CecilSafe Solutions: a Data Driven Approach to Crime Awareness at the Claremont Colleges

Calvin Aylward, Ethan Hardacre, Charles Shaviro, Adam Starr, and Thomas Thornton

June 1, 2018

Executive Summary

We introduce a novel method for presenting crime and safety data to the college public. While the Clery Act mandates that a daily crimelog be maintained, not all constituencies know about the log or are able to easily glean relevant information from it. Our solution leverages online analytical processing (OLAP) to answer multi-dimensional analytical queries about crime and safety in and around the Claremont Colleges as they are posed by members of the public.

1 Finding Our Site

CecilSafe can be found on the world wide web at `thttp://cecilproduction.herokuapp.com/`.

2 Project Outline

We will create an online analytical processing (OLAP) application to provide information about crimes at the Claremont Colleges. This visualization / analysis tool will let users query things like: what type of crimes occur around academic buildings? What types of crimes occur on Tuesday afternoons? Every day after 9pm? In the early morning? On South Campus? Near the Dining Halls? Where are the most bikes stolen? Where are cars most likely to be stolen or broken into? Can we identify high crime days or times for one area, but not another? There are many questions to consider. Users will have the ability to query the crime database and learn about what crimes happen at the Claremont Colleges, as well as when and under what conditions. The intended users for our application are the students at the Claremont Colleges.

2.1 Features

The following are the features that we have implemented:

1. We allow searches based on time, day of the week, date, month and year.
2. We allow searching on disposition such as all arrests and all open crimes.
3. We show show crimes on a map and consider crimes committed in certain campuses and building types.
4. We sort incidents by types of crimes such as theft or vehicle violations.
5. We also allow multiple conditions to be used at a time.
6. We provide a table of incidents and also a graph where the user can select the view.
7. We allow users to filter incidents based on weather.
8. We have detailed information about buildings such as if they are academic or residential.

3 Group Organization

The majority of our group wrote code. Adam spearheaded the parsing of crime logs from the Claremont Colleges and database creation. Chuck took care of supplemental data entry. Adam and Cal lead database design. Cal has taken charge of the Django setup and back-end, with help from Adam and Chuck. Cal and Adam connected the back-end to the DB and wrote the queries. Cal and Adam got Heroku set up. Adam transfered the DB to PosgreSQL.

Thomas and Ethan lead the implementation of visualizations including map display, table creation, and related charts. Adam lead project manager duties. Our code repository, data, and team chats are hosted on Keybase. The server is Heroku for the site and AWS for the DB.

4 Design and Implementation

4.1 System Architecture

The parsing of the crime log data was done in Haskell, with help from the parsing framework Parsec. The data is stored in a PosgreSQL database (it was transfered from MySQL). For querying the database we are using Django and Python. For the front-end, we are using HTML, CSS, and JavaScript. For visualizations, we are using the Google Maps API to display the locations of crimes. In addition, other visualizations (e.g., charts, graphs) are used.

4.2 Parser in Detail

The parser was written in Haskell using the parsec monadic parser library. As the formatting was at times inconsistent, some manual changes to the PDF formating were needed. The parser expects that the pdf log is converted to txt using the `pdftotext` function of the poppler library. Then parser then converts the text to a CSV that then can be converted into table entries for the database.

4.3 Front End in Detail

The front end is divided into two panes. The left pane displays data in several formats. The first is a heat map visualization that uses the Google Maps API. The coloring is scaled based on the size of the crime dataset being displayed. Zones that are fully red are in the top quartile of crime frequency. The second display shows bar graphs of the number of crimes, which are created using a javascript framework called CanvasJS. The axis can be changed to display different categorical data (e.g., day of week, disposition, temperature). The third display is a basic html table of the data.

The right pane is used for filtering data and is divided into several tabs and sub-tabs (e.g., crime type, campus, weather). Crime types also includes aggregate categories that can be used to easily select or deselect types of crimes (e.g., theft, bias). There is one sub-tab that functions differently than the others: building type. While selecting an additional crime type or disposition will increase the number of results, building type restricts the crimes to only those that occurred at the specific type of building (e.g., academic, dormitory).

