CS 530: Developing User Interfaces

Assignment 4

Goals

In this assignment, we will develop a simple "Charity Search" web site for searching a charity database. The site will take advantage of straightforward methods of intelligent information retrieval, building on an open-source search index library (Whoosh). The site interface will allow for standard user search queries as well as search-based recommendations for related charities.

Assignment

As the launching point for implementation of our site, we will start with the Goat Pasture site developed in class and adapt it to our purposes here. Please download the code associated with this assignment and unzip it into a directory **a4**. The code provided to you for this assignment is largely taken from the **webdev4/app5** sample code, but also includes several new files that will assist in developing the Charity Search web site.

Implementation Setup

This assignment uses an open-source library called Whoosh to provide search indexing and query-based search. Whoosh is similar in functionality to the Java-based Lucene library, but is completely implemented in Python and is thus reasonably easy for us to integrate into our Flask-based sites.

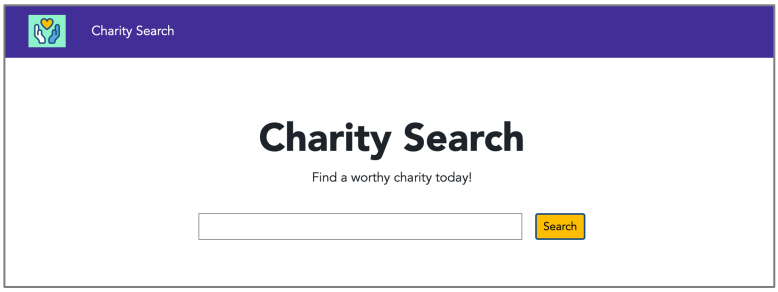
To install Whoosh, please enter

```
> pip3 install whoosh
```

(As mentioned in Assignment 0, you are welcome to use a virtual environment if you prefer, as long as the submitted assignment works as expected.)

Home Search Page

Our main search page for the site will be the site home page (at the site root "/"). When users navigate to the home page, they should see the following:



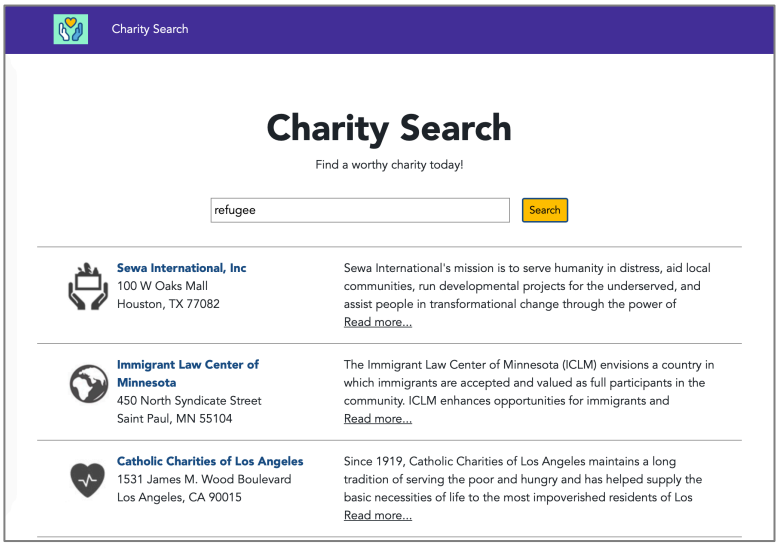
Your first task is to build out this page. We can think of the search area (everything below the menu bar) as a "widget" and develop associated JS and CSS files for this widget. Specifically, you can follow the existing **static/js/goatViewer.js** as a guide and, in a similar manner, build a Searcher component in **static/js/searcher.js** and an associated CSS file in **static/css/searcher.css**.

To mimic this figure as closely as possible, note that the charity icon at the upper-left has been provided to you (**static/img/charity.jpg**), and the menu bar contains only the "Charity Search" button which leads to the home page. The menu bar color is #452D9B, the search button background is #FFC10E, and the search button border is 2px #1A4F83.

The data for our site are in **a4/dev/orgs.json** and were obtained from Charity Navigator [https://www.charitynavigator.org], a site that organizes information and ratings for a host of local, national, and international charities. The data set provided includes only a sampling of their data, specifically, records of 100 charities from each of 11 categories (1100 charities in all).

