**Original description**

Brief description of our project:

We are doing a standard project and creating a web app that will use data from the Duke police to allow users to interact with a database of crimes and locations reported on Duke’s campus. Duke Police provide weekly status reports on crime in the Durham area at this website: http://police.duke.edu/news\_stats/summaries/index.php. We would like to use a map on the web app to let users visually interpret the data, and we are considering using http://maplacejs.com/ to help integrate this feature. If we are feeling ambitious, we would love to include a feature to compute comparative statistics among different areas on campus.

Plan for populating the database:

We plan to scrape pertinent information from the individual weekly reports to build our database. In order to do this more efficiently in the future, we may set up a cron job to pull new reports and add them to the database as they appear on the host website under the ‘Current Week’ header. Furthermore, fields in the Building table such as 'address', 'type', and 'campus', will not be populated by the scrape calls, and will be populated by creating our own table from scratch that provides these categories of information about all buildings on duke's campus, and using this table to fill in said information in the Building table.

List of assumptions about data being modeled:

We assume that all scraped tables will be populated. When mapping extra attributes other than they key to Building/Area entity, we assume that the informational table will populate all values, or else values will be populated as NULL and not included in statistical analysis until that status has been manually changed.

E/R Diagram: Separate File

List of tables with keys underlined:

Crime (ID, Type, Date, Disposition, Area\_Name, inside/outside)

Building (Name, Address, Type, Campus)

Description of Web Interface:

Our app called “crimeloc” will feature a map through which users can see data about crime reports on and around Duke’s campus. Users will be able to select multiple different attributes of a crime, such as date, time, location, and type. Crimes with these attributes will appear on the map, and users will be able to click on the icons to read more about the specific incident (get the attributes they didn’t previously select).

**Updates For submission 2**

**Changes**

Due to difficulties it extrapolating location data from the dataset (i.e. coordinates from names), we may use a wordmap instead of plotting coordinates on the map. Secondly, we may not be able to sort by campus or building type for the same reason. We are going to try to use the information at maps.duke.edu to find a way to overcome the above listed problems, but if that is unsuccessful, we may try to present information in a different way that is still informative.

**Tasks complete to-date**

We have built the basic structure of the app as a NodeJS web app. This includes a basic user interface, and back end code for running the app. Secondly, we have created an SQL file by scraping the duke police data, which we will use with SQLite to operate the database aspect of the app.

**Remaining Tasks for project completion**

We still must solve the issue of how to obtain the needed information not provided by the dataset, or how to display the data we do have in a new manner if not able to generate additional data. Secondly, we must improve the current UI (we’ve done more work on this in a separate branch that is not currently merged, but will be ready by the final project presentation/submission). Thirdly, we will need to make sure we are sanitizing inputs, debug any strange or unwanted behavior, etc.

**Project Report**

**Description:** Our project is a web app that allows users to interact with a database of crimes occurring on Duke’s campus. We programmatically generated this dataset, so it is entirely artificial, not a list of actual crimes. We had initially wanted to use data from the Duke Police, who provide weekly status reports on crime in the Durham area at this website: http://police.duke.edu/news\_stats/summaries/index.php. However, due to difficulties in extrapolating location data from the dataset (i.e. coordinates from names), we considered use a wordmap instead of plotting coordinates on the map. Secondly, we may not have been able to sort by campus or building type for the same reason. We were going to try to use the information at maps.duke.edu to find a way to overcome the above listed problems, but if that is unsuccessful, we were going to present information in a different way that is still informative.

Unfortunately, we were told that this data was not nearly enough, so we decided to generate fake data. Since we are generating data ourselves, we are now able to add information like tags for crimes and exact coordinates for buildings to make our data easy to use.

**Details:** Our app is a single page html web app that uses NodeJS for server-side code and the npm module sqlite3 to allow interaction with an sqlite file where our database is stored. We generated our data using a Javascript file that somewhat randomly generated crimes from lists we hand-typed, and used the same sqlite3 module to create the database and add the building and crime data to it. Our front-end code allows users to select various parameters for viewing specific crimes, and sends the parameters to the server-side code. The server code receives the parameters from the client-side request and builds SQL queries from the parameters, then queries the database and retrieves the data. Then it sends the data in JSON form as a response to the client side, as well as HTML for displaying the data in table form. Upon retrieval of the data from the server, the client code then displays all the crimes on a map using maplace.io, and shows the data in table form using the HTML generated by the server code.

List of database tables:

Crimes: This table has entries for the crimes that occurred. Each crime type (arson, burglary, etc.) has one of several tags, categorizing the crime type as violent, theft, driving, etc. Each entry in the table has a crime ID, crime type, crime tag, date, time, disposition, location, and inside/outside. The primary key is the crime ID, and the location is a foreign key that references the building name. We randomly generated 1,200 crimes to use as entries in this table.

Buildings: This table is a list of buildings where crimes could possibly occur. Each entry stores the building's name, function (athletics, arts, academic, etc.), campus (East, West, Central), and it's latitude and longitude. The building's name is the primary key. There are 266 building entries.