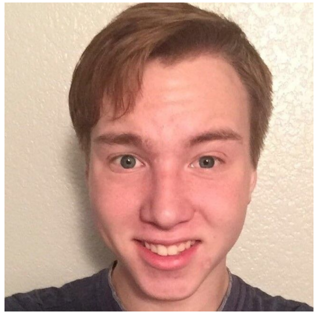
FoodMeOnce

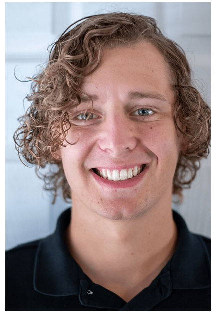
CS 373 Fall 2019 Group 9

Meet the team!


Chris Chasteen

Gyuwon Kim

Shub Trivedi


Brian Dyck

Nithin Pingilli

About the Site

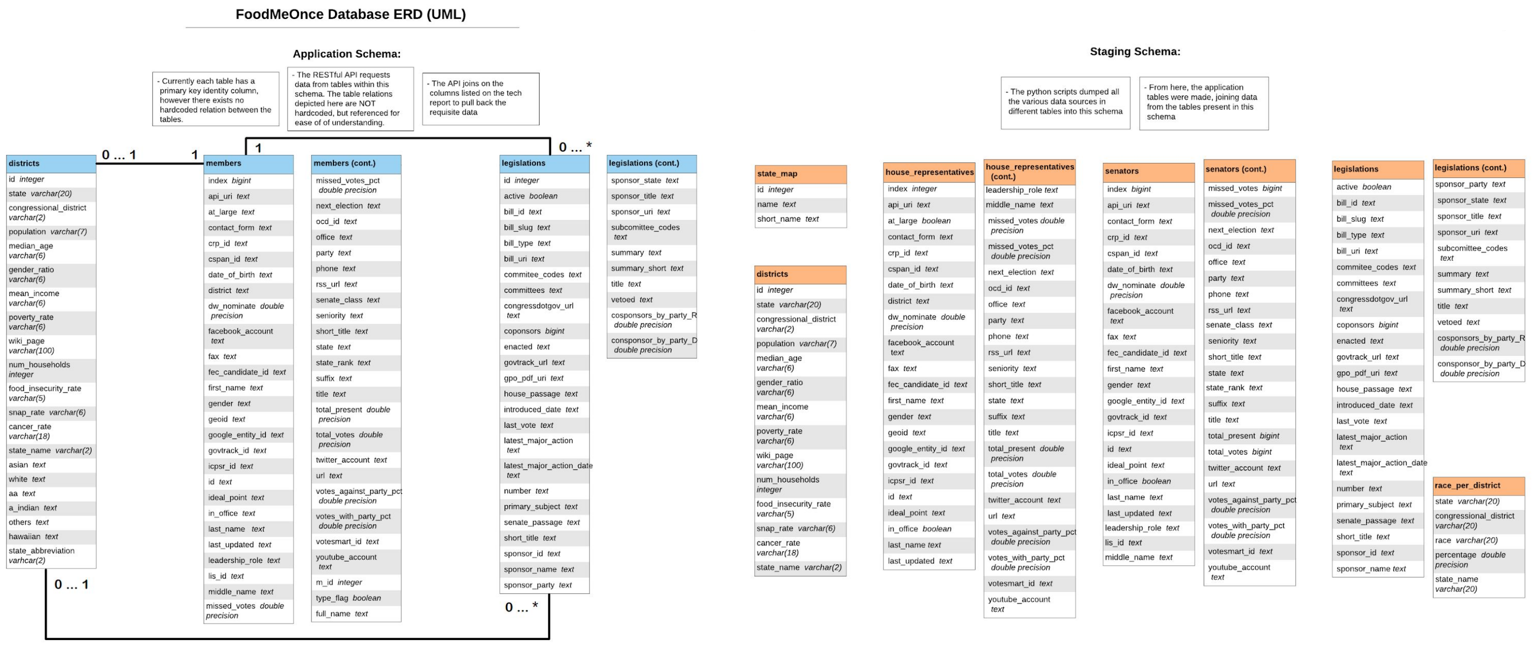
- **4 sprints**
  - *Phase 1*: Basic Static Website
  - *Phase 2*
    - Dynamic React Web Application
    - RESTful API
    - PostGres DB
  - *Phase 3*: Sorting, Searching, Filtering
  - *Phase 4*: Visualizations

- **Models**
  - *Districts*
  - *Representatives*
  - *Legislation*
- **Links**
  - FoodMeOnce GitLab Repository
  - FoodMeOnce API Documentation

What is FoodMeOnce?

