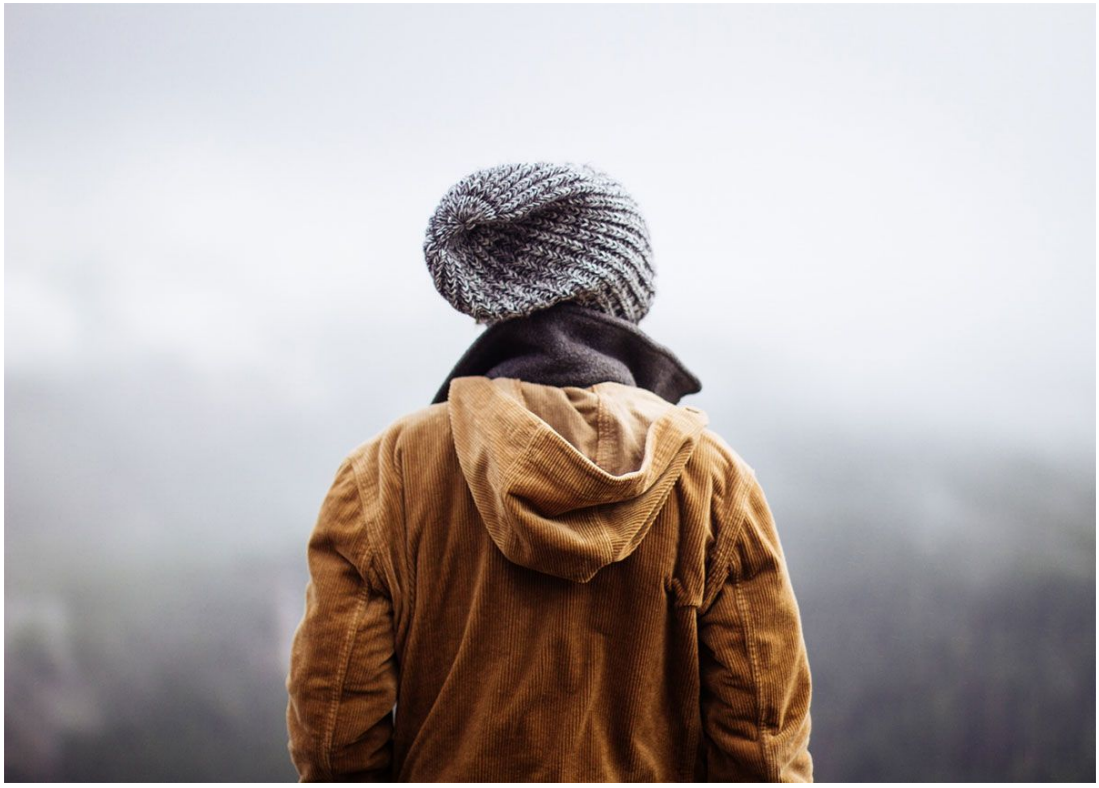
CS 373 - SOFTWARE ENGINEERING

FIGHTPOVERTY.online

RANKING CHARITIES TO FIGHT POVERTY



By: Justin Berman, Chris Amini, Charlie Matar, Yijie Tang, Kevin Singh.

Introduction

Fight Poverty is a web application that aggregates and ranks charities from all around the United States whose main purpose is to aid in the fight against poverty and provide helpful tools and resources to help those in need.

As of the most recent 2016 Census[1], there are about 40.6 million people in America living in poverty situations, or roughly about 12.7%. We still have a long way to go to improve the living conditions in this country, and this is our way of contributing.

Motivation

This project started as a way for us to provide a useful service to our community, using our software engineering and web applications skills. Upon browsing several websites, we realized how tedious it could be for someone to find a trustworthy and reliable charity, based on their interests and close to their community. That's when we became aware we could simplify the process by parsing and accumulating all these different charities into a single website, implementing a system that ranks them in order of their reliability and orders them by city and county. This would therefore simplify and make it easier for people to find a charity they trust and is close to their community.

