Cal ToxTrack:

A Full Stack Web GIS for Mapping Pollution in California

by

Megan Luisa White

A Thesis Presented to the
FACULTY OF THE USC DORNSIFE COLLEGE OF LETTERS, ARTS AND SCIENCES
University of Southern California
In Partial Fulfillment of the
Requirements for the Degree
MASTER OF SCIENCE
(GEOGRAPHIC INFORMATION SCIENCE AND TECHNOLOGY)

May 2021

This one is for you, Dad!

Acknowledgements

Thanks to my family and friends for their continuous support.A tremendous thank you to my committee— Jennifer Bernstein, Jill Sohm, Jennifer Swift, and Robert Vos— for their help and guidance throughout this project. An additional thank you to Peter Jang, Colin Geddes, and Ramiz Sami for their great help.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AQS	Air Quality System
CARB	California Air Resources Board
DMR	Discharge Monitoring Report
EPA	Environmental Protection Agency
GIS	Geographic information system
GDP	Gross Domestic Product
LEPC	Local Emergency Planning Committee
MVC	Model, View, Controller
NEI	National Emissions Inventory
NPDES	National Pollutant Discharge Elimination System
NRCS	Natural Resources Conservation Service
OGC	Open Geospatial Consortium
SERC	State Emergency Response Commission
SSI	Spatial Sciences Institute
TERC	Tribal Emergency Response Commission
TRI	Toxic Release Inventory
USC	University of Southern California
USFS	United States Forest Service
USGS	United States Geological Survey

Abstract

Past chemical emergencies in the United States prompted the initiation of a variety of toxic substance and pollution control programs and regulations, including the Emergency Planning and Community Right-to-Know Act and the Clean Air Act. While these have produced decades-worth of valuable pollution datasets, they are stored on a government website in a collection of CSV tables. This method of accessibility is largely incompatible for analysis due to the static nature of tables— the need to download them locally and appropriately query them to extract relevant data. For analysis, data is best visualized with dynamic tools and within interactive environments.

This project focused on the public's right-to-know about toxic chemical releases in their community by developing a geospatial web application called Cal ToxTrack. Built from scratch using PostgreSQL as a database, GeoDjango as a Python development framework, and Leaflet as a JavaScript framework, it effectively visualizes chemical releases and provides interactive tools to help explore pollution data. To ensure that the application does not depend on access via state and federal governments or whether the developer has continued access to commercial products, this application developed entirely from the backend database through the front end and interface, otherwise known as full stack development, relies on publicly-available pollution datasets downloaded and hosted by an API created by the developer and was entirely created using open source tools. With Cal ToxTrack, users can utilize a map and spatiotemporal tools to visualize what chemicals have been released, to what magnitude, and where; practicing their right-to-know.

Chapter 1 Introduction

A defining characteristic of Web GIS is that applications should target a broad audience, including those that don't know anything about GIS (Fu and Sun 2011). This thesis embraces that recommendation, and aimed to develop a web application that provides the general public with tools and data to explore and analyze pollution in California. Within the application, users are able to (1) zoom to an address or pan a map to visualize locations and magnitudes of toxic chemical releases, (2) filter data by attributes like chemical name or facility, (3) temporally analyze the data with a dynamic time slider widget, and (4) visualize data with a histogram chart. A user-friendly interface relied on publicly available datasets and widgets developed with open source frameworks to facilitate analysis of both recent and historic releases of toxic chemicals. While the related works section reviews the application's use for public awareness of pollution and its applications to environmental justice, the application's flexibility facilitates research questions ranging from research on watershed pollution to decisions as to home values based on proximity to sites generating toxins. This flexibility is facilitated via the user's ability to upload their own data, measure distances, and create buffers while simultaneously visualizing pollution data on the map. The web application's main goal is provide the public with a means to access and better understand chemical output in their neighborhood and beyond.

1.1. Motivation

The motivation for this web application stems from legislation passed in 1986, which is known as the Emergency Planning and Community Right-to-Know Act (EPCRA). The act states that emergency officials and the public need to stay informed as to how toxic chemicals are being handled at facilities. Should an emergency occur, knowledge of the chemicals that are accidentally released can help emergency responders quickly determine risk and response plans best suited for specific chemical's toxicology. This ensures public safety and gives them the right-to-know what chemicals they may be exposed to.

Before it was withdrawn in December of 2019, TOXMAP, a popular web application that mapped pollution, helped enforce the "community right-to-know" under EPCRA. The application was widely used by the public and in classrooms (Roth 2014). It also served as a useful research tool in public health fields. For example, it was used as a large Superfund site source in research that engaged Latino communities in grassroots greenspace intiatives to improve local health outcomes (Fernandez 2018), and in a study (Persico 2020) that aimed to address the scant research conducted on how children in early stages of development may be affected by pollutants released from TRI facilities. TOXMAP was utilized to generate a map of TRI sites in relation to population density across the U.S. (Figure 1).

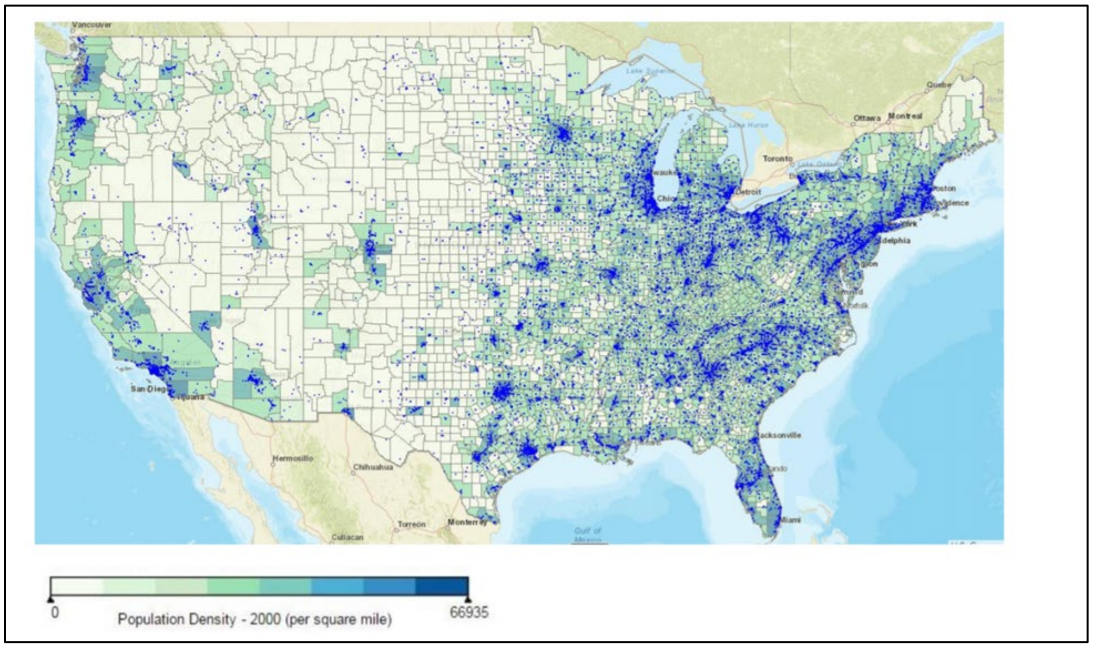


Figure 1. Map created in TOXMAP of population density and TRI reporting facility locations in the United States (Persico 2020)

TOXMAP's widespread use is why people were surprised and concerned when the National Library of Medicine (NLM) announced TOXMAP's retirement. GIS users and developers were encouraged to continue work with the underlying data, which can be found in the archived application website. However, the retirement of this Web App fragments the relationship between a community and their “right-to-know”, given that the general public generally does not have expertise in GIS and thus would not know how to employ open-source data in their own GIS applications, much less have access to these types of data analysis platforms. Additionally, the datasets are currently provided by U.S. government agencies and are not all available in visually or quantitatively friendly formats. As quoted in Schulson (2019), "because this [environmental data] information has gotten so complex, and there's so much of it, it’s very difficult for someone who's not really trained in the area to navigate it. This tool actually cut through all the jargon, all the different interfaces that EPA, for instance, puts up before you get to the actual data that you're interested in”. This project aims to bridge the gap between toxic chemical data and the California public by providing an application that is suited for people who are both familiar and unfamiliar with GIS. It addressed this new obstacle in public accessibility by creating a web interface that expands on the now defunct TOXMAP by aggregating pollution datasets to enable exploration by the general public and enabling additional analysis techniques.

TOXMAP’s retirement violates the necessity for *all* public members to have access to pollution. Public awareness is crucial for environmental justice, environmental change, and public proactivity (Exchange Product 2014). Even with pollution prevention programs in place, any facility use of toxic chemicals can introduce risk to the public. Because of this, it is imperative that citizens have access to this information, not just government and the private sector. Ultimately, the stakeholders who will be most impacted by toxic chemical handling are the citizens who are geographically proximate to risk sites and as they are most likely to be directly affected by their use (Morteza et al. 2019). This places a high level of responsibility on these citizens to monitor toxin management standards, and ensure that they are being met to protect their wellbeing. Given the importance of the dissemination of this type of information, the method of delivery and/or the quantity of data determines whether or not members of the general public will have the tools, education, and/or resources to protect themselves and their families from hazards. Web applications that facilitate knowledge acquisition as to toxic chemical use and release can help communities three-fold: (1) individuals can prepare for evacuations in the event of an accidental leak, (2) they allow individuals to make informed decisions on where to live based on their health conditions and the type of chemical exposure they might be subjected to, and (3) they can facilitate public action when communities are at risk of environmental injustice (Taylor 2014, 13).