After the user clicks 'Apply Filters' a JSON object containing the selected filters is sent in a Jquery POST request to the heroku server. The object contains both string descriptors (e.g., time of day, or date range) and record IDs (e.g., violation, campus, disposition). Crime results are returned in an array of JSON objects. Each object represents one crime with associated data (e.g., average temperature of the day it occurred). The front-end parses the data to calculate the number of crimes at latitude and longitude coordinates for the heat map, update the bar graphs, and refresh the table. There is also an 'About' page that includes information about each of the five team members.

You can see the current version of the website in the below images.

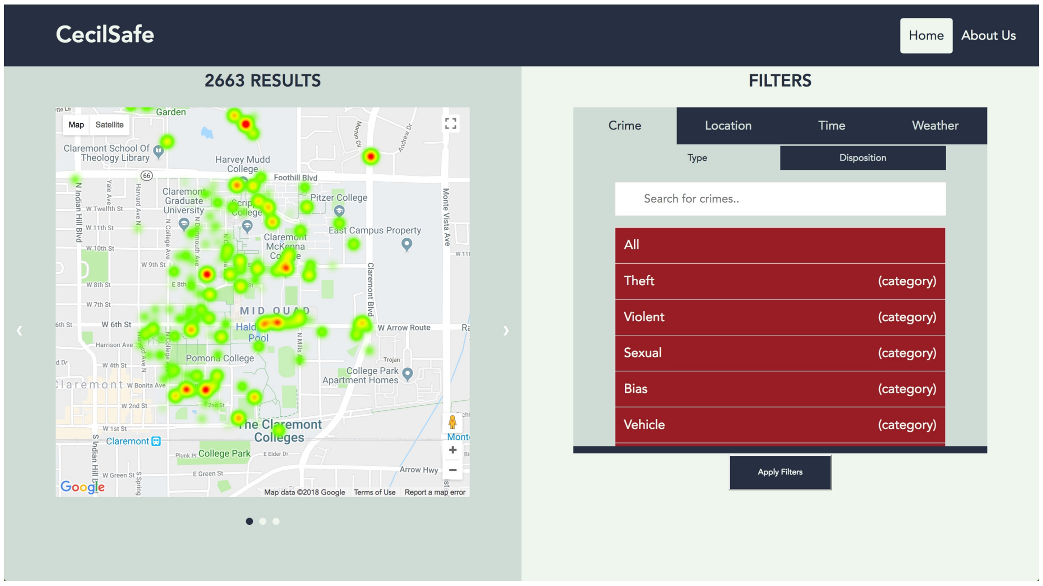


Figure 1: Website with display of heatmap and crime type filter

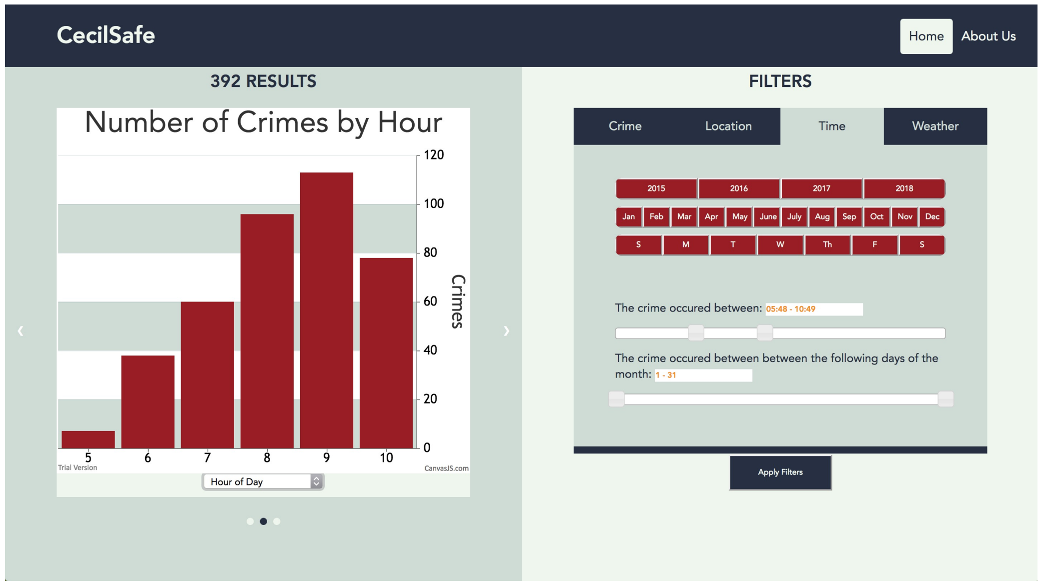
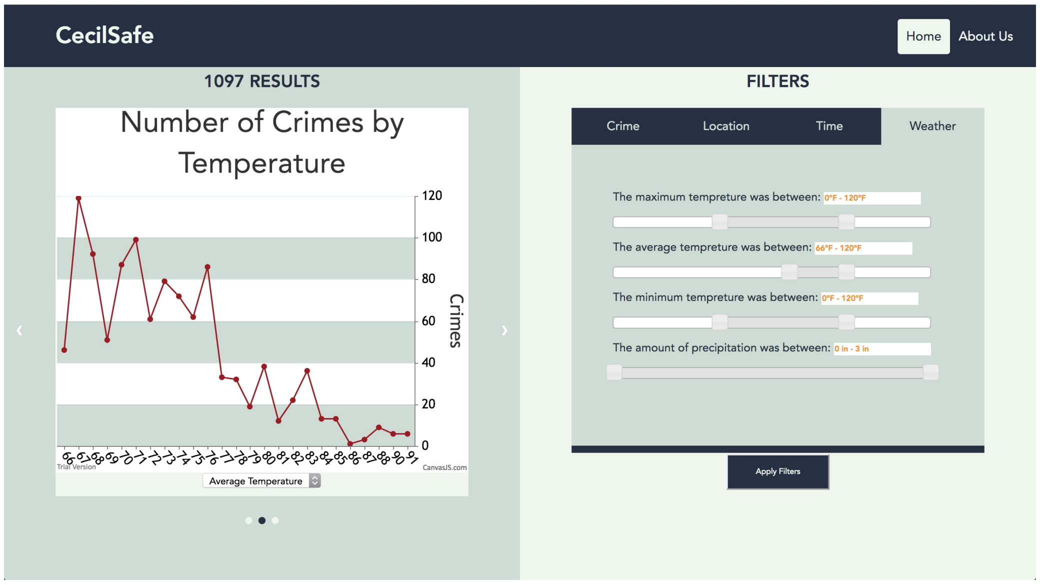

CecilSafe
Home
About Us
392 RESULTS
Number of Crimes by Hour
Crimes
Hour of Day
FILTERS
Crime
Location
Time
Weather
2015
2016
2017
2018
Jan Feb Mar Apr May June July Aug Sep Oct Nov Dec
S M T W Th F S
The crime occured between: 05:48 - 10:49
The crime occured between between the following days of the month: 1 - 31
Apply Filters

Figure 2: Crimes by hour with limited filter applied on hour



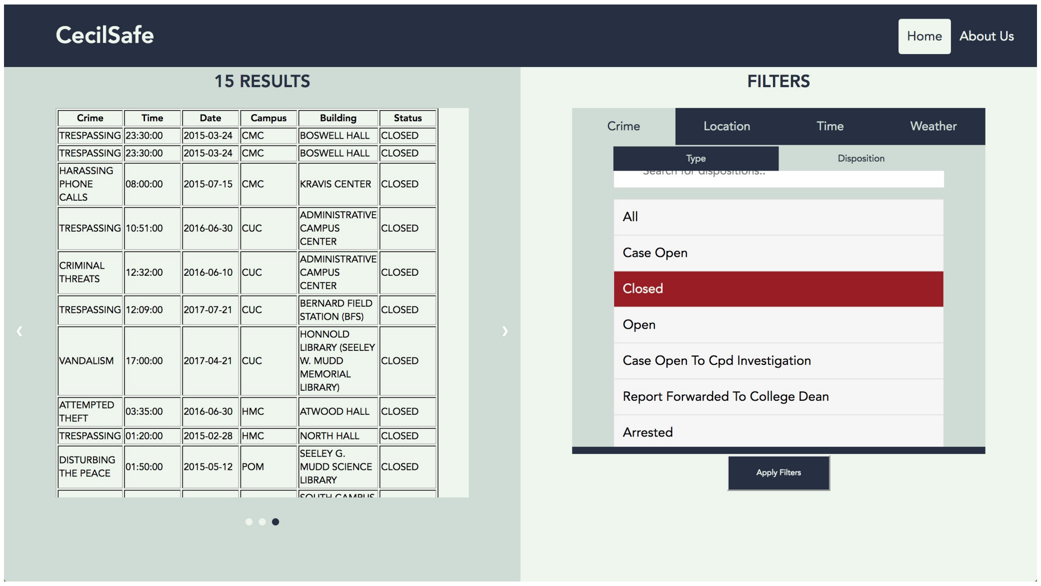
The maximum tempreture was between: 0°F - 120°F

