Maintenance and Help Guide

This document is serves as guidance for future developers or project teams that continue work on the Child Impact Statements Tool.

How to install the system

We have an existing GitHub Oranginzation, COMP-4882-CIS, in which we have two repositories CIS and CIS-FRONTEND. CIS is the backend and CIS-FRONTEND is the frontend for the project.

The first step for setup would be to make duplicates of the repositories for yourself. Once those are set up, clone the repositories locally.

Commands
Files
Code
Comment

Backend

1. Ensure the npm dependencies are installed using `npm i.`
2. Create a .env file based on the template:

```
CENSUS_API_KEY=12312312312312312312
NODE_ENV=development
```

- Also note that the Census API key (CENSUS_API_KEY) is optional here. If you do not have a key or you wish to run the backend without a key, just omit this line in your .env file. Additionally, to use a Redis instance for caching, please set the REDIS_URL environment variable, either in your .env file or passing it in the execution context.

3. Start the backend by running either `npm run start` or `npm run start:dev`

Building

To build the server, please run `npm run build`. The resulting files will be available in `dist/`.

Additionally, If you plan to build and run this application for local development, please run `npm run start:dev`, otherwise you will have to manually restart the server each time you make a change.

Frontend

1. Ensure the backend is currently running.
2. Install the npm dependencies: npm i.
3. Run the application: npm run start or npm run watch
4. One can set the backend url for development and production in the files src/app/environment.ts and src/app/environment.prod.ts respectively:

```
// Development (src/app/environment.ts)
export const environment = {
  production: false,
  apiURL: 'http://localhost:3000'
};

// Production (src/app/environment.prod.ts)
export const environment = {
  production: true,
  apiURL: 'https://cis-backend-comp-4882.herokuapp.com' //URL to where project is deployed
};
```

Building

To build the project, please run npm run build.
The resulting files will be available in dist/CIS.

Once the backend and frontend are both running, you will receive a link on the frontend terminal to the localhost where the you can find the project. Example:
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:----/ **

How the system works

Overview of project

Here is a video on how to use the CIS tool: Child-Impact-Statements Application User Guide

The bulk of the project is just adding additional datasets. Here is an example of adding new datapoints to the map below:

Note: There is plenty of data already added to the project. The process for adding new data is essentially copying the same methods and variable formats as the other data.

Uploading new data - Backend

1. Convert dataset to a .json file and add it here: src/geodata/data
2. In src/geodata/controllers/landmarks.controller.ts,
   create a variable for the data and add it to LandmarksSummaryResponse

   ```
   const [name] = this.landmarksService.get[Name](zipCode);
   ```

   then write a get request to pull the data from the json file

   ```
   @Get('[filename]')
   @ApiQuery({ type: String, required: false, name: 'zipCode' })
   get[Name](@Query('zipCode') zipCode?: string): LandmarksResponse {
     const [name] = this.landmarksService.get[Name](zipCode);

     return new LandmarksResponse([name]);
   }
   ```

3. Add your data to src/geodata/responses/landmarks-summary.response.ts in the same format as the other datasets. Be sure to create your total variable as well.
4. In src/geodata/services/landmarks.service.ts add the data to the FeatureCollection, create a constructor, and create a conversion method for your data. You can view the format of previously added data an follow that format.
5. Add your total variable to src/export/requests/tract-breakdown.request.ts and src/export/requests/zip-breakdown.request.ts in the format

   ```
   @ApiModelProperty({ required: true })
   CACount: number;
   ```

6. Optionally, in src/export/schemas/tract-breakdown.schema.ts and src/export/schemas/zip-breakdown.schema.ts add a column and type for statistic to be show on the sidebar.
7. In src/geodata/types/landmarks/landmark-type.enum.ts add a LandmarkType for your data.

Uploading new data - Frontend

1. Convert dataset to a .geojson file and add it here: src/assets/data
2. In src/app/backend/responses/landmark/landmark-summary.response.ts add your data variable to LandmarkSummaryResponse and create a total variable.
3. Add your count variable to src/app/backend/requests/export/zip-tract.base-request.ts
4. In src/app/backend/services/export.service.ts add your data variable to both: exportTractData and exportZIPData
5. Add your data variable to src/app/backend/types/geo/features/point/point-feature-type.enum.ts
6. Create a .ts file for you data's feature types. The file should be in the following format:

```
import {PointFeature} from "../point-feature.type";
import {PointFeatureType} from "../point-feature-type.enum";

export class DataFeature implements PointFeature {

 displayName: string;
 zipCode: string;
 streetAddress: string;
 date: string;
 type: PointFeatureType = PointFeatureType.CBR;

 /**
  * @param rawJSON
  */
 constructor(rawJSON: {[key: string]: string}) {
// rawJSON[--] means what is variable name in your dataset. For example in the crime datasets, the offense date for the crime is found under column 'Offense Date'
   this.displayName = rawJSON['Category'];
   this.streetAddress = rawJSON['Block Address'];
   this.zipCode = rawJSON['zipcode'];
   this.date = rawJSON['Offense Date'];
 }

}
```