1.2. Study Area

This web application covers the state of California, which has the 5th largest economy in the world. It is the most populous state in the United States, with almost 40 million people and a 3.2 trillion-dollar gross domestic product (GDP) during the last quarter of 2019. Some of the largest industries in California are agriculture (which yields more crops and farm-based goods than any other state), tourism, technology, and manufacturing. In terms of raw contribution to total GDP, the industries bringing in the most value for California were finance and real estate, professional and business services, health care and educational services, and government (Bureau of Economic Analysis 2020). California also leads the nation in cash farm receipts while providing over a third of the country's vegetables and two-thirds of the country's fruits and nuts (CDFA 2018). California's high economic output deems it a valuable subject for analyzing toxic chemical release.

Past studies, conducted at the national level, have found a correlation between high economic output and increased environmental degradation (Collins et al. 2020). This, of course, varies among states based on their state-level environmental laws. However individual industries disproportionately contribute to the nation's total pollution (Collins et al. 2020) which explains why often times the correlation between economic output and total pollution is not apparent. For example, California's total chemical release was 39 million lbs in 2019 from 1,189 TRI-reporting facilities (Figure 2) while nearby states with lower GDPs like Nevada had a total release of approximately 336 million lbs from 142 TRI-reporting facilities. To summarize, California has more facilities and higher economic output than Nevada, but still releases less chemical waste when considering raw output. Thus this suggests that the correlation between GDP and magnitude of pollution is too generalized to apply to state-level pollution patterns.

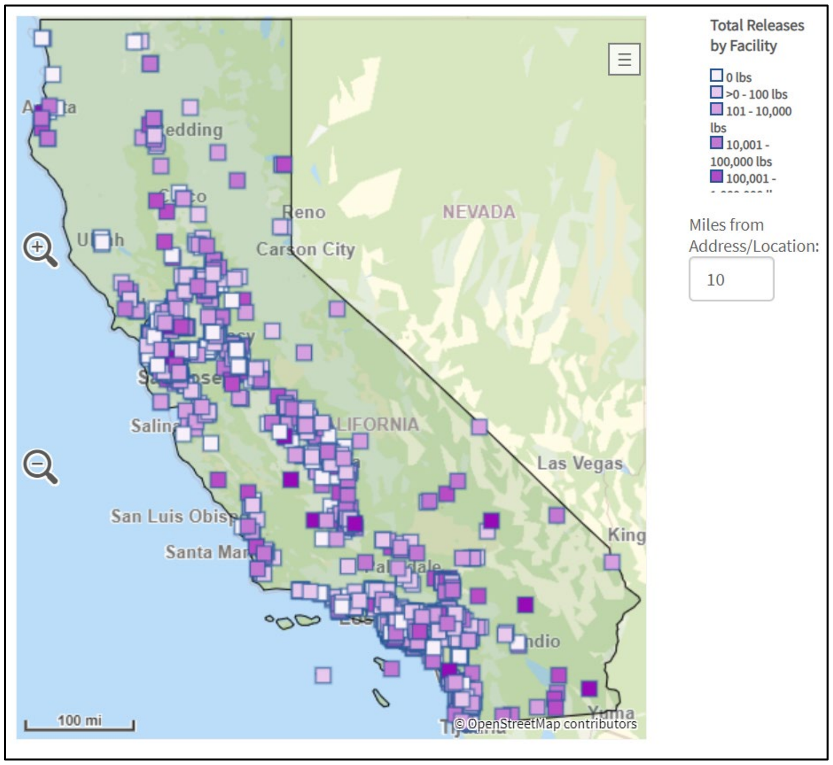


Figure 2. TRI Search Plus results after filtering for California 2019 TRI releases. While this map data appears clustered and contains flaws, it represents the 1,189 facilities reported around 39 million pounds of toxic chemical release in 2019

Furthermore, while perhaps generalizable at the national level, it's apparent that a states economic output is not a strong predictor of higher rates of pollution. For example, when analyzing state contribution to the GDP of the entire U.S., chemical manufacturing, which is California's main TRI-industry and less than 6% of the U.S. total TRI releases, has contributred 378 billion dollars to the total GDP since 2007. Meanwhile, the metal mining industry which contributes 40% of all U.S. chemical releases only contributes 26 billion dollars to the U.S. GDP. Nearly all U.S. metal mining facilities are in Nevada, while the numerous chemical waste facilities are the main contributors to California's toxic waste output.

Therefore, when attempting to understand a state's pollution trends and the implications for communities, there are multiple factors that need to be considered. These include the scale of analysis, number of facilities and respective output, and the geographic distribution of facilities. An interdisciplinary approach environmental justice calls for flexible applications that support various datasets and spatial scales. Not every state has the same political orientation towards environmental issues as California, which is discussed in the next section. However while this application's geographic scope is California, similar efforts can be conducted in other states using this general framework.

Compared to other states, California has an longstanding history of environmental protection legislation. CalEPA partly attributes this to 19th century post gold-rush concerns held by the public with respect to water quality and flood risk. California's water quality program, outlined in the 1969 Porter-Cologne Act, served as the model for the federal Clean Water Act. In the 1950's, the state established the nation's first air quality program, and late-80's state amendments to the Clean Air Act led to the federal Clean Air Act Amendments of 1990 (CalEPA 2016). The California Environmental Protection Agency includes boards and committees including the California Environmental Quality Agency, the California Air Resource Board, and the Department of Toxic Substances Control, which enforce and facilitate important pollution measures. While not necessarily representative of the entire country, California is an ideal region in which to test this type of application. There is environmental data and resources available to not only replicate some of the tools lost with the retirement of TOXMAP, but also to expand on them. With California's historic leadership in environmental regulation, it is an ideal study site on which to build this web application.

1.3. Overview

The methodology for this project consisted of three phases: data preparation, back end web development, and front end web development. This methodology is discussed in detail in Chapter 3. But to summarize, all data sets were downloaded and edited to be displayed. This included aggregating facility releases by year, as well as combining releases by chemical (lbs) and toxic weighted pound equivalent (TWPE). The back and front end development steps were completed using a programming architecture called MVC (Model Controller Viewer). The back end included inputting all relevant datasets into a database then utilizing a programming framework to allow client (web browser) requests to this database for the application to access data dynamically. The front end development involved the web design, map display, and widget configuration. These three components were free and open sourced. Concluding steps for this thesis included user testing once the application was in its beta phase.

Chapter 2 reviews several web applications and tools that have been designed to provide the public with access to environmental datasets, while Chapter 3 outlines the methods used in this thesis.

Chapter 2 Background Information

This chapter reviews relevant studies realted to this California Web GIS for Pollution mapping. Several applications are reviewed and compared to the planned aspects of this proposed application. Section 2.1 discusses the toxic chemical management programs in the U.S. which provide pollution data, which are critical with respect to the robustness of the data inputted into this web application. Section 2.2 discusses applications and data services that involve pollution data and section 2.3 explains the importance of web GIS with respect to public awareness and participation in efforts for protection and environmental justice.

2.1. Environmental and Public Health Protection Programs

Federal, state, and local government agencies have enforced important programs under laws that protect the public health and environment from hazards posed by industrial operations. These programs have maintained and provided the datasets that form the foundation of this application. The acts that mandate the programs are the Emergency Planning and Community-Right-to-Know Act (EPCRA), Clean Water Act (CWA), and the Clean Air Act (CAA).

2.1.1. Laws and Programs that Provide Pollution Data

Following the events of a fatal, extremely toxic chemical escape in Bhopal, India in 1984 and an accidental release of the same chemical in 1985 in West Virginia (The New York Times, August 12, 1985), Congress passed the Emergency Planning and Community Right-to-Know Act (EPCRA) in 1986. This established requirements for states and communities to work with facilities to report hazardous and toxic chemicals so the public could be informed as to industrial chemical use, release, and risks. With this came the creation of the State Emergency Response Commissions, Tribal Emergency Response Commissions (TERCs), and the Local Emergency

Planning Committees (LEPCs), which currently work to implement and review emergency plans, process public information requests, and provide constituents with information on chemicals used in their communities. With these requirements in effect, programs like the Toxic Release Inventory (TRI) evolved. These programs require facilities to report their use of toxic chemicals that pose a threat to human health and the environment, which then becomes an available source of information to the public (Office of Land and Emergency Management 2017). Facilities must report their management of chemical waste if they meet three criteria: 1) they fall within one of specified industries, 2) employ 10 or more full-time employees, and 3) manufacture a TRI-listed chemical in quantities that exceed the threshold value. The threshold value is normally 10, 500, 100, 1,000, or 10,000 pounds depending on the chemical's toxicity (US Environmental Protection Agency 2020). This program is a pillar in environmental safety as it provides the only comprehensive list of industrial chemical use available to the public.

The TRI program has undergone many revisions, changes, and additions since its origin in 1986. In November 1990, under the Pollution Prevention Act (PPA), the TRI was expanded to require facilities to include additional data on the chemicals they release, including how the chemicals are managed through recycling, energy recovery and treatment processes. Following this, a rule was passed in 1994 that added 286 new chemicals and chemical categories to the reporting chemical list, greatly expanding the scope of the program to cover more than 600 chemicals (EPA 2017). There has been instances where the TRI program has issued rules that ease reporting requirements for facilities, only for these rules to be reversed with the following administration. For example, the TRI Burden Reduction Rule was issued by the EPA in 2006 to raise the reporting threshold for all non-PBT chemicals from 500 pounds to 5,000 pounds as long as the amount of toxic chemical waste that was not sent off-site to be recycled or recovered remained below 2,000 pounds. However, this rule was reversed to the original under the Omnibus Appropriations Act of 2009 (EPA 2016).