- ❖ Web Platform to allow users to gather information on food security throughout US Congressional Districts
- ❖ By combining Disparate Data sources we provide a well rounded perspective on food security in relation to political representation and legislation
- ❖ Generates Easy to understand Visualizations to allow a quick grasp of Congressional food support using various dimensions such as population, race, and representation.

Database



FoodMeOnce Database ERD (UML)

Application Schema:

- Currently each table has a primary key identity column, however there exists no hardcoded relation between the tables.

- The RESTful API requests data from tables within this schema. The table relations depicted here are NOT hardcoded, but referenced for ease of of understanding.

- The API joins on the columns listed on the tech report to pull back the requisite data

districts
id *integer*
state *varchar(20)*
congressional_district *varchar(2)*
population *varchar(7)*
median_age *varchar(6)*
gender_ratio *varchar(6)*
mean_income *varchar(6)*
poverty_rate *varchar(6)*
wiki_page *varchar(100)*
num_households *integer*
food_insecurity_rate *varchar(5)*
snap_rate *varchar(6)*
cancer_rate *varchar(18)*
state_name *varchar(2)*
asian *text*
white *text*
aa *text*
a_indian *text*
others *text*
hawaiian *text*
state_abbreviation *varhcar(2)*

members
index *bigint*
api_uri *text*
at_large *text*
contact_form *text*
crp_id *text*
cspan_id *text*
date_of_birth *text*
district *text*
dw_nominate *double precision*
facebook_account *text*
fax *text*
fec_candidate_id *text*
first_name *text*
gender *text*
geoid *text*
google_entity_id *text*
govtrack_id *text*
icpsr_id *text*
id *text*
ideal_point *text*
in_office *text*
last_name *text*
last_updated *text*
leadership_role *text*
lis_id *text*
middle_name *text*
missed_votes *double precision*

members (cont.)
missed_votes_pct *double precision*
next_election *text*
ocd_id *text*
office *text*
party *text*
phone *text*
rss_url *text*
senate_class *text*
seniority *text*
short_title *text*
state *text*
state_rank *text*
suffix *text*
title *text*
total_present *double precision*
total_votes *double precision*
twitter_account *text*
url *text*
votes_against_party_pct *double precision*
votes_with_party_pct *double precision*
votesmart_id *text*
youtube_account *text*
m_id *integer*
type_flag *boolean*
full_name *text*

legislations
id *integer*
active *boolean*
bill_id *text*
bill_slug *text*
bill_type *text*
bill_uri *text*
commitee_codes *text*
committees *text*
congressdotgov_url *text*
coponsors *bigint*
enacted *text*
govtrack_url *text*
gpo_pdf_uri *text*
house_passage *text*
introduced_date *text*
last_vote *text*
latest_major_action *text*
latest_major_action_date *text*
number *text*
primary_subject *text*
senate_passage *text*
short_title *text*
sponsor_id *text*
sponsor_name *text*
sponsor_party *text*

legislations (cont.)
sponsor_state *text*
sponsor_title *text*
sponsor_uri *text*
subcomittee_codes *text*
summary *text*
summary_short *text*
title *text*
vetoed *text*
cosponsors_by_party_R *double precision*
consponsor_by_party_D *double precision*

Staging Schema:

- The python scripts dumped all the various data sources in different tables into this schema

- From here, the application tables were made, joining data from the tables present in this schema

state_map
id *integer*
name *text*
short_name *text*

districts
id *integer*
state *varchar(20)*
congressional_district *varchar(2)*
population *varchar(7)*
median_age *varchar(6)*
gender_ratio *varchar(6)*
mean_income *varchar(6)*
poverty_rate *varchar(6)*
wiki_page *varchar(100)*
num_households *integer*
food_insecurity_rate *varchar(5)*
snap_rate *varchar(6)*
cancer_rate *varchar(18)*
state_name *varchar(2)*

house_representatives
index *integer*
api_uri *text*
at_large *boolean*
contact_form *text*
crp_id *text*
cspan_id *text*
date_of_birth *text*
district *text*
dw_nominate *double precision*
facebook_account *text*
fax *text*
fec_candidate_id *text*
first_name *text*
gender *text*
geoid *text*
google_entity_id *text*
govtrack_id *text*
icpsr_id *text*
id *text*
ideal_point *text*
in_office *boolean*
last_name *text*
last_updated *text*

