Allegheny County Policing project (ACPP)

Final Report

12.08.2022

CS 1980/1640 Capstone team (Fall 2022)

- Anthony Lucchitti, Aaron Palmer, Lingfeng Fan, Frederik Sorg-Taylor

University of Pittsburgh

CASSI Grief to Action

Background

Allegheny County Police Project aims to bring more transparency to the police accountability process at the local level. The website's search query allows users to easily navigate municipal police departments, look up information in police contracts, and become more familiar with the ways these contracts govern police accountability. The site quickly queries police contracts from almost 100 of the 108 Allegheny County governmental police departments available for convenient search and download.

Description

In order to relieve problems like police misconduct and accountability, legal protection, lack of transparency, and political fragmentation, providing a way to easily access data surrounding policing is significant. ACPP consolidates information in easy-to-navigate ways for citizens, researchers, and activists interested in more police accountability in our county.

1) Explore police department Map.
    - Find local and neighboring police departments, etc.
    - Link to police department website, budget percentage in 2019
    - Keywords in contracts.
2) Search police contracts
    - Search and compare over 100 contacts from Allegheny County. Filter contracts by location.
3) Pittsburgh Complaint FAQ.
    - Learn about the process of filling a police misconduct complaint at Pittsburgh.

Instructor/Mentor

Luis Oliveira

- CS 1980/1640 Capstone course instructor

Nicholas L. Farnan

- Capstone advisor (Pitt SCI)

CAASI Director: Sera Linardi (GSPIA)

CAASI Tech leads: Patrick Gavazzi, Collin Griffin

Onboarding Challenges

We experienced many challenges during the onboarding phase of the project. The first obstacle faced after being assigned this project was understanding what the Allegheny County Policing Project is, who is involved in the project, and what exactly our work would be this semester. The folks from CAASI were welcoming and offered us many resources to acclimate ourselves to the project's motivation; while the resources were helpful, it was overwhelming at times to sift through the many videos, articles, and Google Drive folders. Nonetheless, we came together as a team and were excited to start working on the project.

Then came the question of what exactly we would be working on. It seemed as if every person we talked to had a different set of goals. The CS 1980 website contained a description of changes to be made this semester, the folks at CAASI listed off their own set of improvements they would like to see, and the resources shared with us listed off additional enhancements derived from conducting user studies. How were we supposed to aggregate these into a concrete set of goals? It was challenging to say the least, but we learned the importance of communication and compromise when working in a client-consultant relationship.

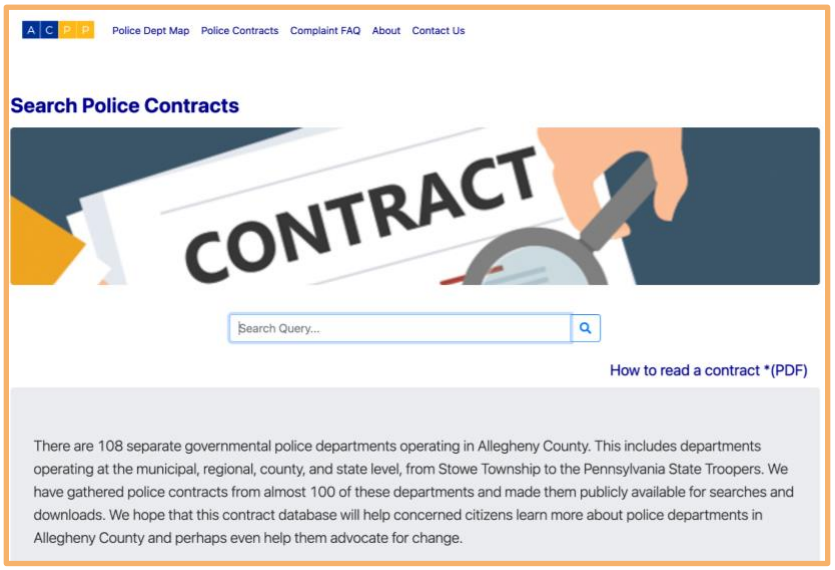
After nailing down a concrete set of goals for the semester, it was time to dig into the code. We as a team followed the explicit guide for setting up the environment locally and had only a few problems running the server on our machines. We began inspecting the code and understanding the general structure of the website. Perhaps naively but arguably logically, all of us cloned the master branch. We assumed the currently deployed website was based off the master branch and that we would create branches off of master to add our features. Little did we know, the master branch is neither the branch in deployment nor the branch our work would be based off of. The team later clarified this and we remedied any confusion. The branch we are working on has a slightly different implementation of the search bar functionality, and *this changed our goals regarding this feature*. The takeaway here was learning the importance of confirming that the assumptions being made are in fact accurate.

