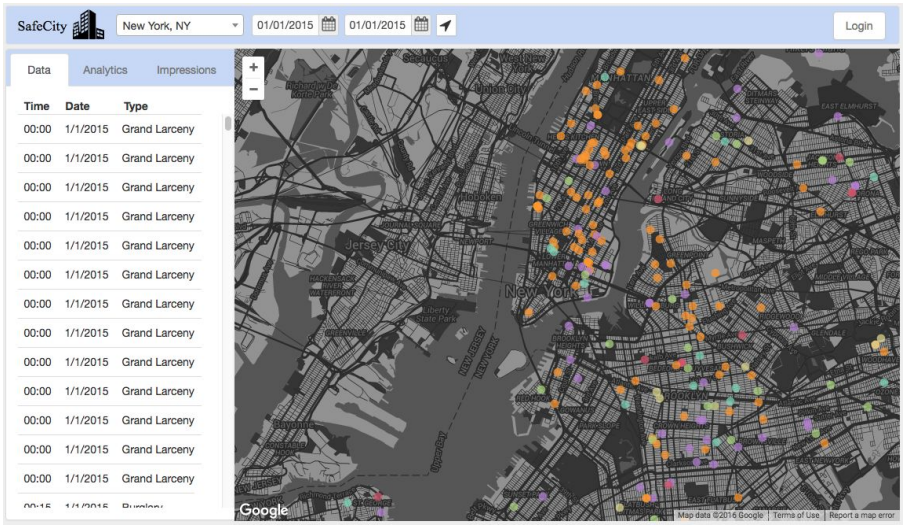
SafeCity Product Guide



Eric He, Stefan Keselj, Kavinayan Sivakumar

05.10.2016

Hansen Zhang

Dr. Christopher Moretti

COS 333: Advanced Programming Techniques

Princeton University

User Guide

Safecity is a crime mapping and analytics tool designed to deliver insights about crime to the user in the most clear and convenient way possible. This guide walks through all of its core features from a user perspective, exploring all the different ways to use the application. it can be accessed at: https://safe-city.herokuapp.com/ . Do not be alarmed if the app takes some time to reload; it is hosted on a basic Heroku package which sleeps if it has not been accessed in over an hour. Once it finishes loading after it has not been touched in awhile, subsequent loads should be much faster.

Map

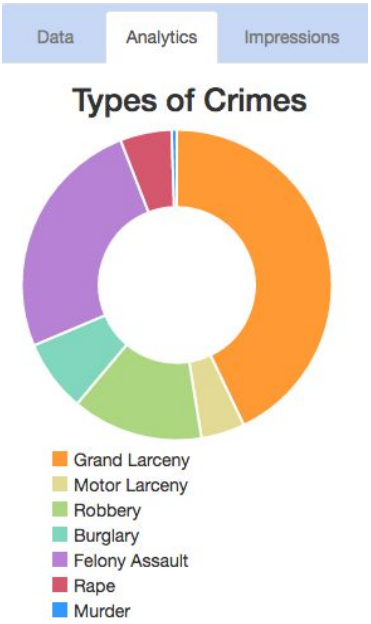
The first thing the user will likely notice is the large black and white map in the center-right of the page. This is the component at the heart of what SafeCity does, and it ties together all the features we will overview in this guide. By default, the user will be looking at a visualization of all the crimes that happened in New York, NY on January 1, 2015. Each crime is represented by a circular marker, with its color indicating its type. This is not the only representation of crimes; as we will see in the Search section there is also a heatmap version, but for now we will focus on the default. The user can pan across the map by clicking and holding and then moving the mouse across the screen. The markers eventually taper off if the user pans far away enough from New York. The map does not automatically load data according to any actions on the map; this initial map of crimes in New York will only display crimes in New York as long as the user is only interacting with the map.

To explore individual crimes, the user can click on any of the markers and see basic information about the crime corresponding to that marker. As shown in the example above, this information will take the form of the crime type, time of day, and date. To remove the window containing this information, the user can click the “x” symbol at the top right. To compare multiple crimes at once, the user can simply click on their respective markers; the map will not automatically close any information windows. To get a better overall idea of what crime in the city looks like, the user can zoom out by clicking the “-” box in the top left of the map or scrolling up with the

mouse wheel. Conversely, the user can get a better idea about the finer details of crimes by zooming in with the "+" box in the top left of the map or by scrolling down with the mouse wheel. The markers automatically resize depending on the zoom setting in order to always present the data in the most informative way. They say a picture is worth a thousand words; if that's the case, then a pannable, zoomable color coded map must be at least a book's worth.