The Clean Water Act of 1972 (CWA) provides the infrastructure for regulation of pollutant discharges in US waterways. It deems any point source pollution discharge into water unlawful unless a permit is acquired through the National Pollution Discharge Elimination System (NPDES). NPDES issues permits, regulates discharges, and provides the Discharge Monitoring Report (DMR) which quantifies the releases into waterways from each permitted facility. The DMR is one of the primary datasets used to assess water quality in tools and applications discussed in this section (33 U.S.C. §1251 et seq.) and in Cal ToxTrack.

The Clean Air Act of 1963 (CAA) requires national ambient air quality standards to be set for pollution control with respect to common criteria pollutants like particulate matter, ozone, sulfur dioxide, nitrogen dioxide, carbon monoxide, and lead. The CAA was the first federal recognition of the smog issues that major cities had been facing for decades. In 1943, in the middle of World War II, a thick cloud of smog covered the city, causing residents' eyes to string to such a degree that they thought the Japanese were attacking with chemical warfare. This later recognized "smog attack" was traced to Southern California Gas Company's Aliso Street Plant that manufactured synthetic rubber. This raised concerns about smog releases from other facility plants, chemical refineries, and car exhaust (South Coast AQMD n.d.). Issues like smog attacks, acid rain, regional haze, and ozone depletion spurred public action and led to the enactment of the CAA. It is responsible for the National Emissions Inventory (NEI) that collects estimated emission data and the Air Quality System (AQS) database that collects monitored ambient air pollution data from over thousands of monitoring stations (42 U.S.C. § 7401).

2.2. Pollution Mapping Applications

The web has potential to enhance opportunities for government agencies to comply with EPCRA's right-to-know requirement via tools and web applications. With the increasing interconnection between GIS and web development, more data download services and pollution mapping applications have emerged. Multiple applications have been built by the EPA, as they have authority over many of the data collection programs. Other applications have been built through EPA's collaboration with organizations like the USGS, USFS, NRCS, and others. This section discusses these tools by source, namely the EPA and the EPA in conjunction with collaborators. Tools include mapping applications, web services, data download tools, and analysis dashboards. This section will only review the applications that are pertinent to the development and goals of this project.

2.2.1. EPA Pollution Resources

The EPA provides the vast majority of publicly available environmental data. They have compiled the extensive pollution and environmental compliance program datasets into a cross-referencing database system called Envirofacts. Envirofacts is a portal to the data used for the applications and tools reviewed here. It provides the public with access to databases like the Toxic Release Inventory (TRI), National Priority List (NPL), registry of reporting facilities, permitting compliance reports, and more. Many tools, services, and web maps either pull data dynamically from Envirofacts through an API, or they use the datasets provided by Envirofacts (Figure 3). For example, user interaction in EnviroMapper for Envirofacts will construct a URL for the Envirofacts API which will return the desired data associated with facilities represented on the map. Meanwhile, the Cleanups in My Community application uploaded datasets that may be acquired from Envirofacts into an ArcGIS Online server.

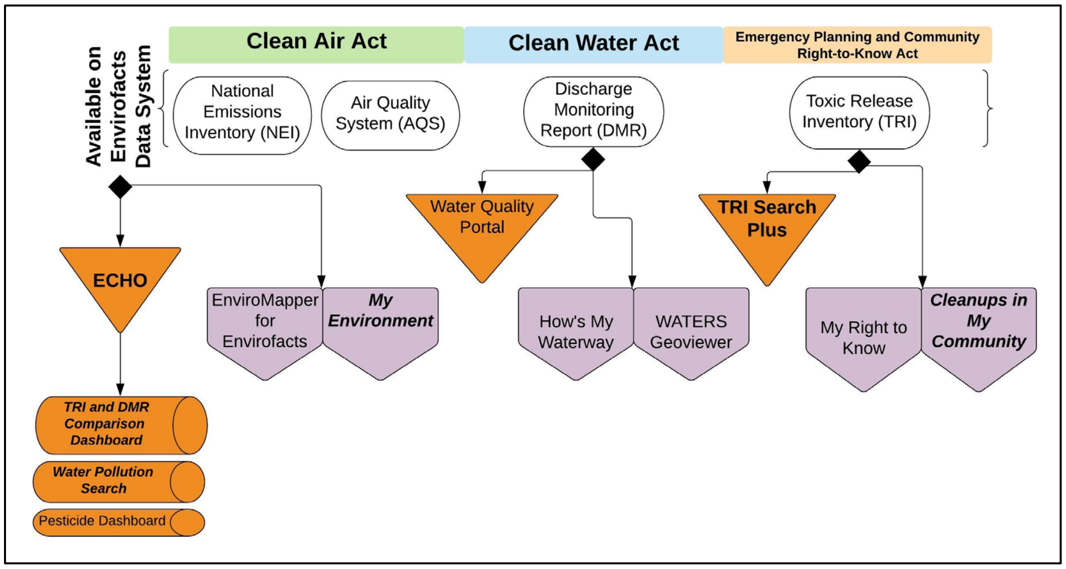

Clean Air Act
Clean Water Act
Emergency Planning and Community Right-to-Know Act
Available on Envirofacts Data System
National Emissions Inventory (NEI)
Air Quality System (AQS)
Discharge Monitoring Report (DMR)
Toxic Release Inventory (TRI)
ECHO
Water Quality Portal
TRI Search Plus
EnviroMapper for Envirofacts
My Environment
How's My Waterway
WATERS Geoviewer
My Right to Know
Cleanups in My Community
TRI and DMR Comparison Dashboard
Water Pollution Search
Pesticide Dashboard

Figure 3. Pollution Dataset and Toolset Overview: Diagram of the datasets that support various EPA-developed tools (orange) and mapping applications (purple). EPA applications discussed in this section are italicized and bolded

Several EPA tools facilitate visualization of their data through the use of graphs or tables, but are not full-fledged web mapping applications because they do not provide an interface for users to interact with data. If they contain a map, their scope is focused on data analysis rather than geographic visualization, and thus the user's mapping capabilities are limited to the viewing of point data. For example, an EPA website called Enforcement and Compliance History Online (ECHO) has over 20 web services for users to explore and download compliance and enforcement information on EPA-regulated facilities. ECHO's tools include the Water Pollution Search and the TRI and DMR Comparison Dashboard. These are useful sub-tools provided by a larger, ECHO-provided toolset called the Pollutant Loading Tool whose purpose is to calculate and report facility pollutant discharges in pounds per year or by monitoring period. The Water Pollution Search tool within the Water Pollutant Loading Tool uses facility and chemical release data from the Discharge Monitoring Report (DMR) and the TRI. These were both integral data sources for this project. After collecting a user query, the Water Pollution Search calculates and reveals top-ten lists of the largest surface water discharges based on total mass and toxicity, indicating the largest impacts on the environment and health. This analytical function is useful, however it contains no geographic reference or display. The TRI and DMR Comparison Dashboard compares wastewater discharge data between the two datasets. Several dashboards support in-depth, side-by-side summaries of the two datasets using pie charts of total releases by chemical or facility and bar graphs representing the number of facilities reporting for each program. Both the TRI and DMR Comparison Dashboard provide flexibility by allowing results to represented in total pounds or in toxic weighted pound equivalents (TWPE). Data may also be downloaded directly from the search results in both tools. This is also a feature of the application created in this project. However, data in both the dashboard and the Water Pollution Search tool can only be viewed on a year to year basis rather than multi-year search bases. This is an additional function supported by the application created in this project, as well as the later-discussed CARB Pollution Mapping Tool.

Another EPA tool called TRI Search Plus is not provided by ECHO or Envirofacts but instead is furnished directly from the TRI program website. It allows users to download TRI data using complex queries. Within the tool are several embedded tabs for users to explore their selected data. By entering an address, watershed, or facility name, the results of TRI data are displayed on a map and users can further filter specific chemicals, years, or industries. Clicking the other tabs then reveals graphs and tables that summarize how TRI facility reporting quantities compare to one another, trends in chemical releases over time, total releases aggregated by the filters selected in the initial query, and other similar analyses. These complex analyses are useful. However, they do not include other datasets that could help assess patterns or provide further meaningful context. The application created in this project integrates TRI data with other datasets.

The EPA has developed two web mapping applications called Cleanups-in-My-Community and MyEnviroment, which enable communities' right-to-know and can facilitate environmental justice. Cleanups-in-My-Community provides a map and tools specifically related to contaminated sites and cleaning efforts (Figure 5) while myEnvironment provides a dashboard interface for the public to explore topics of concern in their region such as air, water, and land use (Figure 6). Data is used from the TRI, DMR, and Superfund programs, among others. Both allow the user to choose a specific location (zip code, county, or city) and then returns summarized datasets with multiple analytical and visualization panels to help assess the condition of environmental pollution in their communities. These objectives in particular overlap with the goals of this application and provide a foundation for the data, concepts, and functionalities to include or be expanded on in the final product. But while pollution-specific data is included in the applications, user interactivity within the mapping portion is limited. For example, in both Cleanups-in-My-Community and myEnvironment, the pollution results are limited by a lack of temporal analysis tools for users to visualize chemical releases over time. Nor are there layers that show the magnitude of chemical releases by facilities. The application developed in the project includes layers reporting pounds released per year and a Time Slider widget.