The average tempreture was between: 66°F - 120°F

The minimum tempreture was between: 0°F - 120°F

The amount of precipitation was between: 0 in - 3 in

Figure 3: Website with view of number of crimes by average temperature of day with filter applied on tempreture range



Crime	Time	Date	Campus	Building	Status
TRESPASSING	23:30:00	2015-03-24	CMC	BOSWELL HALL	CLOSED
TRESPASSING	23:30:00	2015-03-24	CMC	BOSWELL HALL	CLOSED
HARASSING PHONE CALLS	08:00:00	2015-07-15	CMC	KRAVIS CENTER	CLOSED
TRESPASSING	10:51:00	2016-06-30	CUC	ADMINISTRATIVE CAMPUS CENTER	CLOSED
CRIMINAL THREATS	12:32:00	2016-06-10	CUC	ADMINISTRATIVE CAMPUS CENTER	CLOSED
TRESPASSING	12:09:00	2017-07-21	CUC	BERNARD FIELD STATION (BFS)	CLOSED
VANDALISM	17:00:00	2017-04-21	CUC	HONNOLD LIBRARY (SEELEY W. MUDD MEMORIAL LIBRARY)	CLOSED
ATTEMPTED THEFT	03:35:00	2016-06-30	HMC	ATWOOD HALL	CLOSED
TRESPASSING	01:20:00	2015-02-28	HMC	NORTH HALL	CLOSED
DISTURBING THE PEACE	01:50:00	2015-05-12	POM	SEELEY G. MUDD SCIENCE LIBRARY	CLOSED

Search for dispositions..

- All
- Case Open
- Closed
- Open
- Case Open To Cpd Investigation
- Report Forwarded To College Dean
- Arrested

Figure 4: Table of incidents where the case is closed

4.4 Back End in Detail

The back end is written in Python using the Django framework to handle GET and POST requests from the front end. We used the Python library psycopg2 to connect to the PostgreSQL database and make queries based off of user input. The psycopg2 library automatically escapes all strings when making queries for the sake of security. We have an environment file where the login for a database account with read-only permissions is stored and then read when a connection needs to be opened. Our site is hosted on Heroku and uses its Python build pack to setup and run the site. We are using a free Heroku account so there is only one web worker that sleeps after 30 minutes of inactivity causing slow response times on first load. Django provides an easy way to organize the back end, routing all URL requests through to a URL parser that can then forward the request to the associated view file where GET and POST requests are handled. If the request is a POST we read the data that was sent from the front end before making a database query, while GET requests don't have incoming data so the queries are simpler. The data is passed back and forth as JavaScript objects, or JSON, making for easy parsing.

We used left joins to allow for easy searches on traits that may be possibly NULL. For example if the user wants to consider particular types of buildings, only buildings will have those properties, so we must have a flexible enough query to handle possible NULL values.

4.5 Data Collection

We use data provided by the Claremont College's Department of Campus Safety. We also have daily temperature data for the period for which we have crime data from NOAA.

5 Database Details

The data store is hosted on Amazon Web Services (AWS). The software is PostgreSQL.

