**ABOUT**

* Geo Data Dash (GD2) is an intelligent mapping application with an easy to use dashboard. It can be applied to any domain area that leverages spatial data. In this implementation GD2 plugs into a 210,000 record dataset that I created using information derived from the Polaris report on human trafficking ([read it here](https://polarisproject.org/sites/default/files/Polaris-Typology-of-Modern-Slavery.pdf)). The goal of GD2 is to help organizations make data-driven decisions using spatial data.
* A survey of 10 questions was defined then randomly applied to randomly generated coordinate points to produce a country wide dataset the app interacts with. It uses location and a scoring function to alert the user to areas where human trafficking may be occurring. A future release will include AWS Comprehend to improve the accuracy of the alert.
* GD2 was written and functions entirely on front end code (HTML5 and JavaScript). For the data it pulls geojson from several files in the application code.

**ACKNOWLEDGEMENTS**

* This app wouldn't be possible to build in such a short time without the amazing open source community. Going into this project knowing that I'd be releasing this app as open source allowed me to focus on the creative process without worrying about how to protect or monetize my code. It liberated me from the old way of doing things and I believe I built a better product because of the open mindset. I've been inspired to make contributions to projects where I can and within my capabilities.

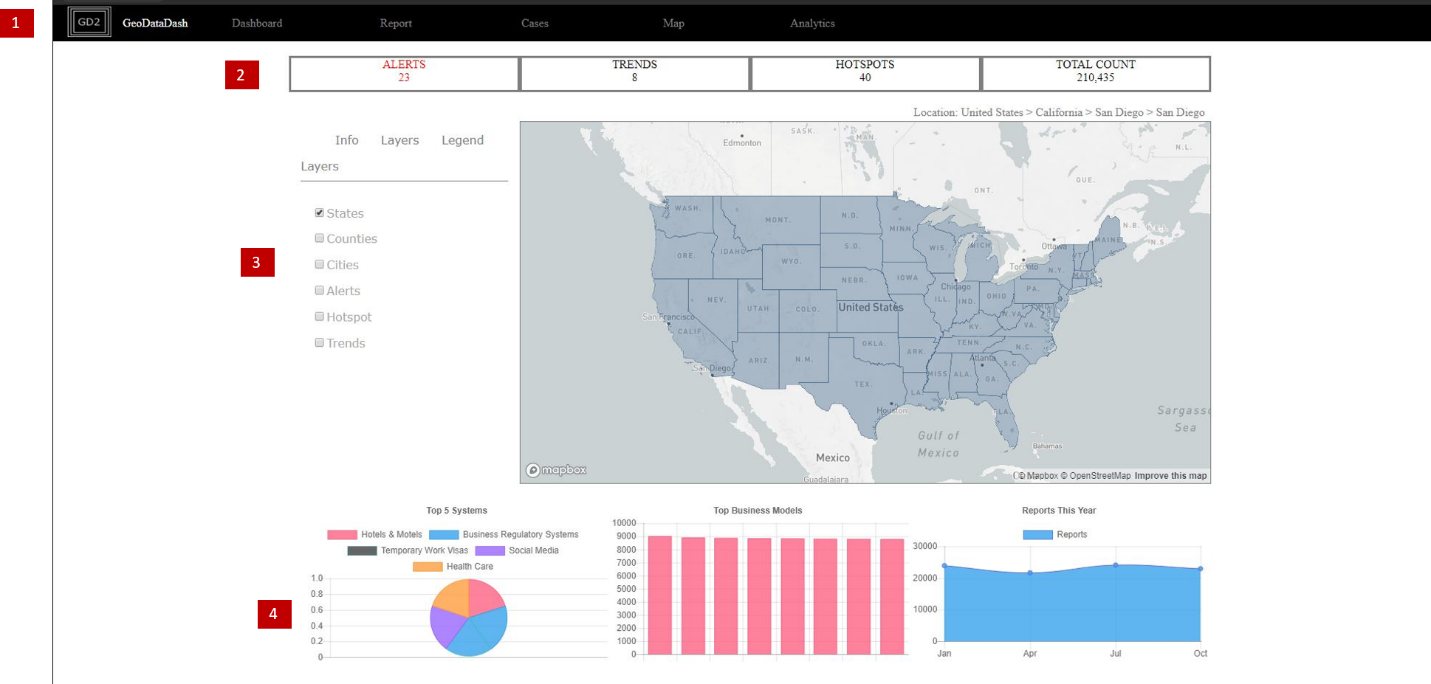
Here's the open source tools I used:

* + Mapbox
    - Open source mapping platform
  + Turf.js
    - Open source geospatial analytic library maintained by Mapbox
  + Chart.js
    - Chart library

**APPLICATION**

The application has two main sections. A dashboard where data is displayed and interacted with and a report tab where new data I added.

* Dashboard



1. Nav Bar

2. Highlights

3. Map section

4. Key Metrics

1. Nav Bar

Navigate through the application. Currently only the 'Dashboard' and 'Report' tab work. In the future functionality will be developed on the other tabs.

* Cases - This is a place where tasks, notes, and other documents on researching human trafficking cases can be stored.
* Map - This is where you can mash up different data layers.
* Analytics – Configure automated reports and other analytics on data in the system.

2. Highlights

* This section contains important information about the current item you click on (not operable). In the future it will update automatically when a polygon I clicked on (e.g. California selected and pulls the highlights from that dataset).