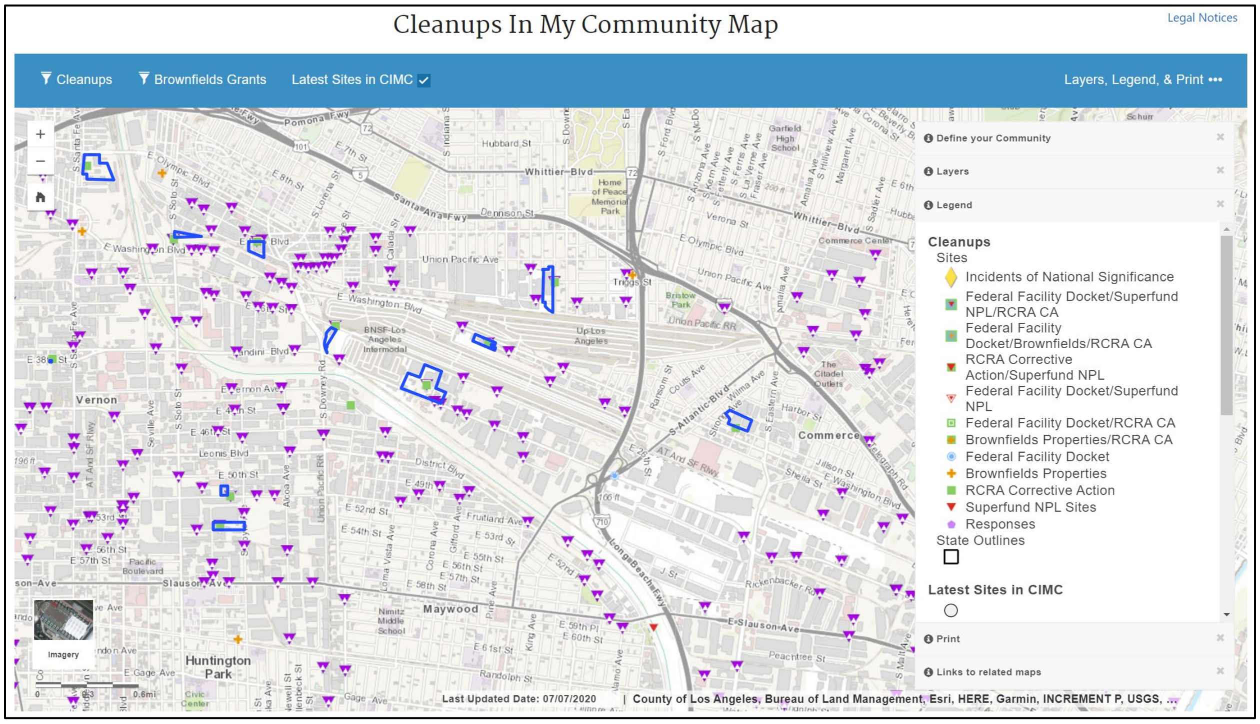
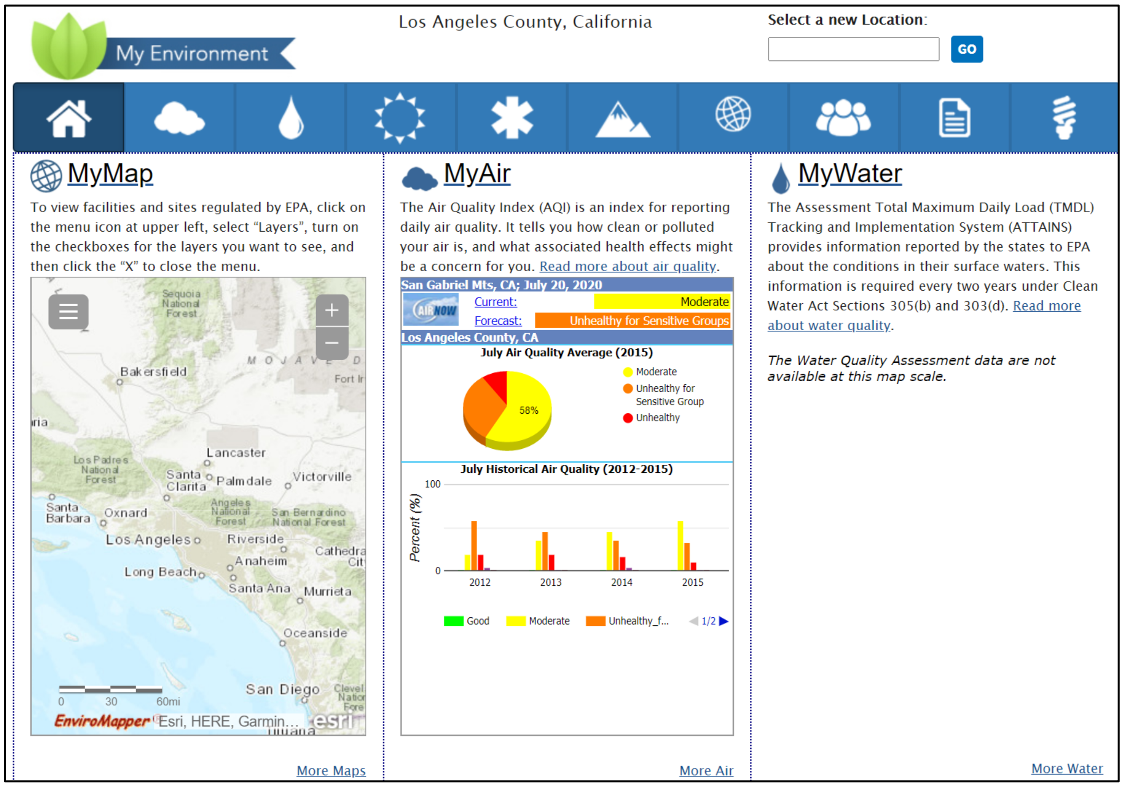

Cleanups In My Community Map
Legal Notices
Cleanups
Brownfields Grants
Latest Sites in CIMC
Layers, Legend, & Print
Define your Community
Layers
Legend
Cleanups
Sites
Incidents of National Significance
Federal Facility Docket/Superfund NPL/RCRA CA
Federal Facility Docket/Brownfields/RCRA CA
RCRA Corrective Action/Superfund NPL
Federal Facility Docket/Superfund NPL
Federal Facility Docket/RCRA CA
Brownfields Properties/RCRA CA
Federal Facility Docket
Brownfields Properties
RCRA Corrective Action
Superfund NPL Sites
Responses
State Outlines
Latest Sites in CIMC
Print
Links to related maps
Imagery
Vernon
Huntington Park
Maywood
Commerce
Last Updated Date: 07/07/2020
County of Los Angeles, Bureau of Land Management, Esri, HERE, Garmin, INCREMENT P, USGS, ...

Figure 4. Cleanups-in-My-Community interface shown mapping cleanup site boundaries (blue polygons) and TRI Facility locations (purple triangles) in Los Angeles County


My Environment

Los Angeles County, California

Select a new Location: GO

MyMap

To view facilities and sites regulated by EPA, click on the menu icon at upper left, select "Layers", turn on the checkboxes for the layers you want to see, and then click the "X" to close the menu.

MyAir

The Air Quality Index (AQI) is an index for reporting daily air quality. It tells you how clean or polluted your air is, and what associated health effects might be a concern for you. Read more about air quality.

San Gabriel Mts, CA; July 20, 2020	
Current:	Moderate
Forecast:	Unhealthy for Sensitive Groups

Los Angeles County, CA

MyWater

The Assessment Total Maximum Daily Load (TMDL) Tracking and Implementation System (ATTAINS) provides information reported by the states to EPA about the conditions in their surface waters. This information is required every two years under Clean Water Act Sections 305(b) and 303(d). Read more about water quality.

The Water Quality Assessment data are not available at this map scale.

More Maps More Air More Water

Figure 5. My Environment interface for exploring environmental-related data for Los Angeles County

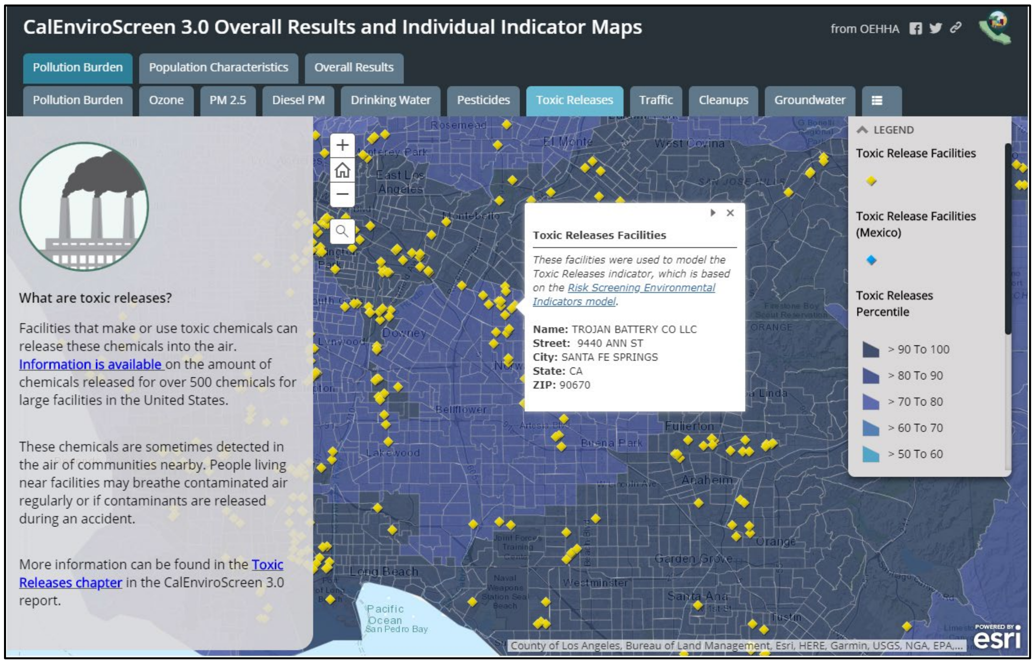
2.2.2. *Collaborative Pollution Mapping*

This section offers an overview of applications that have been developed to visualize pollution data using datasets provided by the EPA. But while the EPA has collaborated with various agencies in the development of these applications, none are directly hosted by the EPA. The three web applications described here are CalEnviroScreen, EnviroAtlas, and the California Air Resources Board (CARB) Air Pollution Tool.