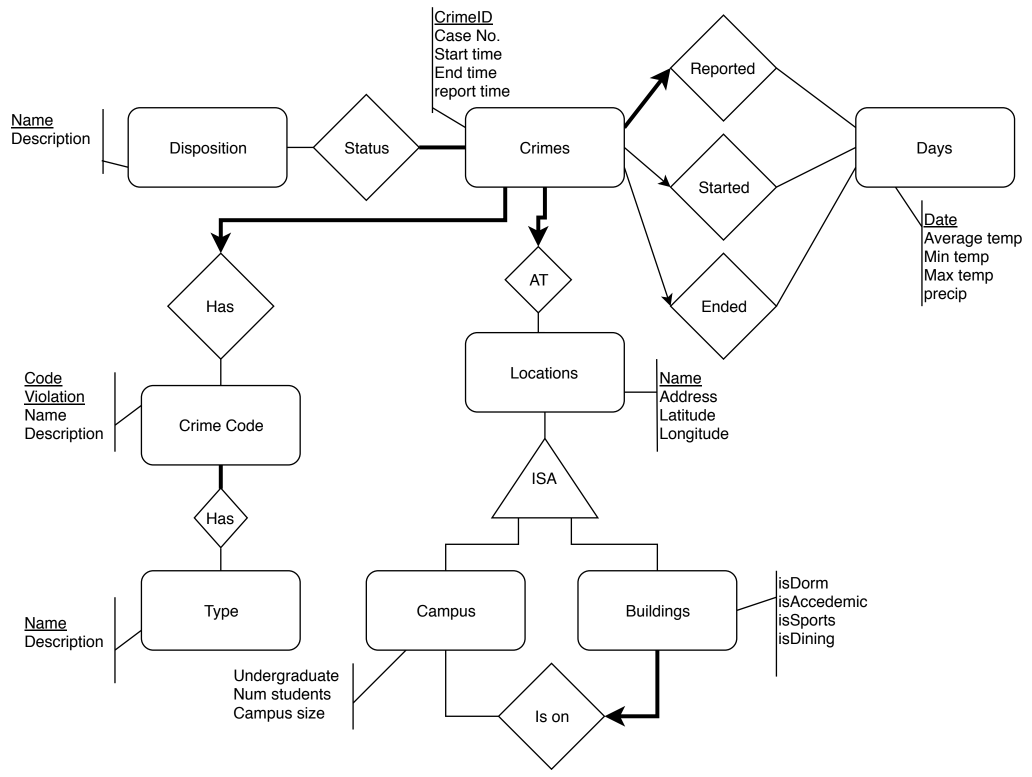


Figure 5: The Entity relationship diagram for CecilSafe

The entity relationship diagram in Figure 5 shows the structure of the database.

Each crime occurs at exactly one location. The types of locations vary. Some crime logs only list a campus as the location, while others may have a specific building and address. In order to address this difference in specificity, the location of a crime is a hierarchical relationship. Either a campus or a building can describe the location of the crime. Additionally, each building also belongs to exactly one campus.

Each crime has exactly one crime code that describes what the crime is that committed (e.g., 23 corresponds to “HARASSING PHONE CALLS”). Each crime type describes different qualities of the crime (e.g., violent or theft).

A disposition describes the current status of the crime (e.g., open, forwarded to police). A crime can also have multiple statuses. Finally, the crimes are reported on specific days (which have additional data that may show correlations, like temperature). Each relationship for the time of a crime (reported, started, ended) also has an attribute of the relationship which is the specific time it is reported at (not shown in Figure 5).

The relationship between dispositions and crimes could not be folded into one of tables because it is a many-to-many relationship. A crime can have more than one disposition status attached to it, for example, a crime could be open and forwarded to CPD. Therefore, in our schema in Figure 6 we have a table for the relationship which we call “Status”, linking our crimes with their dispositions.

There will be a many-to-many relationship between Crime Types and Crimes. The relationship will allow us to represent crimes that fall under multiple Crime Types, for example, Grand Theft Auto is an auto related crime and also theft. Crime Types are related to a specific crime through the relationship Types which connects the Penal Code of a crime to the appropriate descriptive types.