Completed Goals

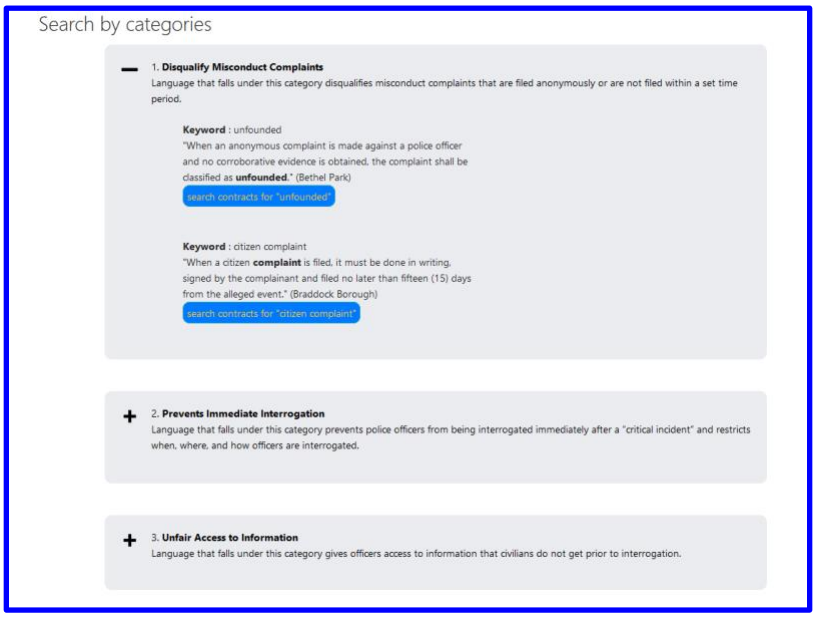
Improve Access to Search Bar Functionality

The previous search bar does not offer an intuitive way for users to access the search bar functionality in terms of searching police contracts for each municipality. In detail, the previous interface shows a link for users to be directed to the police contract searching page, which is not friendly enough. The extra step of being directed to the searching page is unnecessary. Therefore, we put this as our first frontend goal to migrate the search bar directly on the contract page to highlight the primary role of that page and grant users who might omit the text stating the existence of a search bar.

In brief, this is a relatively easy task to achieve as the first frontend task. We rearrange the existing React frontend framework to migrate the location of the search bar to the upper level interface, which saves the user an unnecessary step to redirect the page and makes the searching functionality clear and easy-accessible. The hardest part in this task is to get familiar with the existing framework and how each file interacts with each other. The accomplishment of such a migration gives a head start for the subsequent goals.



Restructure the new Category and Keyword System



Originally the category system was large and unwieldy. It also had the challenge of being in an awkward spot. This largeness made it also awkward/impossible to move it over to the page where database searches occur. The point of the category system was to give ideas to searchers to find what they are looking for, if they may not know what they want to search for.