   Use src/app/backend/types/geo/features/point/types/cbr-feature.type.ts as an example as well.
7. Add your feature point .ts file to src/app/backend/types/geo/features/point/types/index.ts
8. Add you data variable to src/app/components/breakdown-summary/breakdown-summary.component.ts.
9. Create a summary folder for your data. Refer to src/app/components/crime-breakdown-summary/. You need to create an HTML file,

similar to crime-breakdown-summary.component.html to show how you want the data on the sidebar displayed. You need to create an scss file, similar to crime-breakdown-summary.component.scss to show how you want the data on the sidebar style (keep it the same as the others for consistency). Finally, a .ts file is needed, refer to crime-breakdown-summary.component.ts.

10. Add your data variable to src/app/components/map/map.component.ts. There are multiple points where it needs to be added. This file sets how the markers will display on the map. You can also add a filter by adding to var baseTree
11. Add you data point feature type to src/app/components/map-box/map-box.component.ts
12. Create an isData MapSidebarData method and add it to src/app/components/map-sidebar/map-sidebar.component.ts. It will look like this:

```
isData(data: MapSidebarData) {
  return data.pointFeatureData!.type === PointFeatureType.Data ;
}
```

13. In src/app/components/point-feature-summary/point-feature-summary.component.ts, add you data variable and if it has any special or unique attribute to display in its sidebar, create a hasThis get method to add it. Example: Crime data has a unique attribute: Offense Date so hasDate was created

```
get hasDate(): boolean {
  return this.pointFeature instanceof CAFeature || this.pointFeature instanceof
CBRFeature ||
  this.pointFeature instanceof CDFeature || this.pointFeature instanceof COFeature ||
  this.pointFeature instanceof CTFeature || this.pointFeature instanceof CWFeature &&
  !!(this.pointFeature as CAFeature).date;
}
```

14. Add the data variables to src/app/helpers/sidebar-data.helper.ts the are plenty of examples in the file.
15. Create a svg icon for the data point markers on the map. Add it to src/assets/icons.
16. In src/app/helpers/feature.helper.ts, you will need to add your data variables here. iconUrl should be to the svg you make for the markers.

You should now be able to see the data points on the map.

Next you can find additional information about the previously added data.

Civic Assests (Parks, Libraries, Schools)

Note: This data will likely not need to be updated for a while.

The majority of the data/locations on the CIS map is from The Memphis Data Hub, there are still many datasets that could prove useful for the project. Alll but the crime data should be downloaded as json (backend src/geodata/data) and geojson (frontend src/assets/data). Other datasets like the childcare locations are easily added in the same format.

Crime (Public Safety)

In the Memphis Data Hub, the crime data is found here: Memphis Police Department: Public Safety Incidents.

1. Download the data as an excel file. There will be a lot of entries so we only grab the data for the last month.
2. After cutting the data down, you will need to use a reverse geocoder due to the zipcode not being included. Unfortunately, the free version: Geoapify only processes 500 rows at a time, so it could take several iteration to complete. Could another online and pay a few dollars to process the entire file.
3. Once the file is done you will need to separate the zip code from the address in excel and then separate the crime in to categories, I used 6: Assualts, Burglary/Robbery, Drugs Related, Theft, Traffic/Other, Weapon Related. You can break them up however you see fit.
4. You will then need to separate them into their own excel files and convert each into json (backend src/geodata/data) and geojson (frontend src/assets/data) files.
5. Then you can upload them to the project and then follow the above adding new data steps.

Census

Note: This likely will not need to be uploaded for a while since the American Community Survey comes out about once every 5 years.

The Census data for Shelby County can be found here Shelby County, Tennessee Profile. The Census API is used to grab the data from the tables. You can find need to update the other files in /census to get the data added. It is straightforward to add, the easiest way we found was to search in the files for variables names of the other datasets and just copied the format of those for all files.

Known Issues

1. Statistics on crime, health, and public locations (parks, etc.) are displayed when clicking on a tract even though those datasets to not have a census tract attribute. So the data always displays as 0 or empty. There is currently no fix for it since the sidebar components for tracts are connected to the zipcodes.
2. Technically not an issue but it was pointed out by the instructor that when selecting a location using the search bar, you will be navigated to that location but the popup will not automatically open. The map is created using Leaflet, so their documentation is the best bet for fixing/implementing that.