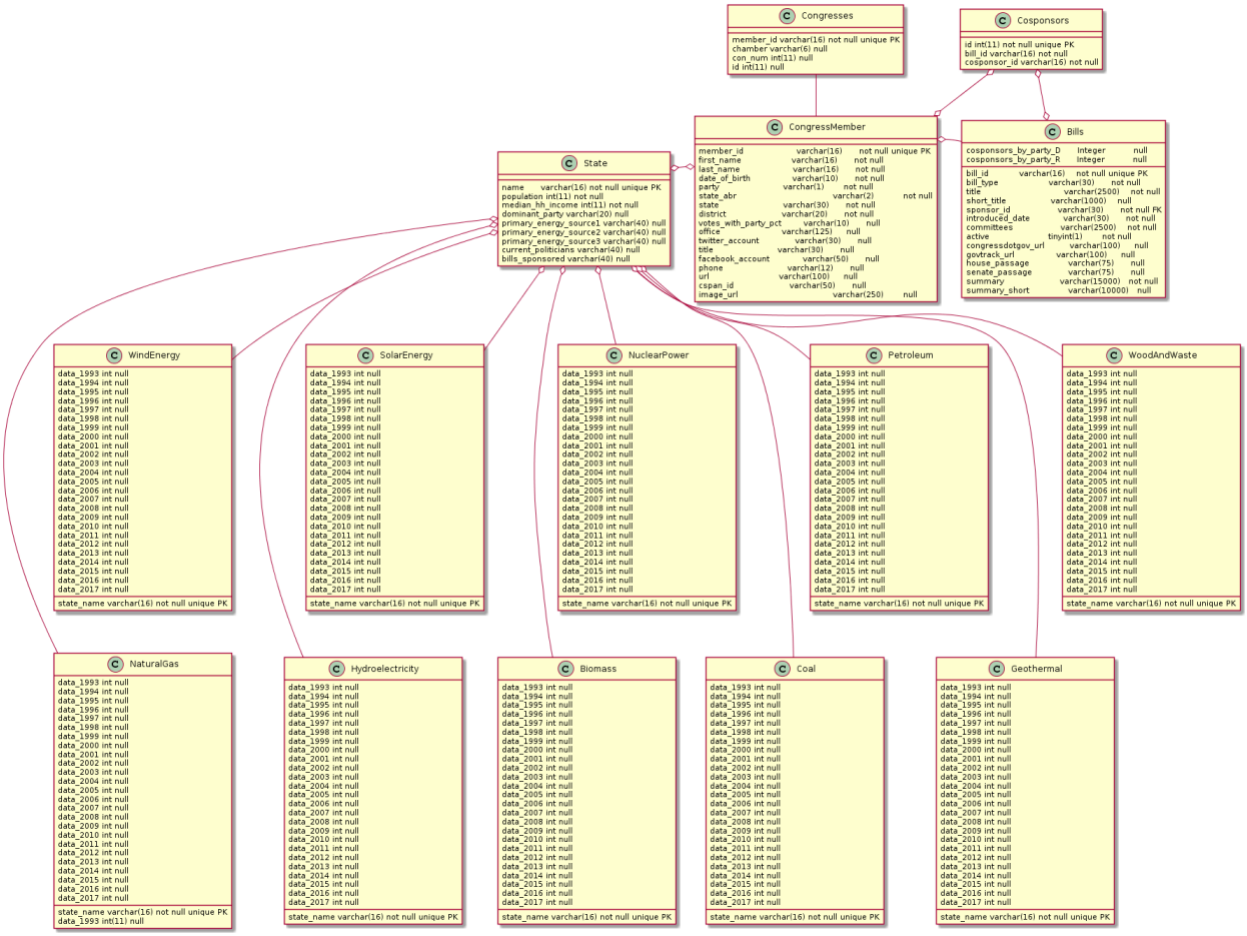
Database

To set up our database, we used AWS RDS to set up a MySQL database. We chose MySQL over alternatives such as PostgreSQL as after some research, we found that PostegreSQL had some more features, but those features were unnecessary and that MySQL would have an easier setup.

