Food Me Once
Phase 1: Technical Report

Development Team:

- Christoper Chasteen, *Full Stack Developer*
- Gyuwon Kim, *Full Stack Developer*
- Shubhendra Trivedi, *Full Stack Developer*
- Brian Dyck, *Full Stack Developer*
- Nithin Pingilli, *Full Stack Developer*

Purpose

Food Me Once is being developed to provide a platform for users to gather information on the food security of communities across the United States. It combines disparate data sources about food security across districts/counties, political representation and legislation to present a well-rounded perspective. It will enable users to understand how political representation affects health outcomes as well as what actions have been undertaken to ensure equitable access to healthy food and eradicating food deserts. The website will generate statistics/visualizations across various dimensions (population, representation, race, etc.).

Motivations

The motivation was very clear. Every city/state has an extreme contrast to health outcomes per the zip code and socio-economic status. This is visibly apparent driving through different parts of a city/state, an intuition well supported by academic research.
Initial research conducted by the development team provided insight into vast regions of the United States where regular and easy access to what is categorized as healthy food was not prevalent. "Food deserts" being the term to describe regions where it becomes hard (w.r.t distance and affordability) to gain access to healthy food.
This inspired the genesis of the idea to uncover (if any) trends between food security and political representation as well as what legislative action has been taken. As software

engineers, the team decided to leverage their technical skills to provide a spotlight on this issue.

User Stories

Our customer was the Put It In Park team. We provided 5 user stories for their phase 1 development cycle.

1. It would be useful to see the description of the project along with the aim. Potentially, the details of each model and how they correlate. Additionally, a cool background (not necessary) might be a nice-to-have. Overall, the website can be static for now - it will need to be dynamic in the future phases.
2. A profile picture for each group member. Along with that, a total summary of the entire repository commits + issues would be cool to see. It would not be necessary to make the page look "pretty" for now - just a simple picture + commits & issues.
3. It would be preferred if the model pages had a grid or table with links to the instance pages. Each instance listed on the model page should have 5 attributes. We must be able to click on a link/somewhere to navigate to the instance page. To be clear, if the model does not lend itself to a variety of media (for eg: outdoor activities) - then a table is probably more idea. However, a model like National Parks might have lots of pictures/videos specific to each park, and hence a good candidate for a grid. Regardless, a table is fine for now too.
4. There should be a link to the API documentation - designed by Postman. It would be ideal to have 'GETs' for models + instances.
    a. return a list of models
    b. return the details of the instances (attributes)
    c. return a detailed list of models with their 5 basic attributes. No PUTs/POSTs.
5. Provide 3 instance pages for each of the 3 models. Each instance page must display the original 5 attributes from the model page + additional data points. Along with that, there must be some sort of media (more than one preferred) on each instance page, that is specific to the instance: links to videos, images, etc. It does not need to be "pretty" or formatted super well.

We were assigned a customer team as well, found at: After Work.
They also provided 5 user stories for our 1st phase of development. Two of these have been closed, while the remaining 5 are out of the scope of this push and has been communicated as such.

1. Have the home page explain the name of the website a bit more, since right now it's a bit confusing and it sounds like its a mismatch between the name and content.
    a. Estimated Weight: 1.
    b. Implemented such that the splash page explains in brief the motivation of the site.
2. Switch the layout of legislation to a table view since it doesn't really make sense to have a picture with every new legislation.
    a. Estimated Weight: 2.
    b. Changed from grid to lists
3. Add a petitions page where we can see what current petitions are going on about certain legislation, maybe requires another API.
    a. This was out of the scope for this phase
4. I would like to see information about obesity in these districts.
    a. This was out of the scope for this phase
5. Can I see the cost of living, income inequality, and homelessness stats on the site?
    a. This was out of the scope for this phase

RESTful API

For phase 1, we have simply designed what calls to our API might look like in the future, although the website currently operates without a RESTful API. We used Postman to design and document the API calls. The process was fairly intricate, as most of us were attempting to understand implement this for the first time.
In future phases, we will scrape data from at least three distinct databases to populate a cloud instance of Postgres DB of our own (on AWS). Our first major data source will be a health security map of the United States, allowing us to obtain information about the demographics of certain locations, as well as the health and food security of those regions. Secondly, we will scrape data to obtain information about all the political leaders in both the House of Representatives and Congress. Lastly, we will scrape data from Congressional databases regarding food, health and food security acts/legislations in the United States. We believe that by combining these three distinct sources we can use this information to create a website that allows users to truly make connections between food security and

political representation and actions. The API's we currently plan to scrape from can be viewed at the bottom of this report.

Models

For this phase of the project, we designed three model pages. A model, in this case, is the highest level abstraction of our data sources - District, Representative, Legislation.
Each model page has a table with three instances and 5 pertinent data items. We implemented a dummy filter and search feature for future phases, which does not yet work.
Each instance page (linked from their respective model page) has the original 5 attributes, along with further detail and media (social media, Wikipedia, maps, pictures). They also link to the other 2 models (specifically instances of the other 2 models).
A neat feature we implemented on the district instance page is a dynamic map based on the district selected. We decided to make our website dynamic for this phase, which was not required. However, we determined that this would make our development in the future phases much smoother. This meant utilizing the React framework instead of creating static web pages using plain HTML and CSS. Our instance pages dynamically pass the values of the selection from the model page and pull back the relevant data items and media.
The relation between each instance is as follows:

- A District is linked to the elected Representative, as well as Legislation that the Representative voted/sponsored on
- A Representative is linked to the District they were elected in, as well as Legislation(s) they sponsored/voted on
- A Legislation is sponsored by a Representative(s), and linked to the impacted District(s).

Tools

Front-end framework: React Javascript with HTML and Bootstrap CSS
Hosting: The website is hosted on an Amazon S3 instance.
Domain: We acquired the pretty URL from Namecheap
IDEs: Pycharm IDE, VSCode

Hosting

Food Me Once (https://foodmeonce.me) is a website hosted on an AWS S3 bucket. All of the files in S3 bucket are the files generated by invoking the "npm run build" command which is then pushed via "npm run deploy". We had to configure IAM profiles, to enable automatic deployment to the bucket without having to manually upload the files.
The AWS S3 instance is then connected to Cloudfront to provision the secure https access.
A brief issue we encountered was that the S3 instance, continued to serve up a cached (and hence outdated) version of our site, despite newer builds being deployed. We had to edit the Cloudfront settings, to update each time we pushed.
Lastly, we configured Route 53 for our domain name which we got using Namecheap. This allowed for our website to be accessible at foodmeonce.me.

Relevant Links

- FoodMeOnce
- FoodMeOnce GitLab Repository
- FoodMeOnce Postman documentation

- Data Sources (to be scraped programmatically via RESTful API calls)
    - Food Access Atlas API
- Socioeconomic Estimate API
- Congressional Data API

The FoodMeOnce team can be contacted at FoodMeOnce2019@gmail.com