Another problem with it that was aimed to be tackled outside of our original goals, was that different counties and activist groups had different goals. This becomes a problem because, individually, each category was hardcoded into the front end.

```
"Disqualify Misconduct Complaints":
    "unfounded",
    "citizen complaint"
],
"Prevents Immediate Interrogation":
    "interview",
    "critical incident"
],
"Unfair Access to Information": [
    "interrogation",
    "accused"
],
```

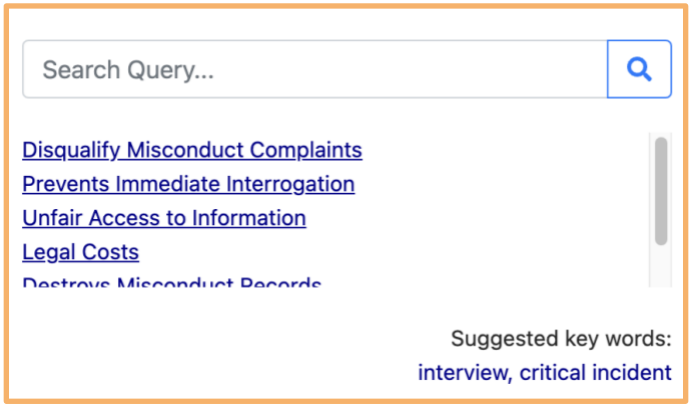
In order to remedy this a lot of thought was put into how to change this for greater usability for administrators. A JSON was used to store a dictionary of categories and their related keywords. So that it could be generated at runtime, without changing any code from inside the program, that would need a web programmer to do.

This was meant to be temporary, however it stayed this way due to a few challenges. One, the tech lead of the project said it was a good solution. So there was not much incentive to further explore it. However, it felt a bit lacking. Initially it was meant to be a CSV, so it would be very easy for anyone with a spreadsheet program to edit it. Unfortunately React did not natively have CSV functions, though somewhat easy to implement, would not give enough reason to do it.

Then another problem with it was there was no description. The fix for this is simple, and a note will be left. Simply, the first item in the array will be a description string. Displaying this string is a challenge on its own. Should it be a tooltip? That feels too small, and also esoteric enough that it might be a hard feature to discover for users. Another solution

could have been to display it in its own div. But that once again would take up lots of space, which is something that we were trying to avoid. This still does seem like the best approach though.

The final problem was implementing this feature. We took notes on the proposal given to us by the G2A people and implemented it by their vision.

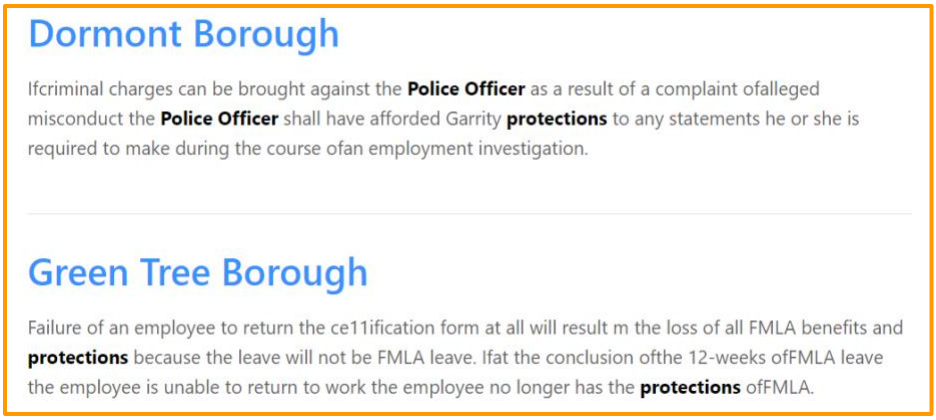


Then CASSI hired a UX designer who wanted changes that made the website much less usable to the intended audience. Equipped with two search bars, the website became much less readable. This came with a very frustrating meeting of trying to explain why this is not a good solution.

Instead we decided, with the UX designer, to make it into two dropdowns. One for the categories another for the keywords of said categories. Though more work is to be done, for its ease of use to the user. By making it more clear how these dropdowns work without large walls of text obscuring the page.