After our database was set up, we interacted with it using SQLAlchemy and MySQLWorkbench. SQLAlchemy was very helpful for implementing our schemas and creating our tables and then pushing the data we gathered from our APIs into the tables. MySQLWorkbench was very key as it has a visual interface that lets users see the tables in a database, all of the columns of the tables, and the data in each table. It made it easy to verify our data entry worked and made it easy to run basic tests such as creating sample tables and adding sample data. SQLAlchemy was a very good tool too as it required minimal knowledge of SQL since you never really had to write complete SQL queries and statements.

Our actual database comprises of 13 tables. Ten of these tables represent energy types and have rows representing each of the states with each of the columns representing the data from those years. Then, there is a table on States, which has the name of the states along with information such as the population and median household income. Lastly, we have a CongressMember table that includes key information on the member of Congress such as name, party, state, district, etc., and a table on Bills, which includes information such as a summary, title, sponsors, and bill type.

For the next phase, we added 2 more tables. These include one on Cosponsors which links a bill to its cosponsors and one on Congresses for Congress members that served multiple terms. This links a member id to different chambers in Congress and the Congress session served.



Hosting

Frontend

To host our website, we used AWS S3, CloudFront, Certificate Manager, and Route 53. We used Namecheap to acquire our URL and DNS. In S3, we have a bucket that holds all of our files for our react app. The bucket also acts as a host for our static website. To redirect https and provide fast access to our website, we used CloudFront. ACM provided us with a custom SSL certificate for our website. Lastly, Route 53 allowed us to connect our Namecheap domain name to the AWS DNS.

Backend

To host our backend server, we created an Ec2 instance and push our backend files onto it. The instance runs our flask database. In order to reroute from HTTP to HTTPS so our Frontend could pull from it, we use CloudFront. Our flask app runs on port 5000 so in the CloudFront distribution, we had to change the port it was listening to, to 5000. Once our distribution was up and running, we added a new record to our hosted zone in Route 53 for our main website, that points to the CloudFront distribution.

Visualizations

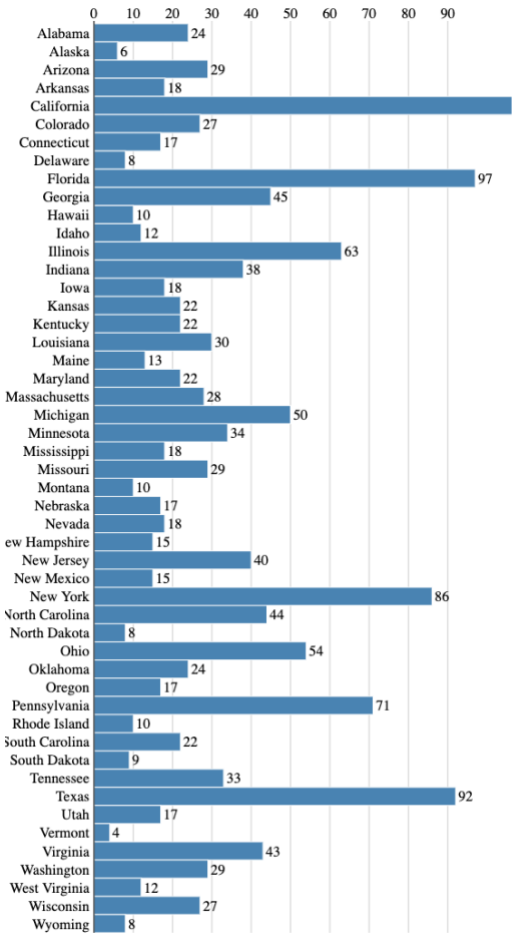
EmpoweringKnowledge Visualization #1

What is this visualization showing:

Our first visualization uses the States model and the CongressMember model to see how many representatives there are per state. It is making use of the fact that we created a foreign key relationship between the CongressMember table and its “state” attribute to the primary key “state_name” in our States model table. This meant that when we used an API call to our “states” API, we had access to the Congress members for each state and simply had to find the length of the “CongressMember” attribute under the State table for each state.