To use these data, there is an **Index** class in **index.py** which wraps the Whoosh library code with convenient helper functions (i.e., you don't need to interface directly with Whoosh, only with the given **Index** class). We recommend modifying the **Database** class in **db.py** and creating the index there. If the index is empty, it will need to be populated with the data set: you should load the JSON data from **dev/orgs.json** and pass the data to **index.populate()** method. You can pass "index" as the path for the index; Whoosh will create files within the directory **index/** as it needs to store and index the items. The **'mission'** field in the organization records will be the primary search field.

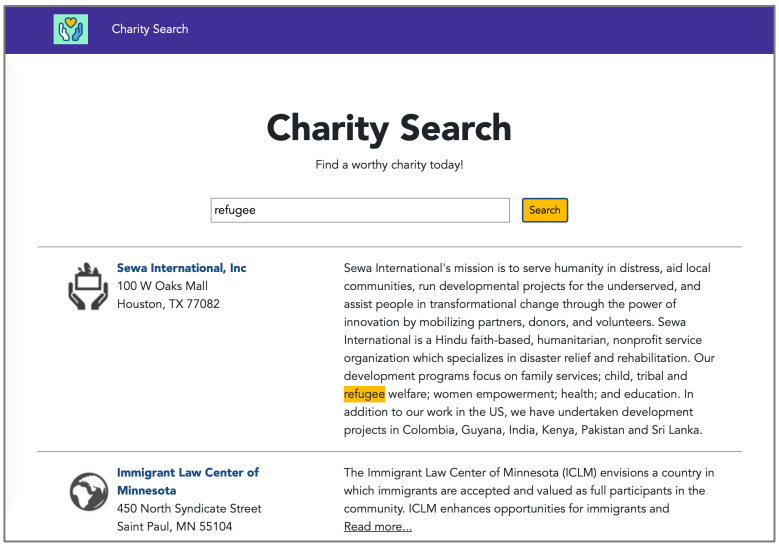
Once your **Database** class is fleshed out to interface with the **Index**, you should be able to link the **Searcher** JS widget to the back-end database via API calls. Specifically, when the user either presses "Return" in the search input box or clicks the "Search" button to the right of the input box, the Searcher should call the API, receive the incoming search results, and populate the organization items below the search input. For example, here is a screen after searching for the query "refugees":



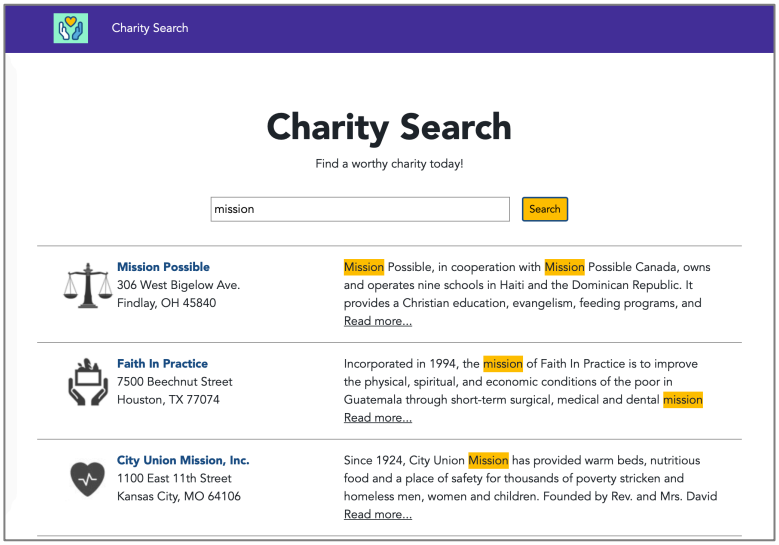
Again, you should mimic the styling here as closely as possible, which should be straightforward using Bootstrap and the data fields in the organization records. The image on the left side comes from the **categoryImage** field in the data.