house_representatives (cont.)
leadership_role *text*
middle_name *text*
missed_votes *double precision*
missed_votes_pct *double precision*
next_election *text*
ocd_id *text*
office *text*
party *text*
phone *text*
rss_url *text*
seniority *text*
short_title *text*
state *text*
suffix *text*
title *text*
total_present *double precision*
total_votes *double precision*
twitter_account *text*
url *text*
votes_against_party_pct *double precision*
votes_with_party_pct *double precision*
votesmart_id *text*
youtube_account *text*

senators
index *bigint*
api_uri *text*
contact_form *text*
crp_id *text*
cspan_id *text*
date_of_birth *text*
dw_nominate *double precision*
facebook_account *text*
fax *text*
fec_candidate_id *text*
first_name *text*
gender *text*
google_entity_id *text*
govtrack_id *text*
icpsr_id *text*
id *text*
ideal_point *text*
in_office *boolean*
last_name *text*
last_updated *text*
leadership_role *text*
lis_id *text*
middle_name *text*

senators (cont.)
missed_votes *bigint*
missed_votes_pct *double precision*
next_election *text*
ocd_id *text*
office *text*
party *text*
phone *text*
rss_url *text*
senate_class *text*
seniority *text*
short_title *text*
state *text*
state_rank *text*
suffix *text*
title *text*
total_present *bigint*
total_votes *bigint*
twitter_account *text*
url *text*
votes_against_party_pct *double precision*
votes_with_party_pct *double precision*
votesmart_id *text*
youtube_account *text*

legislations
active *boolean*
bill_id *text*
bill_slug *text*
bill_type *text*
bill_uri *text*
commitee_codes *text*
committees *text*
congressdotgov_url *text*
coponsors *bigint*
enacted *text*
govtrack_url *text*
gpo_pdf_uri *text*
house_passage *text*
introduced_date *text*
last_vote *text*
latest_major_action *text*
latest_major_action_date *text*
number *text*
primary_subject *text*
senate_passage *text*
short_title *text*
sponsor_id *text*
sponsor_name *text*

legislations (cont.)
sponsor_party *text*
sponsor_state *text*
sponsor_title *text*
sponsor_uri *text*
subcomittee_codes *text*
summary *text*
summary_short *text*
title *text*
vetoed *text*
cosponsors_by_party_R *double precision*
consponsor_by_party_D *double precision*

race_per_district
state *varchar(20)*
congressional_district *varchar(20)*
race *varchar(20)*
percentage *double precision*
state_name *varchar(20)*

Tool stack

Front-end framework

- React Javascript
- Bootstrap and CSS
- Selenium
- Mocha
- D3

Back-End Tools

- PostgreSQL
- POSTMAN
- SQLAlchemy
- Flask
- Python

IDEs

- Pycharm IDE
- VSCode

Backend

- Amazon S3

Domain

- NameCheap
- Route53

Others

- Gitlab
- LucidChart
- Docker
- Gitlab CI/CD

Demonstration

What did we do well?

- ❖ Made website dynamic in phase 1
- ❖ Searching, Sorting, Filtering
- ❖ Getting each phase reviewed by the TAs
- ❖ Well paced throughout each phase of the project.
- ❖ Gitlab Issue Board
- ❖ Communication with our Customer team

What did we learn?

- E2E website building
  - Deploying and hosting in AWS
  - Designing an API
  - Domains and subdomains
  - Database design
- React and Flask frameworks
- Postman API documentation
- Importance of code readability

What can we do better?

- UI/UX
  - Relative spacing of elements on screen
  - Did not consider mobile usage
  - Prettier visuals
  - Splash page
- Code refactoring/maintainability
- Could have used flask-restless
- District model page - dynamic map load time

What puzzled us?

- How to get started
- Asynchronous API requests
- Searching Data
- D3
- Gitlab CI setup

Developer Team - PutItInPark

What did they do well?

- UI/UX is great
  - On mouse hovering over cards
  - Overall look and feel
  - Buttons
  - Visualizations located on model pages
  - Pagination has 5 pages
- Communicating on user stories was very thorough
- Made site mobile friendly

What did we learn from their website?

Technical gains:

- ❖ Hovering effect over cards
- ❖ Using entire space for the instance page
- ❖ Use of icons(search icon, loading icon)

Website specific:

- ❖ Central USA has the most frequency of parks + recreational activities

What can they do better?

- Meeting phase requirements
  - API subdomain
- Model page search results sometimes do not contain search term
  - National Parks search

What puzzles us?

Technical puzzles:

- How the model specific search works?
  - Possibly not correctly matching (ex) acadia in National Parks and tenn in States
  - No highlighting search terms on model specific search
- Website down at one point

Website specific:

- Why does Central USA have such low visitor count despite having the most parks + recreational activities?

Questions?

Thank you!