2.2.2.1. CalEnviroScreen

OEHHA created the application CalEnviroScreen at the behest of CalEPA (Figure 7), which focuses on environmental justice by enabling the public to learn how their communities are being affected by pollution. It identifies the communities that are disproportionately burdened through the comparison of population vulnerability and multiple pollution sources. This comparison is represented by the “CalEnviroScreen Score”, which is calculated for each census tract by multiplying a “Pollution Burden” score with a “Population Characteristics” score. Each score is made up of two components: exposures and environmental effects, and average of sensitive populations and socioeconomic factors, respectively. The CalEnviroScreen Scores are represented on a map using a percentile scale between 0-100% to show the census tracts most burdened by environmental pollution. CalEnviroScreen is an application that presents the results from a model that uses specific environmental, toxicity, and demographic data parameters to represent relative pollution among census tracts. The CalEnviroScreen model needed to define adequate indices to be used as variables in the calculation. This required a strict selection of units to use for each dataset. For example, TRI chemical release is an indicator for the Pollution Burden score, which is defined by the chemical releases’ toxic weighted pound equivalent (RSEI hazard score). The toxic weighted pound equivalent is used in this project application.

Some components that Cal ToxTrack develops that are not present in CalEnviroScreen are due to different goals, are additional toolsets such as the ability to query datasets on the same map or to create spatiotemporal representation of data.



What are toxic releases?

Facilities that make or use toxic chemicals can release these chemicals into the air. Information is available on the amount of chemicals released for over 500 chemicals for large facilities in the United States.

These chemicals are sometimes detected in the air of communities nearby. People living near facilities may breathe contaminated air regularly or if contaminants are released during an accident.

More information can be found in the Toxic Releases chapter in the CalEnviroScreen 3.0 report.

Toxic Releases Facilities

These facilities were used to model the Toxic Releases indicator, which is based on the Risk Screening Environmental Indicators model.

Name: TROJAN BATTERY CO LLC
Street: 9440 ANN ST
City: SANTA FE SPRINGS
State: CA
ZIP: 90670

Figure 6. Toxic release map by CalEnviroScreen showing facility locations reporting to the TRI and the relative release quantities with the toxic release percentile symbology

2.2.2.2. EnviroAtlas

EnviroAtlas is an application developed via a collaboration between the EPA and other government agencies. The goal was to provide the interface, data, and tools needed for the analysis of topics related to ecosystem services, chemical and non-chemical stressors, and human health (Figure 8). The user can create maps using a variety of datasets related to environmental services. There are over 400 layers available. These include protected lands, commuting and walkability, employment, building vacancies, political boundaries, ACS demographics, climate projections, and even functionality to import data from a local machine or a URL. While there are many options for contextual data that can be displayed on EnviroAtlas, the datasets related to pollution are limited. For example, as seen with all of the other mapping applications, only TRI facilities can be displayed on the map. These do not contain any information on the chemical(s) released at a particular site or when they were released- rather only the most recent release date is reported. Additionally, while EnviroAtlas utilizes a time slider, the time slider is only available for use on a single climate model data layer. Since none of the pollution datasets are time-enabled for the time slider widget, it is difficult to know when a facility started or stopped reporting, and even more difficult to visualize how the release of specific chemicals have changed over time and space. Cal ToxTrack, developed in this project, makes pollution datasets available, similar to EnviroAtlas, but is allows chemical releases to be displayed, queried, and viewed temporally.

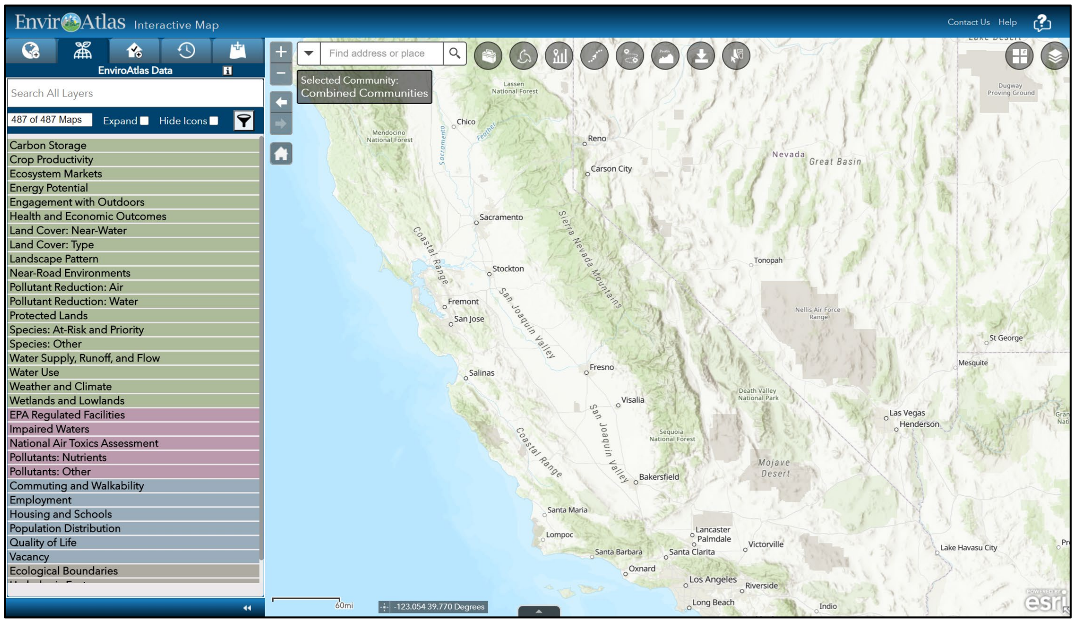
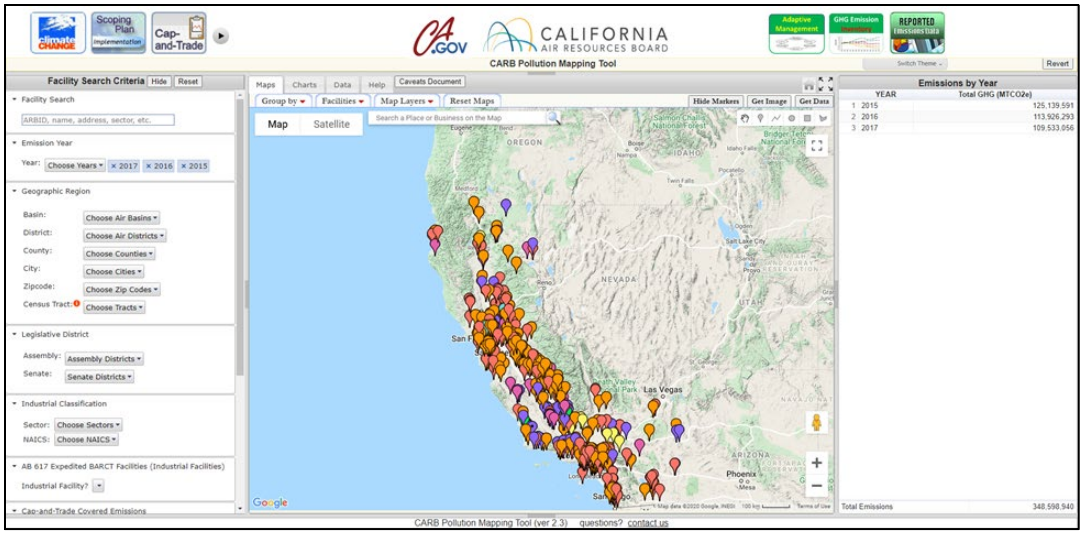


Figure 7. EnviroAtlas interface

2.2.2.3. California Air Resources Board (CARB) Pollution Mapping Tool

The CARB Pollution Mapping Tool allows users to locate, view, and analyze emissions of greenhouse gases, criteria pollutants, and toxic air contaminants from large facilities in California. This application includes a mapping interface that displays facilities reporting emissions, interactive data summary tables, and dynamic data visualization using charts and graphs. Users can choose which facilities are mapped by creating multi-criteria queries that select data to be displayed and explored. This application hosts a highly sophisticated interactive interface. User queries dynamically update panels that summarize the selected data and extend analysis. A “Facility Search Criteria” panel enables a user to query for location, industry classification, air pollutant, etc., while another panel summarizes the selected data. Furthermore, each facility displayed on the map can be clicked to reveal a pop up for further visualization of emission data through bar graphs and charts, which update with the queries. A taskbar at the top of the map allows for further personalization by providing options for the user to display other layers (district boundaries, oil and gas fields, results from the CalEnviroScreen model, etc.) on the map. In another drop-down menu on the taskbar, the user can decide on the grouping of the summary table. If the user chooses to group “By Year”, total emissions are aggregated by the years selected in their query (Figure 9). If they choose to add another year to their query on the Facility Criteria panel, the table will update as will the charts in the facility point pop ups. If they chose to group by industry, the summary table will be updated to aggregate the selected emission data by industry.