User Stories

1. As a user, I want to be able to identify charities by the type of cause it supports.
    a. If you navigate to the Charities tab of our website, you will be able to click on any of the displayed charities and more information will pop up on the specified charity, including its cause.
    b. On the backend, we plan to have a RESTful API that will return charities given a cause. The expected format of the request and response can be seen here under the GET request titled "Charities by cause."
    c. Discussion: we have a very limited set of causes since we are focused on charities specifically aiming to fight poverty. The user might be disappointed there are very few different types of charities in our database.
    d. Estimated completion time: 6 hours after RESTful API set up (which should take 8 hours)
2. As a user, I want to be able to compare two different charities between cities and counties.
    a. Our website provides two different tabs, one for cities, another for counties, where you can access and compare the different charities in each region.
    b. On the backend, we plan to have RESTful API's that return charities in a city or county when searching for either. The expected format of the requests and responses can be seen here in all GET requests in the Cities and Counties folders. Those responses would return charity id's which can then

be used in the GET request titled “Charity by id” to get the data of the charities the user wants to compare.

   c. Discussion: a better user experience would be to have an input form/checkboxes to compare charities in different regions, rather than forcing the user to jump around the website to find the charities the user wants to compare. Need to look into a way to do this.
   d. Estimated completion time: 20 hours after RESTful API set up
3. As a donator, I want to know more information about how my money is being spent by the charity I choose to support.
   a. Any charity you select through our website provides you with all of the information on the selected charity, including what they do, how your donations help them, what their financial rating is based on how well they spend money, how they provide their service and more.
   b. On the backend, we plan to have a RESTful API that will return detailed information about a charity. All expected data can be seen in the GET requests in the Charities folder at this link.
   c. Discussion: a user might not be satisfied with the data we have, though we can easily provide links to sources that might give more information.
   d. Estimated completion time: 6 hours after RESTful API set up
4. As a user, I want to be able to find charities closest to me.
   a. Our charities are organized by location. Furthermore, clicking on any charity will give you its full address and zip code.
   b. On the backend, we plan to have a RESTful API that will return charities given a city/state, county/state, or zip code. The expected format of the request and response can be seen at this link in the GET request titled “Charities by location.”
   c. Discussion: a better user experience would be to have a search box where the user can provide their location to find charities, rather than forcing the user to jump around the website to find the charities closest to the user.
   d. Estimated completion time: 12 hours after RESTful API set up
5. As a user, I want to be able to click on a charity from the list and get information about its rank, city, county, what they do to fight poverty, and other statistics.
   a. All of this is available by default on each charity published on our website.

b. On the backend, we plan to have a RESTful API that will return detailed information about a charity. All expected data can be seen in the GET requests in the Charities folder at this link.
c. Estimated completion time: 4 hours after RESTful API set up

RESTful API

Documentation to our expected RESTful API is published via Postman here. We expect requests to be formatted as detailed in the documentation, and responses to come as JSON objects as seen in the examples in the sidebar on the right.

We used Postman to document the API's (see information on Postman in the Tools section). In order for anyone to pick up with continuing work on the RESTful API in Postman (without revealing our username and password), simply find the latest Postman collection from the GitLab repo here, and import that file into your Postman app. All of our RESTful API's would then appear inside your Postman and you can add/edit/delete API's as needed.

Right now you can access our collected datasets at api.fightpoverty.online, however, the RESTful API's documented in the link above are not currently working because our backend server is not set up yet.

Models

Charities

We expect each individual Charity record to look like this JSON object:

```
{
      "Name": "West Regional Food Bank",
      "id": 1
      "Mission Statement": "Feed the hungry",
      "Cause": "Human Services",
      "City": {
          "Name": "West City",
          "id": 10
      },
      "County": {
          "Name": "West County",
```

```
            "id": 100
        },
        "State": "West State",
        "Zip Code": 11111,
        "Accountability Rating": 9.3,
        "Financial Rating": 6.2,
        "FightPoverty Rating": 8.3
}
```

Each id will be created by the FightPoverty database when each record is created. With exception to the FightPoverty Rating and id's, all data in the provided JSON object has been scraped from the Charity Navigator API. The FightPoverty Rating will be determined via our own algorithm that takes into account the poverty level of that county, as well as Charity Navigator's provided Accountability and Financial Rating.