Data and Analytics

Aside from the map, there are other helpful sources of information in the application that the user can browse through, namely the data table and the graphs. The data table, which is the default panel of the section located directly left of the map, lists out the time, date, and type for every single crime in the date range that the user selects. Although the table may not be best tool to gain insights from, it provides the application with a sense of transparency and honesty. It serves to give the graph and the data points on the map more legitimacy, as there is a lot of data the application is dealing with at any given time.



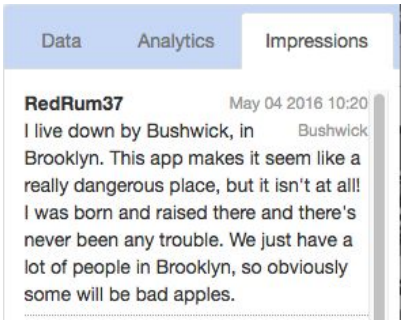
When querying a time frame that may have too many points, simply looking at the map may not be as helpful. Instead, a more compact and insightful mode of analyzing the data can be found in the analytics tab. This tab is located to the right of the data tab that shows the table of points. When the tab is clicked, a doughnut/ pie chart of the types of crimes in the range the user queries is shown. Visually examining the doughnut chart can give the user a good sense of which crimes occur the most frequently and which ones are more rare. Specifically hovering over the sections of the pie chart will allow the user to see a tally of the total number of occurrences of the selected type of crime. Finally, located directly beneath the chart is a legend that lists all the types of crimes that occurred during the time period queries. The legend serves not only as a reference for the sections of the doughnut chart, but also for the colors of the points on the map as well.

Search

The search functionally is located at the very top of the application. It allows the user to customize a query using three fields, city, start date, and end date. By default, when the user visits the webpage, the search is set to New York on January 1st (start date: January 1st, end date: January 1st). The city can then be chosen through a drop-down menu featuring New York, Chicago, and Los Angeles. The start and end dates can then be chosen through a calendar picker, or typed in manually in the search box. One all three fields are chosen and selected to the user's desire, the "go" button can be pressed, and the page will reload with the map, data table, doughnut chart, and impressions page containing the corresponding information.

Due to the large amount of data that the application is managing, queries between one and two days will display singular dots on the map, queries between three days to a week will generate a heat map, while queries over a week a temporarily prohibited, and will throw an error.

Impressions

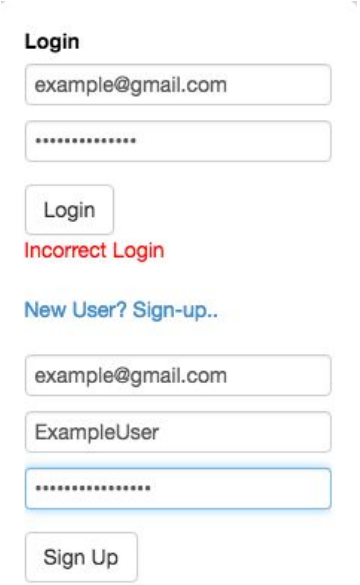


But there is much more to the story than just points on a graph or figures in a table. Crime is a social issue, and with any such issue there are many considerations that must be taken with respect to the subtle nuances of all the different lives involved. To interact with this dimension of the situation, the user can click on the third and final tab in the control panel: Impressions. This is the component of our app that will not be found in any similar products, which truly differentiates SafeCity. In this tab the user can scroll through a table of tweet like entries called impressions. Each impression display contains the author's username in bold in the top left, the date and time of post in the top right, the specific neighborhood of the selected city that the impression is about also in the top right, and the the main body filling the space below.

The user's experience will be enriched by taking some time to scroll through the impressions using either the mouse wheel or the clickable scroll bar to the right of the

window. Each impression was written by a member of its respective community, and contains information that mere crime statistics cannot convey. Perhaps an area is rife with markers, but in reality it simply has a much higher population density in recent years. Perhaps an area seems completely safe, but is actually so bad that the police are scared to go there and therefore no crimes get reported. Impressions are sorted by time and carefully moderated by our team to ensure content is appropriate.

Login and Sign Up



In order for the users to be able to create impressions of their own, they will have to become members of our community. This can be done by navigating to the top right of the webpage and clicking the button labelled “Login” and then clicking the “New User? Sign-up..” link. What pops up is a standard Signup form where all the user has to do is enter their email, username, and password and click the “Sign Up” button. It should be noted that there are multiple restrictions on the fields the user can enter. An email must contain an “@” character and a password must be at least six characters long. Furthermore, if an email or username have already been registered, a red error message will pop up saying “Email/Username Already in Use”.