The CARB Pollution Mapping application relates to the application developed for this project in that it focuses on user interaction with pollution data. All the functionalities of the application revolve around the mapping of pollution and it provides data from former years.



Year: Choose Years × 2017 × 2016 × 2015

Emissions by Year

	YEAR	Total GHG (MTCO2e)
1	2015	125,139,591
2	2016	113,926,293
3	2017	109,533,056
Total Emissions		348,598,940

Figure 8. CARB Pollution Mapping Tool aggregating emission data for all facilities in California during the years 2015, 2016, and 2017

2.2.3. Summary and Analysis of Application Review

There were many applications discussed in this section, and thus it is evident that the public has multiple opportunities to explore environmental pollution. However, within each application, there are analytical limitations because they either only focus on a specific pollutant source (water pollution or air pollution), which narrows the scope of analysis, or they are so expansive in scope that pollution data is not available for use in tools for temporal pollution analysis (such as in the CARB Pollution Mapping Tool). Meanwhile, other powerful tools and applications like CalEnviroScreen or the Pollutant Loading Tool dashboards have entirely different purposes, showing pollution modeling results or facilitating bulk data downloads, which do not allow users to comprehensively map and analyze pollution data over space and time.

Another limitation, especially within the EPA resource framework, is that there are so many aspects of the available tools relating to pollution that to be able to conduct an analysis, one would need to navigate through multiple taskbars and web services to utilize the relevant tools. For example, the TRI Search Plus tool is not advertised or listed with TRI-related data anywhere within Envirofacts. Under the list of TRI data in Envirofacts, only the basic TRI Search tool is listed, which facilitates simple searches of the TRI dataset. Databases that contextualize a search with other datasets can limit the ability to conduct useful analysis. While the TRI Search Plus tool provides an interface for user-friendly analysis, it is quite difficult to find, as it is not listed as a TRI Search Tool in the Envirofacts system, and can only be accessed through a Google search or through direct navigation to the TRI program home page. Furthermore, while some EPA tools are easily accessible, they come with such a steep learning curve that it defeats the purpose of many web applications, which normally require no training to use. For example, the ECHO toolset is so complex that there is a Help Page that features multiple webinars and tutorial videos that explain how one could use the tool. While this level of sophistication enables particular types of analysis, this project aimed to develop an easy-to-use, aesthetically pleasing interface that provides extensive pollution data that can be compared with the users data of interest.

2.2.4. TOXMAP

TOXMAP provided a simple interface for the public to map their own analyses of toxic chemical data. TOXMAP linked chemical releases to the chemical's toxicology information which was available in a database called PubMeb provided by the NLM (Hochstein et al., 2014). Users could search datasets by location, chemical name, release medium, release amount, facility name and ID, or by selecting a defined geographic region. Search results could be exported to Google Maps and Google Earth. Non-EPA datasets that were used as overlay layers in TOXMAP were U.S. Census population data, income figures from the Bureau of Economic Analysis, and health data from the National Cancer Institute and the National Center for Health Statistics (NLM 2018). A significant difference between TOXMAP and this application is the additional representation of chemical release data; there is an option to represent the chemical releases in units of total pounds released in the toxic weighted pound equivalent (TWPE). As stated by the developers (Hochstein and Szczur 2006), making analytic tools available to users who may not fully understand the implications of the tools or the dataset background may lead to misrepresentation. In this case, some chemicals will be released in larger quantities but are lower in toxicity and less harmful than other toxic chemicals released in smaller quantities. Every application that renders datasets for widespread availability risks the possibility of misrepresentation. This project is different from TOXMAP in that it will not only provide access to databases, but it will address the misrepresentation of toxicity, or valuation of "pollution" by accounting for the toxic weighting factor of each chemical and then including this in the map display. These values, known as the "toxic weighted pounds equivalents" (TWPEs) are not meant to solely assess the impacts to aquatic life or human health. Instead, they are utilized to understand how treatment technologies, individual facility discharges, and industry discharges compare to one another (EPA 2012). While this method does not determine health risk, it still creates a relative scale with which human toxicity of each release can be measured. The scale in the application will, at a minimum, keep the public aware that the release in pounds cannot be used transitively to determine "toxicity". This was a core goal of this thesis project.

2.3. Web GIS and the Community Right-to-Know

This section provides background on the risk of industrial toxic chemical use near communities, the role of the public in historical cases of emergencies, and background on the state of chemical usage in California industry.

2.3.1. Inherent Risk in Industrial Chemical Use

Even with programs in place like the Toxic Release Inventory and stringent monitoring, inherent risks are present with the handling of toxic chemicals. This warrants the need for the public to be aware of the chemicals they are surrounded by and potentially exposed to. Some of these risks include unintentional toxic chemical release, explosions, and fires. This affects onsite workers and the surrounding communities, making transparency crucial. In Morteza et al. (2019), consequence models were compared for hypothetical accidental releases of hydrogen sulfide (H_2S), a colorless, highly toxic gas found in most refineries, which is also on the TRI chemical list. This modeling determined hazard distances (HDs) under 314 possible H_2S release scenarios that varied in release source (pipe or vessel) and volume (based on pipe/vessel diameter, and response time). Results indicated that under a scenario of a 90-second release time from a pipe that ultimately dispels 204,371 kilograms (kg) of H_2S, the distance hazard would be 10,000 meters (m), meaning people within approximately 6 miles of the facility would be exposed to levels of H_2S that pose adverse effects, and thus must have proper access to resources for hazard information (Morteza et al., 2019). These facts represent the need for continuous efforts to enhance public awareness in the realm of toxic chemical use.

2.4. Information on Datasets Used in Cal ToxTrack

The two primary pollution datasets selected for this application were the Toxic Release Inventory (TRI) and the Discharge Monitoring Report (DMR). The TRI provides information for pollution to land and water while the DMR provides all point-source releases to surface waters. Both datasets have strengths and weaknesses that are reviewed in this section to identify the issues this application aims to address and acknowldge those it cannot.

The TRI program requires that companies from select industries annually report the quantity and means of disposal of toxic chemicals used above their threshold value. This data has been collected since 1987. It promotes transparency by covering a wide range of chemical life cycles such as treatment on or off site, accidental spills, injection into the ground, and direct release onto land, water, and air. However, this data also has significant limitations. Of the 767 toxic chemicals on the TRI list, 367 have not been individually tested for their assigned reporting threshold values and have no toxicity information associated with them. Another limitation is that the TRI program excludes small industries like dry cleaning and auto body shops (EPA 2018), despite dry cleaning's history of using perchloroethylene. This chemical has prompted several clean-up initiatives by the government (EPA 1993) and persists in water and soil today. Additionally, TRI is self-reported by facilities and values are deduced using a best-guess estimate, meaning there could be inaccuracies. While these limitations should be taken seriously, it is currently the only dataset that provides the public with insight on chemicals being used in industries.

The DMR has a similar self-reporting system with respect to point-source pollution discharged directly to surface water. This includes conventional pollutants like oil and grease, toxic pollutants, and nonconventional pollutants like nitrogen, phosphorus and even heat. Facilities releasing to surface water must obtain a National Pollution Discharge Elimination System (NPDES) permit which allows them to release a specified amount of a pollutant. Under this permit, facilities are required to have monitoring stations, collect samples, and report their total annual releases to the EPA. One strength of this dataset is that while releases are self-reported and maintained by the facilities themselves, limitations are predefined in the permit and quantitative measurement methods are verified before the issue is permitted. This stricter approach to permit distribution with continuous monitoring provides the public with more accurate results. However, a major limitation of the DMR program is that it only considers point source pollution even though a large source of pollution also comes from nonpoint sources like runoff (EPA 2010).

A review of these applications that provide access to pollution datasets concludes that Cal ToxTrack expands on them by supplying an intuitive interface and additional analysis tools with its time slider and filtering capabilities. Cal ToxTrack also derives similar visualization techniques that enhance user experience, like data point values represented by graduated symbols and the combination of charts and maps to represent the entire dataset. Ultimately, this app compliments existing work in the field by providing another avenue for powerful pollution data exploration.

2.5. User Experience (UX) Design

User experience design must be considered in the development stages of an application to ensure the targeted user needs are accomplished. If user experience design is effective, users will find the application easy to use and will enjoy in interacting with the interface. As Nodder (2013) explains, it is imperative to decide who a website will be designed for by sketching out the potential users' different attributes, goals, values, and concerns. This should be completed before choosing any tool or platform to develop on. Thinking this through will ensure users do not get lost on the site and will continue to use the application.

Cal ToxTrack fullfilled UX planning in the early phase of development with "user cards", which were created by brainstorming various potential users of the application and writing down case stories on index cards including what each of their goals would be, what they would want to accomplish, and what components of the application would be important to them. This ultimately led to the chosen components (time slider, informational modal, and query tool) and their layout and functionality in the application. This was selected based on the conclusion that this application would target users from the general public who are interested in toxic chemical pollution, and researchers in the environmental or public health fields who want to analyze past chemical releases to identify trends.