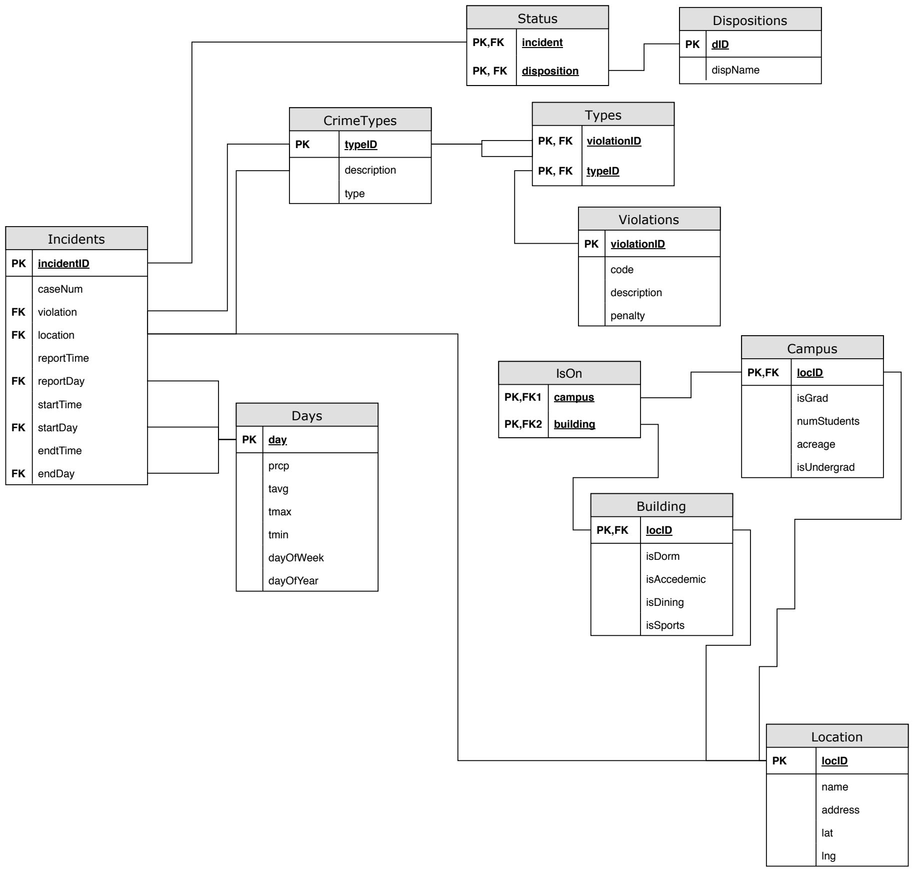


Figure 6: The relational schema for CecilSafe

6 Lessons Learned

We learned a number of valuable lessons throughout the development of CecilSafe. For one thing, our extremely divergent schedules made finding time to work together, and check-in with Professor Wu, quite hard to schedule. From practicing with the track and field team, to hosting mentor sessions, to meeting with other organizations, not to mention work for other classes and projects, our disagreeing time commitments had very little overlap, only exaggerating the importance of successfully communicating with each other, pushing changes, and otherwise keeping ourselves focused and in-tune as a team.

Although we occasionally struggled with making time for our Friday meeting, our weekly Monday night coding sessions were very productive, and frequently continued into the early hours of the morning. It was in these sessions that we pivoted between different hosting services and programming languages, as we gradually worked, through highs and lows, towards this final product that we built. These times also functioned as a great check-in period, when we'd update each other on the week's work. The stability that our weekly Monday meetings granted really helped us stay on task and focused on the next goal.

Connecting various technologies is often non-trivial and a lot of time is spent organizing the system architecture. Many bugs and limitations popped up, but we overcame them. For example, we found limitations with the Google Maps API, and previously used a MySQL database before we discovered that it was not compatible with Django. One other difficulty we faced was ensuring that our filters were applied correctly. Originally we were having difficulties passing the correct record IDs (e.g., crime ID, violation ID) between the front and back end. It was key for us to stay accessible to each other and on the same page as we encountered different problems arising from the combination of our different contributions.