In order to scrape for charities, we first requested a key from the Charity Navigator website. Then we used this python module to scrape data from the API into a JSON file. See this folder to see which queries were used and the JSON files.

Cities

We expect each individual City record to look like this JSON object:

```
{
    "city": "West City",
    "id": 10,
    "counties": [
        {
            "Name": "West County",
             "id": 100
        }
    ],
    "state": "West State",
    "population": 100000,
    "charities": [
        {
            "Name": "West Regional Food Bank",
            "id": 1
```

```
        },
        {
            "Name": "West Homeless Shelter",
            "id": 2
        }
    ]
}
```

In addition to the charity data scraped via the method highlighted in the Charities section, the rest of the data provided in this JSON object has been scraped from both the US census as well as a Zip Code database.

To scrape from the US census, we used the same Python module linked above to scrape the data from the API into JSON files. These folders show exact queries used and JSON files: cities folder, counties folder, states folder, and zip code folder.

To scrape from the Zip Code database, we downloaded the CSV file from that page linked, then used this python module, to convert the downloaded CSV into this JSON file.

Counties

We expect each individual County record to look like this JSON object:

```
{
    "name": "West County",
          "id": 100,
    "state": "West State",
    "poverty percentage": 10.3
    "population": 1000000,
          "charities": [
        {
            "Name": "West Regional Food Bank",
            "id": 1
              },
              {
                  "Name": "West Homeless Shelter",
            "id": 2
              }
          ],
          "cities": [
              {
```

```
            "Name": "West City",
            "id": 10
              }
          ]
}
```

In addition to the geographic info already accounted for above, to get the poverty percentage by county, we used the following data from the US Census. We used the same Python module linked above to scrape the data from the API into a JSON file. This counties folder shows the API used and the JSON file.

Tools

Namecheap

We used Namecheap.com to register and manage our domain. Their dashboard makes it simple to manage your domain's properties, redirects, and more. Plus, all of their domains include WhoIsGuard privacy protection free of charge for the duration of your domain.

Amazon Web Services - Simple Secure Storage (S3)

Using AWS S3 made it simple to host our static website, manage our files and provide a place that dynamically and immediately adjusts to our changes. It also provides over 99.99% durability[2] and one of the best availability and scalability[2].

Gitlab

Gitlab is where we hosted our development repository. It's the perfect place for collaboration, issue tracking and most importantly, continuous integration and deployment. It helped us automate our deployment to our S3 bucket and publish our changes with every push.

Postman

Postman enables us to have an easy to use tool to test and document our website's RESTful API's. It also provides an easily accessible reference for anyone to consult when attempting to use our RESTful API's via automatically generated documentation linked here.

Bootstrap

Bootstrap is a great front-end framework that allowed us to easily design our responsive web application. It's easy to learn, you can get it up and running in minutes and it does most of the heavy lifting for you. Although you don't need to know CSS to use Bootstrap, it is recommended that you have a strong understanding of CSS and HTML beforehand to be able to make unique and elegant websites.

Google Docs

We used Google Docs to collaborate on our project proposal, requirements, technical report and anything else that required us to share information through writing.

Hosting

We used a combination of Namecheap and AWS S3 to host our website online. Namecheap's DNS made it simple for us to obtain our domain, create our API's subdomain, manage our redirects, and link our domain to our S3 bucket. AWS S3, on the other hand, allowed us to host our files, immediately publish our changes, and link our bucket to our domain to make it available online.

References

1. https://www.census.gov/library/publications/2017/demo/p60-259.html
2. https://aws.amazon.com/s3/