2.6. Open Source Programming

As more companies begin to embrace and promote the use of open source in the modern tech world (Sharma 2021), this application was developed using entirely open sourced tools with the intent to promote the Open Source Initiative corporation's definition of open source and to maintain access to development tools after graduation. Open source programming is defined as any program that allows access to it's source code, is not distributed for a fee, and does not restrict use to anyone orany field (DFSG 2004). There are many benefits to hosting and using an open source solution. Open source fosters a community among developers, where contributions are made to software and learning is promoted. By sharing source code, other developers can work on solving issues and bugs that haven't been solved already instead of re-creating individual versions of the same software and running into the same problems. A similar benefit in open souce over propietry software is that there are more developers to fix the code instead of the same specific group that works for a company. The more people that have access to developing the software means likely the code was built with more diversity, lower-severity bugs since the code is widely scrutinized, and greater innovation by sheer benefit of numbers. Also, widespead development and use of source code through public distribution means issues can be more rapidly identified and addressed. By installing and using Django for this project, the open source tool was tested, installation count for it increased (showing the contributors support for the tool), and this project serves as a potential use case example for the tool.

Chapter 3 Application Development

Cal ToxTrack was designed to provide users with an interface to explore public datasets on air and water pollution in California. Section 3.1 describes the functionality requirements for the application and reviews possible user queries. Section 3.2 discusses the two main pollution datasets used in the application, the Toxic Release Inventory and Discharge Monitoring Report. Section 3.3 describes the methods used in this thesis based on a common programming architecture known as MVC (Model, View, Controller). The three categories- Model, View, and Controller- provide a universal/uniform design pattern that organizes code and files during the development process. MVC is widely used and incorporated in popular frameworks including Django, Ruby on Rails, and the now-retired ASP.NET. Section 3.4 discusses the user testing methods for the beta application.

3.1. Requirements

The application's objective was to provide users with a clear interface and the tools necessary to spatiotemporally analyze pollution with public datasets. Requirements of the application were:

- A time slider that allows for exploration of toxic chemical release over time
- Access to location services, including a geocoding address bar and controls for zoom, panning, and layer toggling
- The goals, data limitations and functionalities of the application must be clearly defined and understood by the user

- Friendly UX experience, intuitive to use and does not require prior GIS knowledge
- Availability and consistent functionality on any device with access to a browser

The intended users are members of the general public that want to assess toxic chemical release in California over time. More specifically, this group includes students, members of environmental organizations, and engaged citizens. The intuitive interface is designed for a general audience that may not be trained in the field of GIS or environmental justice, but have questions they want to visually answer on the topic including, but not limited to:

- How has toxic chemical usage and release changed over time in California?
- Which toxic chemicals have been released, at what volume, and how toxic are they to the environment and public?
- Are there areas being used for release of the same class of chemical, often caused by industry clustering?
- Are certain communities disproportionately burdened by toxic chemical release location by a common occurrence of releases over several years?

3.2. Data

This is a data-driven application. Thus, the user's understanding of the source, significance, and limitations of the data is crucial for the success of this project. Users need to be aware that raw pollution values should not be taken at face value and considerations should be made as to location, the type of company producing it, the size of the company, why they are releasing the chemical, and any associated history. This issue is addressed in the application through the visualization of data on a map and the inclusion of facility details and industries in the pop ups and data filter. These provide context as to the industry releasing the chemical, the proximity of the release to populous locations, and the size of the release, which can all be explored within the application. For example, the DMR dataset release data also includes an attribute on the toxic weighted pound equivalent (TWPE), since not every chemical is equal in toxicity and the TWPE normalizes chemicals by their toxicity. By including both values, the user can have a more nuanced understanding of the toxicity of each release. Additionally, all other data attributes provided in the application provide information that can be researched to learn more about each release and its context, such as facilities details. It is anticipated that the data used in Cal ToxTrack (Table 1) will initiate further research on pollution in California.

Table 1. Final data sources used in Cal ToxTack

DATA SOURCE	CONTENTS	DATUM	PURPOSE	SCALE
Toxic Release Inventory (TRI) - EPA	Provides facility-reported toxic chemical releases since 1987, including information on the facility, the medium of release, and the quantity.	North American Datum of 1983	California releases were filtered for pollution data in application	USA
Discharge Monitoring Report (DMR) - EPA	Provides locations of point source discharges into U.S. waterways	North American Datum of 1983	Add data to map that covers all releases into surface waters	USA

As described in Section 2.4, there are several limitations with both datasets, including self-reporting, a lack of risk assessment for every chemical on the program's toxic chemical list, and possible misrepresentation of raw data based due to the varying toxicity of various chemicals.

While not all limitations can be ameliorated via this application (such as the self-reporting), one limitation that this project addressed was the misrepresentation of risk from emissions when reviewing raw data out of context. Releases that are identified as high risk based on the quantity and the toxicity of data may appear harmful, when in reality they may be less of a risk to the public because of their location and distant proximity to populations. When raw numbers are used to determine risk without geographic context, pressure may be unfairly placed on companies and ultimately result in more harm than good (Neuman 2009). Conversely, smaller quantities of pollution may be overlooked. Using a GIS application to analyze the data will provide the opportunity for a fuller, more accurate risk assessment. For example, Cal ToxTrack facilitates a more accurate risk assessment by providing abundant and specific attribute data with pollution points, like facility name, facility address, chemical release in pounds and TWPE. For further context, this attribute information can then be layered with a satellite basemap to determine what neighborhood proximity is to more or less toxic chemicals.

3.3. Development Methods

This application, Cal ToxTrack, relies entirely on open source solutions, despite the existence of products like Esri's Web App Builder that can create an engaging interface without requiring the developer to write the entire codebase. Cal ToxTrack, having been developed on the server side and the client side, is what is known as a "full-stack" application. Full stack applications manage and host data on servers and also manage front end user interaction with the data once data are passed to the client. The decision to develop the application without "out-of-the-box" tools and instead with open source tools serves the author's goals threefold. First, the decision provides more flexibility in the application's functionality and styling since customization is not limited to what another developer has already coded. Secondly, it ensures free and continued access during any stage of the application's existence by giving the author the ability to use the same development tools post-graduation when access to paid resources may be limited. And finally, as the GIS web development field continues to expand, requests for GIS developers with server-side experience who can work with open source solutions or build GIS software from scratch, in addition to client side experience, is increasing. While the application may appear relatively simple when compared to applications made with "out of the box" tools like Esri's WebApp builder, it represents a more sophisticated skill set that is sought out by employers.

The tools chosen for Cal ToxTrack development are as follows: data was managed and maintained in the open source database system, namely PostgreSQL with a PostGIS extension that extends the PostgreSQL database to support spatial data. The back end of the application was facilitated by Django, an open source Python framework which provides development protocols, code, and structure (defined by MVC). Django connects to the PostgreSQL database that hosts the data, manages the data using models and Python classes, and creates APIs to provide data to the front end of the application. The front end development depends on another open source framework called Leaflet. Leaflet is Open Geospatial Consortium (OGC) compliant, supporting Web Map Service (WMS) protocol layers, GeoJSON layers, and vector layers. Section 3.2.1 discusses how the data was used in conjunction with these open-source tools, including their application, curation, and limitations. Section 3.2.2 discusses the full-stack development process using MVC and includes sections discussing the Model, Controller, and View components of this application.

3.3.1. Database Creation and Management

Both the DMR and TRI data are available to the public on the EPA website in CSV format and were used in Cal ToxTrack to provide data on air and water pollution. While the EPA website provides an API service that hosts the two datasets, the author wanted to have the datasets downloaded locally to ensure consistent access to the information. If the API service were somehow rescinded from the web, Cal ToxTrack could still be of use to the public with the legacy data, though updating it appropriately would prove challenging. A unique analytical strength of this application is its ability to analyze historic toxic release data. Even if the programs under EPCRA were removed and no new data were supplied each year to the public, having record of historic pollution datasets through this application could still be of assistance. For this reason, the data were downloaded and hosted individually in this project through an API service created using Django.

Both datasets contain integral information on facility location, facility details, pollutant name, release quantity, and year. These table attributes were first prepared in R to follow the data schema (Figure 10) for the PostgreSQL database. An integral component of the application is a time slider that represents data on the map according to the year they were reported. For this reason, each table within the database must have a column that indicates which year the input row is from. The original DMR data are provided as separate CSV files for each year, but these do not include a year column, which was one of the tasks completed using R. Another column was added to to TRI dataset which specified the medium by which the toxic chemical was released in, since the TRI was the data source for both air and water pollution

The platform of R was selected because the table sizes were too large to be reasonably managed in a program like Excel. The TRI dataset has been published annually since 1987, and includes over 16,000 rows of data. For this reason, R was used on the original CSV files to edit column names, filter null values, append each year's table to one large table, and export the final tables as CSV files ready to be imported into the database. To match the schema, a total of four tables were exported to CSV files: TRI facilities, TRI releases, DMR facilities, and DMR releases.