How this visualization can be used:

This visualization can be used to understand that the number of representatives per state can differ greatly. Some states have very few, such as less than 5, while others have more than 25. This brings up the question for users why some states have significantly more representation than others. However, it does answer questions for the user such as what states are the most populous as those states have higher representation. Additionally, it can be used to see which states are easier to influence in terms of changing energy legislation as a state with fewer representatives has fewer points of contact to lobby to push change.



EmpoweringKnowledge Visualization #2

What is this visualization showing:

Our second visualization shows a pie chart comparing the total number of energy bills sponsored by each party. This took into use the Energy Bills model and the Representatives model. We used the “sponsor” attribute for the Energy Bills model as a foreign key into the Representatives model to find out what the party of the sponsor was using the “party” attribute of the member of Congress. We displayed the Democrats in blue on the chart, Republicans in red, and other parties in gray.

How this visualization can be used:

This visualization can be used to see which party sponsors more energy bills. It can answer questions such as which party invests more of its platform into energy reform or change. As a pie chart, it has an even scale as the data is represented in 360-degrees, so there is no malformity in the relative size of the three slices.

EmpoweringKnowledge Visualization #3

What is this visualization showing:

Our final visualization for our own site was a scatterplot that compared the median household income of a state to the population of a state. We used the States model and took advantage of the "population" and "median income" attributes of the model and plotted the data points for each of the states.

How this visualization can be used:

This visualization can be used to see if there is a correlation between household income and population. While this may seem irrelevant to our website, the household income data we received was post-tax, so we wanted to see if there was a trend in higher population states having higher taxation on energy usage, which could result in a slightly lower (less than 1 slope when the scales on the axes are normalized) slope.

Congressofor.me Visualization #1

What is this visualization showing:

Our visualization is showing how much money each political party received from donors. We used the Donors model and looked at the "Democratic Contributions" and the "Republican Contributions" attributes to aggregate them per each donor and create a total for each column. We then created a bar chart with a starting y-axis of 0 so it can visibly be compared by height how much the donations differed per party.

How this visualization can be used:

This visualization can be used to determine which party has received more money from outside donors. This can answer questions for users such as whether receiving more donation money results in a higher win (election) rate and how much money each party actually receives. It can also lead to further investigation on topics such as how much money goes to presidential candidates vs local candidates, whether the government should allow for outside donations for candidates, and what the money is actually used for.

Congressfor.me Visualization #2

What is this visualization showing:

Our second visualization is showing the number of districts per state compared to how many unique donors donated to representatives in that state. We made use of the Representatives model to examine the “Top Contributors” attribute and the “State” attribute to figure out how many unique donors there were for each state. We then used the “Districts” model to figure out how many districts there were per state using the “State” attribute. From there, we were able to create a scatter plot with the x-axis being the number of districts and the y-axis being the number of unique donors.

How this visualization can be used:

This visualization can be used to see if there is a correlation between the number of districts and unique donors. Typically, the number of districts corresponds to the population of the state, so this would be a good indicator if donors try to donate more to larger states or states with a smaller population with the effort to change the vote. It can also answers questions if smaller states with large industry interests, such as major oil and gas producing states like Alaska and Oklahoma, receive large donations as they have important industrial implications.

Congressfor.me Visualization #3

What is this visualization showing:

Our last visualization for our developer is how many representatives there are per party. We did this using a pie chart with two slices: one for Democrats and one for Republicans. We used the Representatives model and utilized the “Party” attribute within

How this visualization can be used:

This visualization can be used to see what the state of the House of Representatives was when the website was created. This is important to put the information on donation amounts to representatives and parties into context. For example, if it was a primarily Republican house, it would give clarity on why either Republicans were receiving more or less donations than Democrats.