Once the Sign Up process has been completed, the user will have full access to member privileges, specifically, the user can now add impressions of their own. This can be done by going to the “Impressions” tab in the control panel and scrolling to to the bottom, where a simple form as shown to the right should appear. The first input field is where the main body of the impression is inputted and the second field is a place to tag the impression by neighborhood. An impression must not be purely white-space, as that would be meaningless, so the form will not submit if the impression field does not have a non-whitespace input. The neighborhood field, on the other hand, is optional because not every impression can appropriately be tagged by neighborhood. To submit the impression, the user simply clicks the “+” button to the right of the fields or hits the enter key while typing the impression.

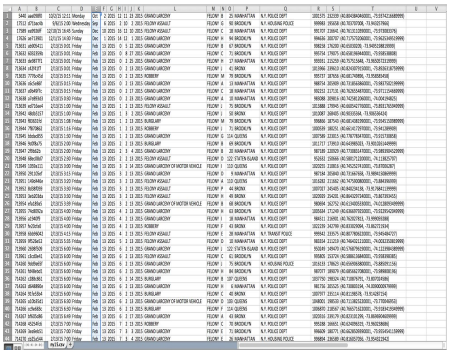
To logout, the user can press the “Logout” button which appears at the top-right of the screen just like the “Login” button did before. As with incorrect Sign Up

information, if a user attempts to login with the wrong email and password pair then a red error message saying “Incorrect Login” will appear beneath the input fields.

Developer Guide

SafeCity is a three tiered Meteor application which uses MongoDB as its database, a plugin called Iron Router for its middleware, and various Javascript libraries for its presentation layer. This guide walks through all of its core implementation details from the perspective of a developer who is about to continue our work on the project. The source code can be downloaded at: https://github.com/skeselj/SafeCity/tree/master/maps . To run the code, download Meteor and execute `meteor run` in the app directory.

Data Collection



The main driver of the SafeCity is its data. Data was collected for three cities, New York, Chicago, and Los Angeles, for the year of 2015. There was data available for 102,655 individual crimes in New York, 252,718 in Chicago, and 104,603 in Los Angeles; a total of 459,976 tuples of information about crimes. These files can be downloaded using the sources at the end of this guide, and a typical portion of the data is shown to the right.

Data files of crimes for each city in 2015 were downloaded in CSV format, and then sent through AWK and Python scripts in order to limit the categories to the relevant ones: latitude, longitude, offense, time, month, day, year, magnitude, city, and weight. These categories were chosen because they all have a certain function in the application such as pinpointing the location to plot the points, determining the categories to use in the chart, or figuring out a weight to be able to sort the crimes by date.

Backend

SafeCity is a Meteor application, as thus by default uses MongoDB for its backend. The main collection of the application is called markers, which hosts all the

data points mentioned above. The way that the data points were imported was first connecting to the heroku application hosting service and then using the mongoimport command. In order to maintain a certain degree of performance when the user queries data, most of the irrelevant data points will never be sent to the user. Depending on the day range and city that the user chooses, he or she will generally have to host around 300 to 3000 data points, which is far less than the close to 500,000 points hosted on the server. The way that the management of data points between server and client is done is through a Meteor feature known as publications and subscriptions. A publication defines a kind of subset that the client would want to retrieve from the server, and a subscription to a publication officially draws data to the client based upon the publication. When the user fills up the search fields and hits submit, a new subscription is generated, and a new set of data is sent to the client.

The other two collections which the application uses are called impressions and users, which store exactly what their names say they do. The use of these is less intricate than the use of the markers collection, because these are queried much less often and in fewer ways. Impressions are queried by city at the same place where markers are queries by city, and users are only ever queried using specific email, username, and password fields. These queries, are once again done through the Meteor publication and subscription system.

Middleware

One of the core tenants of Meteor is the use of templates. Although templates are a cool way of organizing features of the application, they present two problems. The first is how to get different templates to talk to each other, for example, how to get the date-time picker to talk to the map. The second problem is sequential loading, for example, making sure that the pie chart loads itself after the data has been received from the database.

Both of these problems are solved with a popular Meteor package known as Iron Router. Iron Router not only stores the necessarily variables in the URL (city, start date, end date) so that relevant data can be retrieved accordingly by a certain template, it also has a waitOn function that allows for a template to be loaded only after a certain event has happened. In SafeCity, the main template is only loaded after the proper subscription is finished processing, or in other words all the data is received. In addition, Iron Router also has a nice loading template attribute, which displays a custom loader as the subscription is being processed. Furthermore, Iron Router will

ensure that the user will have all the functionality of the application without altering any of the defaults on the server side. Iron Router is essential to the proper management of data and a smooth and consistent user experience.