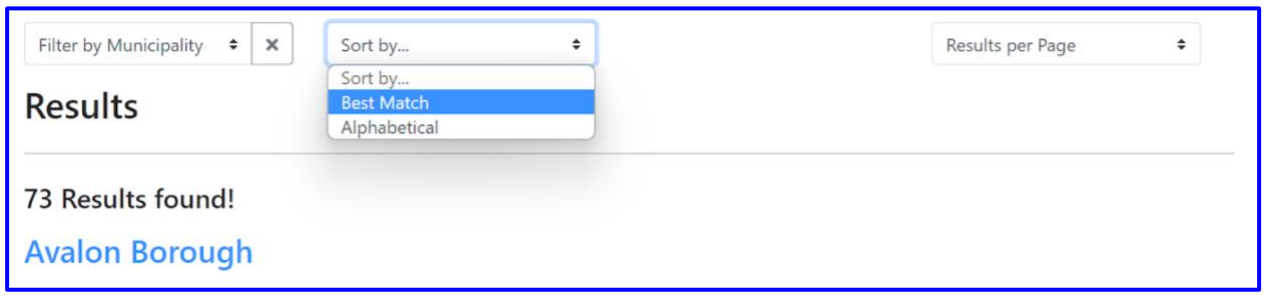
Enhance Search Bar Functionality - Stop Word Removal

A high level problem with the previous implementation was relevance of results. The goal of providing the user with the most intuitive process and relevant results is certainly multifaceted, but we identified one fix that would contribute to reaching this goal. Namely, we decided to remove stop words from the search query to provide more relevant results as well as highlight relevant keywords in the matching contracts. Stop words are terms in Natural Language Processing that can be removed from a phrase without affecting the meaning of it. Some common stop words include the, an, can, and, etc. After the user inputs their search query into the search bar, it is stripped of all stop words. The revised query is sent to the backend which generates the matching contracts. After the contracts are returned to the front end, the matching contracts highlight keywords in the contracts' sentences that align with the search query. The previous implementation highlighted stop words such as the, a, and can. Clearly, this is not helpful for the user and even creates a busy visual for the user to digest. Therefore, we also removed stop words from the highlighting on the front end to emphasize the key words most relevant to their search. The images below depict a before and after of the newly added highlighting feature for the search phrase "protections for a police officer".



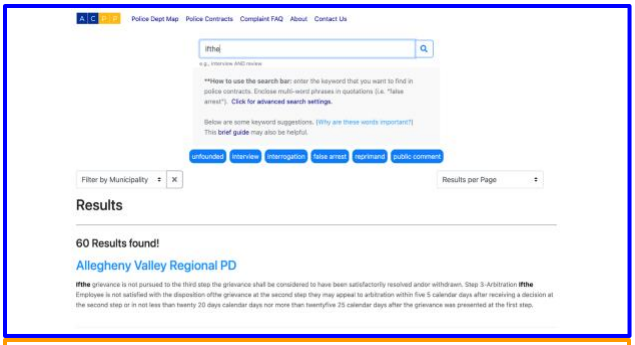
Enhance Search Bar Functionality - Sort By

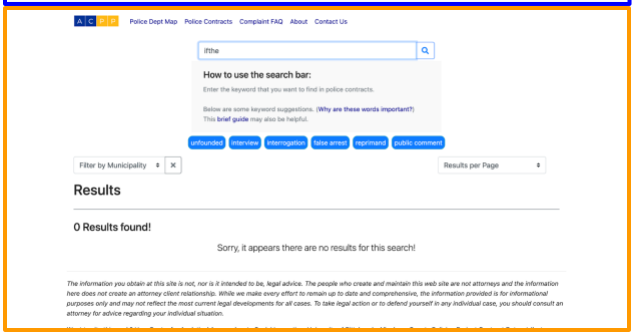
A realization our team had was that users of the ACPP are likely expecting a searching process similar to that of website they interact with on a regular basis such as Walmart, Google, or Amazon. With that, one signature feature that many search engines and e-commerce websites have that the previous implementation of the ACPP did not was a sort by option on the search results page. The previous implementation defaulted to returning the matching contracts in alphabetical order by municipality name. This may be helpful for some users, but other users would likely prefer to see the results in a sorted order based on relevancy. Therefore, we implemented a sort by front end component that allows the user to choose whether they want the matching contracts to be sorted by best match or alphabetically. The best match option is implemented in the back end by scoring each contract based on its match with the search query. A contract containing the search phrase verbatim receives a higher grade than that containing just one word. We altered the object being returned from the backend to include this scoring which is then used on the front end to rank the contracts by best match. Within each grade, the contracts are sorted by number of matching sentences. The image below depicts the added front end feature.