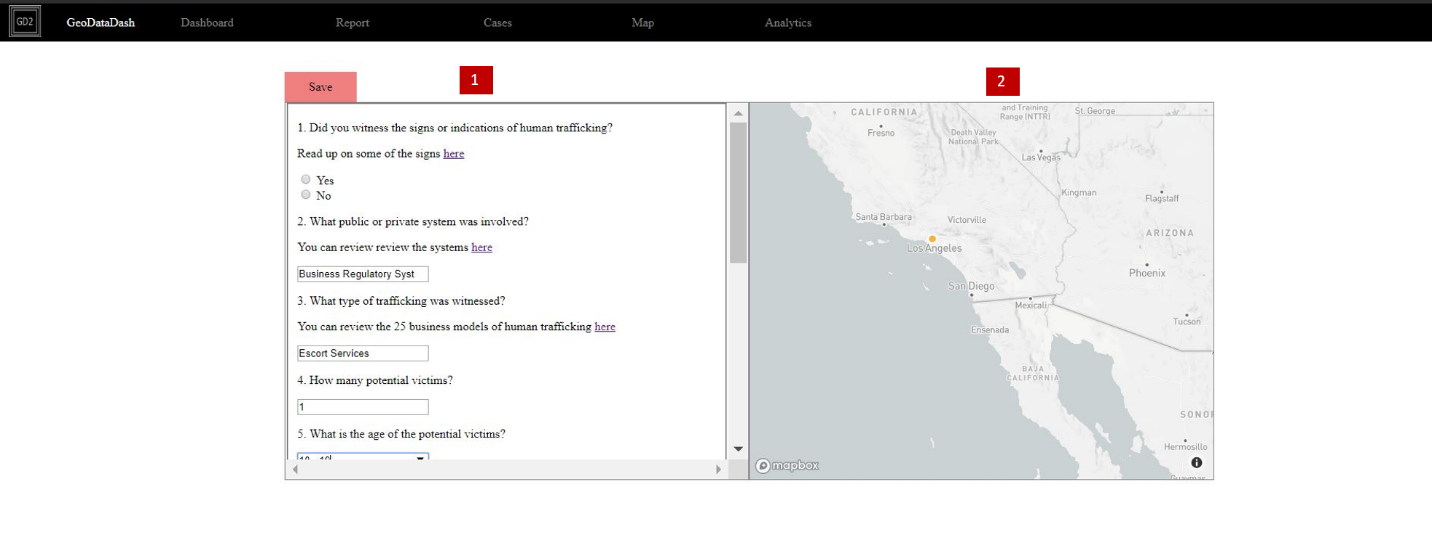
3. Map Section

* The bulk of functionality is here. The area on the left allows you to turn on/off different layers. Currently the ‘States’, ‘Counties’, and ‘Cities’ layer work.
* When a polygon is clicked on the map all reports within that area are displayed. You can move down from the state level to individual cities (the cities layer won’t retrieve reports). This allows for focused research on a given geographical area.
* A ‘Location’ text string also displays at the top right of the map. This will let the user know what they have clicked on (currently not functional).

4. Key Metrics

- When a polygon is clicked all the summary tables are retrieved and the charts updated. This provides a quick look at key information in area. In this case the Top 5 Systems, Top Business Models, and reports by date are fetched from the selected geography.

* Report



1. Data fields

2. Map

1. Data Fields

- The report section allows an app user to enter a human trafficking report. This information can be collected during a phone call, email, or other source. The report standardizes information to capture and enables a central location for all information to be logged and stored.

2. Map

* When all questions are filled the user clicks the map on the right to add a location to the point. Once finished the ‘Save’ button is clicked to submit the data.
* When a report is submitted the report is compared to all other reports within a 20 mile radius. Each field in the submitted report is compared to the corresponding field in the nearby points. A similarity score is calculated for each nearby point. If a threshold is reached that nearby point will be added to a list.
* This list of point is then converted to a polygon area that will indicate the area of similarity.
* In the context of the human trafficking dataset similar points may indicate related activity of human trafficking. When enough points are similar an alert is triggered to notify of potential trafficking activity.

**PORTABILITY AND OTHER USE CASES**

Other Use Cases

* GD2 can be utilized with any geo enabled data. In this implementation it was applied to the domain of non-profits that are fighting human trafficking.
  + Real Estate
    - If a home successfully sells for more than average market price, compare its categories (year built, square footage, style, neighborhood, etc.) to other nearby properties. If a threshold is passed, alert the user.
    - This could help agents identify properties with characteristics to make better decisions on where they invest their capital.
  + Civil Engineering
    - If sink hole occurs because of a storm pipe failure during the rainy season, this app could compare the fields (install date, material, condition, etc.) in that failed pipe to nearby storm pipes. If a high enough similarity is found an alert is triggered in the area where other pipes could fail.
    - This could help PW departments be proactive in preventing asset failures.

Portability

Although GD2 is written entirely with front end code it can be ported to a more scalable, serverless architecture. With some work and a bit of modifications more of the JavaScript could be implemented in Lambda functions. A database (like Dynamo DB) could then be integrated into the Lambda functions and you’re ready to go.

**FUTURE WORK**

* There’s a lot of UI/UX improvements that can be made (even well-developed apps are always evolving). For functionality I would like to integrate AWS Comprehend into the proximity scoring of GD2.
* With this you could take someone’s description of a potential human trafficking incident, run it through Comprehend, and pick out key phrases. These key phrases could then be added as another dimension to the location buffer and field comparison code to improve the accuracy of alerts. Imagine multiple people in a large geographical area describing suspicious behavior at a hotel involving a blue truck.
* Modeling human behavior in code is challenging. The methods I used to create alerts are quite simple and a lot of work would need to be done to make the data and alerts more useful.