Presentation

The general front-end theme for our website is done in Bootstrap because it makes the application resizable and gives us access to basic preformatted, consistent components to start with as baselines. The two most important such components are the navigation bar and control panel, which make use of the `navbar-default` and `panel` built in bootstrap classes. In addition to these we also borrowed code for the impressions scrollable table form a Dashboard template on StartBootstrap and tweaked it to conform to our standards in CSS. One specialized component of the presentation layer which was not offered in Bootstrap or its templates was the pie chart used in Analytics. Here we used a Meteor package known as Chart.js. The chart itself is reactive, and is updated after every new subscription by using the appropriate find queries in the Mongo collection. Chart.js was preferable to Google Charts because it is a built in package and can be directly manipulated through the Meteor javascript.

The map component was built by integrating the Google Maps Javascript API with Meteor using an Atmosphere package called `dburles: google-maps`. This package abstracts out a lot of the boilerplate code that Meteor requires in order to use external libraries by allowing us to call simple and intuitive functions like `create, load,` or `ready` on an a `GoogleMaps` object instead of repeatedly copy-pasting mechanical code. Most of the code behind the map is contained in the `Template.map.onCreated` method on the client side javascript code. It is here where we configure the center, zoom, and colors of the markers using basic Javascript functions to return the parameters of each. To style the map itself, a template downloaded from Snazzy Maps in the form of a JSON file was used. If the middle layer tells us that the number of days selected is 1 or 2 through a boolean variable `heat`, then we use the function `google.maps.Marker` to plot every point in the collection of crimes given to us by the middle layer. In order to be able to resize the markers if zoom level is changed and display information windows if markers are clicked, we add listeners for these two events through the built in *`event`*`.addListener` method. If the number of days selected is between 3 and 7, we pass the collection to a Google

Heatmap using `google.maps.visualization - .HeatmapLayer` instead to get the heat map. If the number of days selected is more than 7, we display the map but do not feed it any marker or heatmap data. The reason that the map is limited in this way is to preserve performance. Usually, the difficulty with a large volume of data is not with retrieving it from the database, but rather having Google Maps process it. After repeated testing, it was found that queries longer than a week, especially for Chicago which has a denser dataset, would slow the loading time down a bit too much.

To present our query components, the date selector and city selector, in a way which was quick and intuitive for the user we needed to use intermediate templates compatible with Meteor's Session system. The packages which were most consistent with our standards and the simplest for us to use were `datetimepicker` by `eonasdan` and `meteor-autoform-select2` by `aldeed`. Both were logical in their design and intuitive to use, we simply fed in the data given by the middleware into them to get the desired search and selection functionality.

To format our login and sign-up panel, we built on top of a Meteor package called `accounts-password`. The package only came with basic email and password fields, so we had to add a username field on top in order to keep identity anonymous in impressions. Another important addition we made to the existing package was error throwing, as it did not do anything different when a user tried to login or signup with existing credentials. This was resolved using the `callback` function built into Meteor accounts, which allowed us to automatically check the most recent query to check for errors.

Hosting

SafeCity is hosted on Heroku, which has built in support and packages for Meteor applications. The data is managed through a plugin called mLab MongoDB, which allows for convenient access to MongoDB data and tools. The markers collection, which hosts all the data points the application can use, takes up about 120 MB of space. Both the users and the impressions collections currently take up less than 100 KB each, but can change drastically depending on the amount of user content produced in the future.

The amount of traffic and data that we have for our application allows us to use the free version of Heroku. This gives us limited access to dynos, which manage the application processes, as well as only 18 hours of uptime a day. In addition, if the

website is not pinged for over an hour, the application goes to sleep, and the next user that tries to access the page will experience longer loading times. In order to prevent this last problem, however, the application is registered on a third party website called Kaffeine that pings it every thirty minutes.

In the future, if the site expands with more data, or receives more traffic, it is highly likely that the current free service will need to be upgraded.

Data Sources

1. New York
   https://data.cityofnewyork.us/Public-Safety/Historical-New-York-City-Crime-Data/hqhv-9zeg
2. Chicago
   https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2
3. Los Angeles
   https://data.lacity.org/A-Safe-City/LAPD-Crime-and-Collision-Raw-Data-2014/eta5-h8qx

Package Dependencies

1. Google maps Javascript API
   https://developers.google.com/maps/documentation/javascript/tutorial
2. Dburles' google-maps
   https://atmospherejs.com/dburles/google-maps
3. Snazzy Maps
   https://snazzymaps.com/style/38/shades-of-grey
4. Bootstrap
   https://atmospherejs.com/twbs/bootstrap
5. StartBootstrap Dashboard Template
   http://startbootstrap.com/template-categories/admin-dashboard/
6. Eonasdan's datetimepicker
   https://eonasdan.github.io/bootstrap-datetimepicker/
7. Aldeed's meteor-autoform-select2
   https://github.com/aldeed/meteor-autoform-select2