Contract Parsing - Space Check

The approach taken for space checking was first trying to figure out a way to do spell checking on the contract documents. In searching for different python modules that would be useful for spell checking, the 'symspellpy' module was found. Using this module, implementation of a spell checking algorithm that would fix many of the spelling and spacing errors that were a result of the PDF to text functions. There were many issues that arose with this, the major one being, many of the spelling errors in the text documents ended up not being solved by the 'symspellpy' because the edit distance (number of edits to a string it would take to create a string in the 'symspellpy' dictionary) was too great. Another major issue was that certain words were being incorrectly replaced, replaced with wrong capitalization, or replaced by a word that was not grammatically correct. Thus, there were many bugs in using the spell checking from 'symspellpy'. But, this did not mean that this module would not still be useful in helping to fix these documents. Since many of the documents had spacing errors, and this was one of the more common spelling errors in the documents, another approach was to try and get rid of as many accidentally combined words, or accidental splits in words, as possible. The 'symspellpy' module had a function that tackled exactly this issue, and when applied it worked really well. Still, it was not perfect, as it did not fix 100% of the spacing issues, so it was necessary to code in edge cases. All of this was done in python on the back-end of the website, as the new code is used during the setup of the website. The most difficult issue of this task was searching for working spell checking modules that would help keep the format of the text documents, while simultaneously fixing spacing errors. Once that was found, it was just trying to figure out how to make the module best work for the project assigned, and although it was not able to solve every issue, it was able to help reduce many of the spelling errors. One last thing with the implementation is that it slows local setup time as the function itself is looping through all the files before sending them through the website's original function. This is a design choice that can be fixed once it is decided whether to incorporate or delete the original algorithm.





Link Contract With Preview Text Upon Selection

We set a goal at the beginning of the semester to research an improvement to the pdf viewer to add an automatic scroll and highlight relevant words to the search query. The timeline, unfortunately, did not allow for research into this problem and whether a feasible solution exists. We acknowledge this is an improvement that would simplify the user experience, but we're hopeful our other enhancements helped contribute to an improved experience for users.

Development

Technologies/Frameworks

Python, React, JavaScript, Django, SQL, Figma, Docker

Communication

True to our proposal, we kept contact with our CASSI points of contact very regularly through slack. And had weekly meetings with them and professor Farnan. We also kept very regular contact with each member of our capstone team, meeting every monday at around 8pm. Sometimes through text channels, which were just as helpful. We made sure to communicate our progress and ask often if any help was needed from each other. Though rarely was it needed. However there was a point setting up where we all joined together to figure out all problems with onboarding and starting to program the website. There were a few times that communication was difficult with the CASSI team, but it was a rare occurrence appearing twice. But they were always knowledgeable, and understanding of what we said and were always prepared to help us with any problems we had faced.

We had a few meetings with people outside of CASSI. Our meeting with Brandi Fisher, a local Pittsburgh activist. It was enlightening to get her opinions on what we were doing. She was not a tech user for the most part, but the site was being created for people like her, who are trying to help the community, but may have limited tech experience. When she saw what we had done, she was very excited to actually start using the site, which was also very exciting to us!