Of note, the mission statement on the right side of the results should have a fixed height so that it doesn't take up too much vertical space. Below the mission statement should be a link "Read more…" which, when clicked, opens up the box to its full height. For

example, if we click “Read more…” for the first mission statement, we get the following screen:



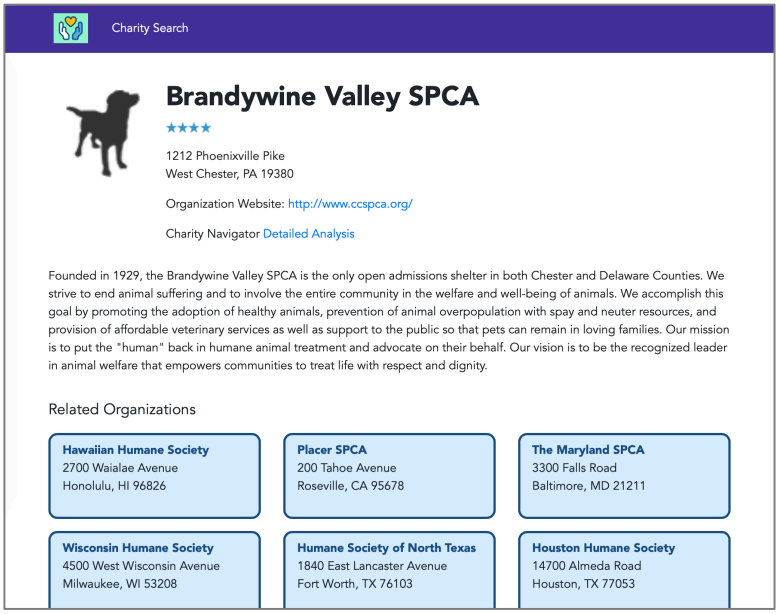
Finally, as can be seen above, we add highlighting to the search results. When the results are populated onto the page, you should take the query and break it down into individual words; then, using JavaScript regular expressions, find these words in the mission text and surround the words with a highlighting wrapper (for example, “ … ” and defining this class in your CSS file to have a yellow background). Highlighting should work regardless of lower/uppercase, as can be seen in the example below:



Finally, the title of the organization should be a clickable link that brings the user to the View page described next.

View Page

When the user views a single organization’s information, they will navigate to a page that takes, in its URL, a single parameter **uid** for the unique identifier of the organization (e.g., **/view?uid=4**). The page then shows various information about the organization, as illustrated below:



Again, you should strive to mimic the layout and styling here as closely as possible. The boxes at the bottom have a border color #1A4F83 and background color #D2EEFF. They also have a minimum height of four lines of text.

Like before, you should create a **Viewer** widget with files **static/js/viewer.js** and **static/css/viewer.css** that builds out the page. The widget should make two separate API calls: one to get the basic information for the organization (as identified by the provided **uid**), and another to find related organizations. The **uid** parameter can be gotten from the URL with this JavaScript snippet:

```
const params = (new URL(document.location)).searchParams;
const uid = params.get('uid');
```

The 6 “Related Organizations” at the bottom come from search-based suggestions. In fact, you can use the same search index that you use for standard search: take the mission text from the original organization and use this text as a search query, thus getting other organizations that have similar mission statements. Note that when doing this, the parameters of the search are slightly different: the terms need to be joined by ‘**or**’ (instead of ‘**and**’ for a user-typed query), and the search needs to avoid the organization with the same **uid** (or it will simply return the same organization as the best-matching result). Please see the search parameters in the **Index** class for more information. The organization names in the “Related Organizations” blocks should be linked to take users to the View page for that organization, allowing a user to quickly explore a number of organizations.

Documentation

It is expected that all code written for this assignment is properly commented where needed, especially in places where you have made particular choices about data structures and/or algorithms to employ. Also, please add the following identification header to every file you create:

```
# Your Name, Your Email
# CS530: DUI, Assignment [#]
```

(or whatever commenting syntax is appropriate for the file at hand).

Submission

Please submit your files as an attachment for the assignment on Blackboard Learn. Please use a compression utility to compress your files into a single ZIP file (not RAR or any other compression format). The final ZIP file must be submitted electronically using Blackboard—please do not email your assignment to a TA or instructor. If you are having difficulty with your Blackboard account, you are responsible for resolving these problems with a TA or someone from IRT before the assignment it due. If you have any doubts, complete your work early so that someone can help you.