LA Homeless Helper Technical Report

Purpose

The purpose of this project is to create a database of resources that can help the homeless population in the city of Los Angeles.

User Stories

We received stories from our users and answered them accordingly.

Documentation

https://www.postman.com/pmarathi/workspace/cs373-idb-la-homeless-helper/documentation/28474521-c2680f7e-715c-41eb-9829-8d39392a2939

Models

Cities in LA County

- Unsheltered population
- Sheltered population
- Total homeless population
- Square miles of city
- Density of total homeless population

Shelters and Services

- Name
- City
- Hours
- Zip code
- Phone Number

Medicare and Medicaid Offices

- Name
- Address
- Hours
- Phone number
- URL for their website

Frontend

How the Architecture Works

- Every page has a folder in the components folder, and they are routed from the main page, App.tsx. Images are hosted in the assets folder.
- We used typescript, bootstrap, and CSS
- The backend data is not connected to the main frontend yet, so it is hard coded into the instances

Challenges

- It was a learning curve to figure out how to do everything, as neither of us are experienced in Frontend development.
- Before we figured out bootstrap, we had a hard time with the framework for instances

Hosting

- AWS Amplify with Gitlab, domain hosted by AWS

Backend

How the Architecture Works

- We have two parts of the architecture right now: main.py and query_APIs.py
- Toolchains: this part was written in Python using Flask

main.py

- This is the main Flask app that contains the functions that need to be called in order to query our API
- Each model route/function currently calls the respective function in queryAPIs which handles the logic to actually request the information. We currently render a dummy page that lists the names of each instance just for testing purposes.
  - Eventually, we will want this to query our database instead of querying the rest API.
- Each specific instance gets the necessary information by querying the whole model and looping through to find the needed information. Then, it renders a dummy page for testing purposes. Commented out is a jsonify return statement that will return a

dictionary of all information about the instance. This jsonify return statement will be used in our API.
    - Once we create our database, we will just want to query the database for the specific instance instead of looping through all of them.
- For more details on what each function does check out our API:
    - https://www.postman.com/pmarathi/workspace/cs373-idb-la-homeless-helper/documentation/28474521-c2680f7e-715c-41eb-9829-8d39392a2939

queryAPIs.py

- This method handles the actual queries to the individual APIs
- query_API takes in a string related to what database is needed(“shelters”, “cities”, or “medicare”) and retrieves the information before returning it in the form of a list of dictionaries. Each dictionary contains all information about that instance.
- query_gitlab queries the gitlab API to get information about commits and issues. This method returns a dictionary mapping string names to a dictionary with keys “commits” and “issues” which map to an integer representing the number of commits made and issues closed respectively. We use two helper functions and then essentially zip those dictionaries together.
    - query_commits is a helper function that returns a dictionary mapping names to their number of commits.
    - query_issues is a helper function that returns a dictionary mapping names to their number of issues closed.

Challenges

- This part of the project was fairly straightforward, so there were few challenges besides the GitLab API. There was a pagination problem which for issues we have monkeypatched, but we solved for commits. Unfortunately, we cannot use the same method for issues as we did for commits, and we will likely have to rewrite query_issues() later.