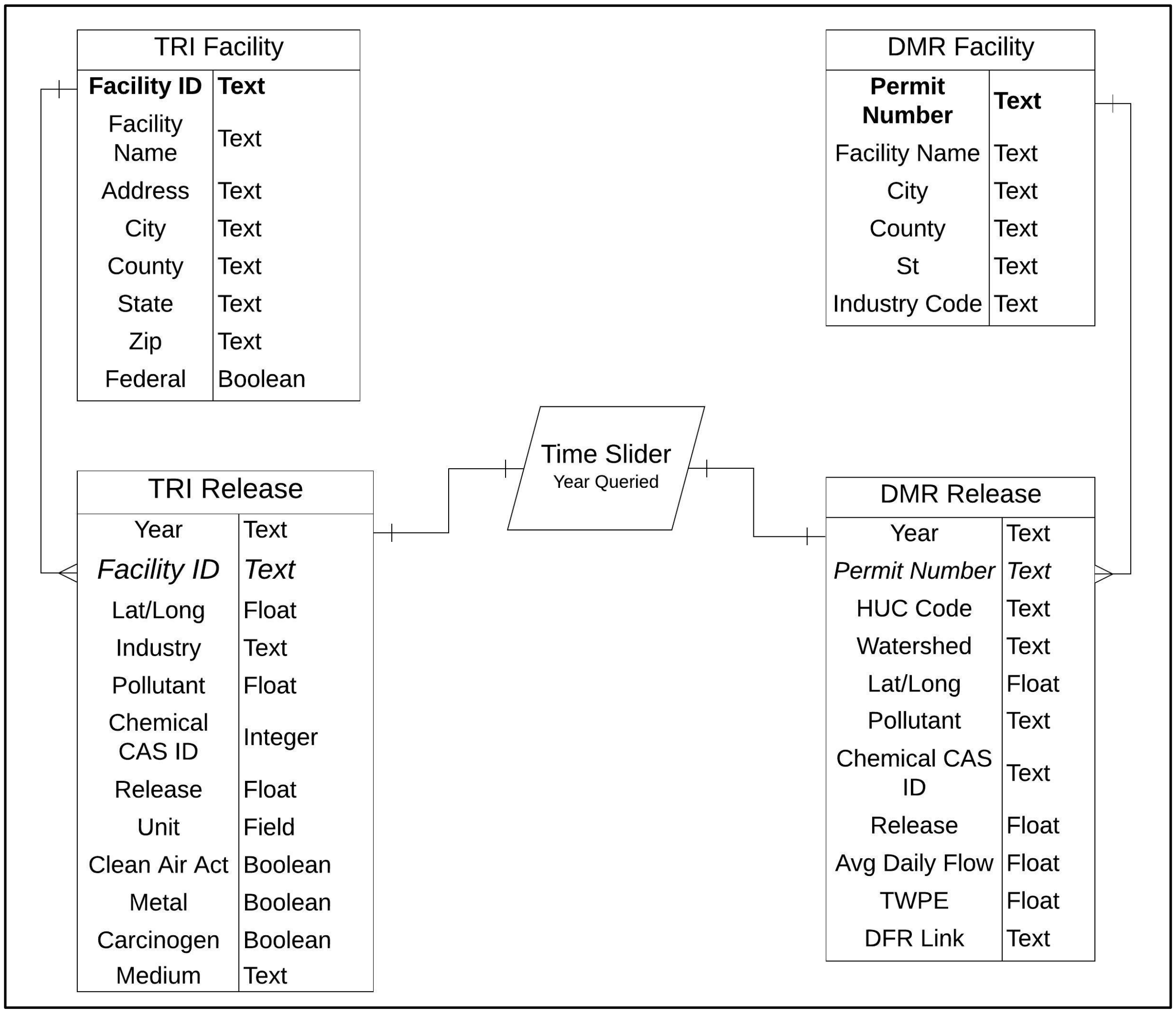


Figure 9. Entity-Relationship diagram representing pollution database schema and how it relates to the time slider component

The next step of the process was to populate the PostgreSQL database with the table. The DMR and TRI facility tables do not contain the geographic information to be used for the application, so they were added to the PostgreSQL database, called *thesis_db*, conventionally through commands as shown in this psql command line example:

```
C: > psql -U postgres -d thesis_db
```

```
>
>  CREATE TABLE tri_facility (
>  f_id VARCHAR(255) PRIMARY KEY,
>  frs_id CHAR(12),
>  f_name VARCHAR(255),
>  address VARCHAR(255),
>  city VARCHAR(255),
>  county VARCHAR(255),
>  st CHAR(2),
>  zip VARCHAR(10),
>  federal BOOLEAN
>  );



>  \COPY tri_facility
   FROM 'C:\tri_facility.csv'
   DELIMITER ','
   CSV HEADER;
```

In order to input the TRI and DMR release tables, which contain spatial columns, they first needed to be converted to shapefiles so they could be imported into PostgreSQL with their associated spatial attribute data, such as projection. This task was approached using graphical user interfaces. FME Workbench, a data integration platform from Safe Software, was used to convert the two release tables to shapefiles. These shapefiles were then added to the database through the PostGIS shapefile tool, which inserts spatial data into PostgreSQL databases. Within the tool, the connection settings were set to connect to the “thesis_db” database and the options were set to automatically generate spatial indexes after import. The specified datum was matched with the datum the original dataset was collected under, the North American Datum 1983 (NAD83) with a Spatial Reference Identifier (SRID) of 4296.

3.3.2. *Back End and Front End Development*

After the data were collected and cleaned and the database created, the next stage of development focused on creating a back end that provided data to the application and a front end that generated the user interface. This was designed in Cal ToxTrack by following the MVC architecture via the Django framework. MVC's design pattern is separated into the components Model, View, and, Controller (Pop and Altar 2014), which explain the steps that control the application's ability to be "full-stack" in handling client requests, communicating with the database to acquire the necessary resources, providing responses to the client, and creating templates for the interface (Figure 11). The Model component encompasses data logic (real-life objects and representations) which are held and managed in the PostGIS database but handled and manipulated in the framework as "models" or Python "classes". The View component functions to facilitate user experience and incorporate front end development with templates that use web design logic with HTML, CSS, and JavaScript. The View for Cal ToxTrack also implements a JavaScript mapping framework that provides geospatial JavaScript libraries to build code for components of the application like basemaps, layers, zoom controls, and other minimal functionalities. Finally, the Controller facilitates interactions between the Model and View. The Django framework enforces MVC through the collections of Python libraries that support the architecture for each MVC component.

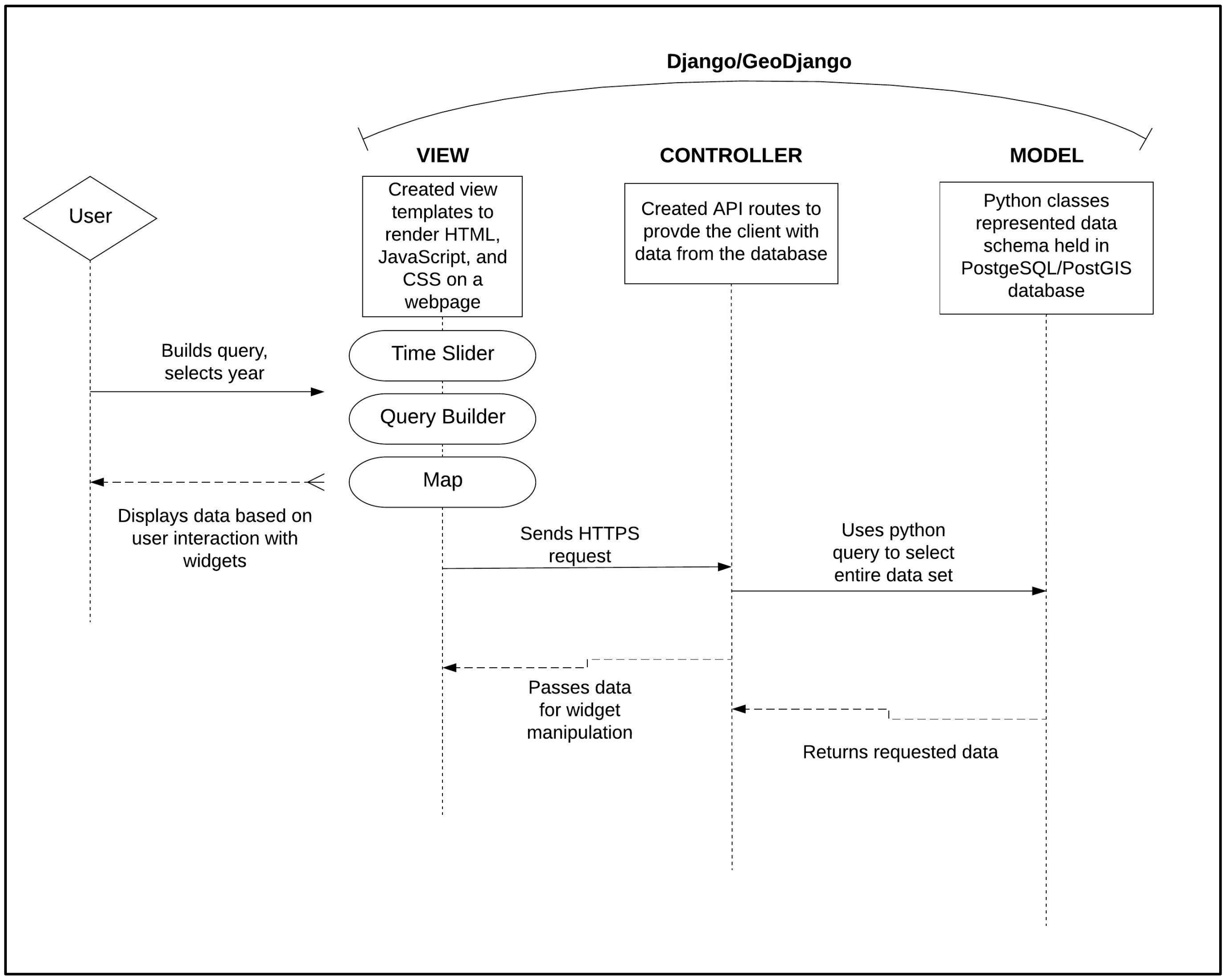
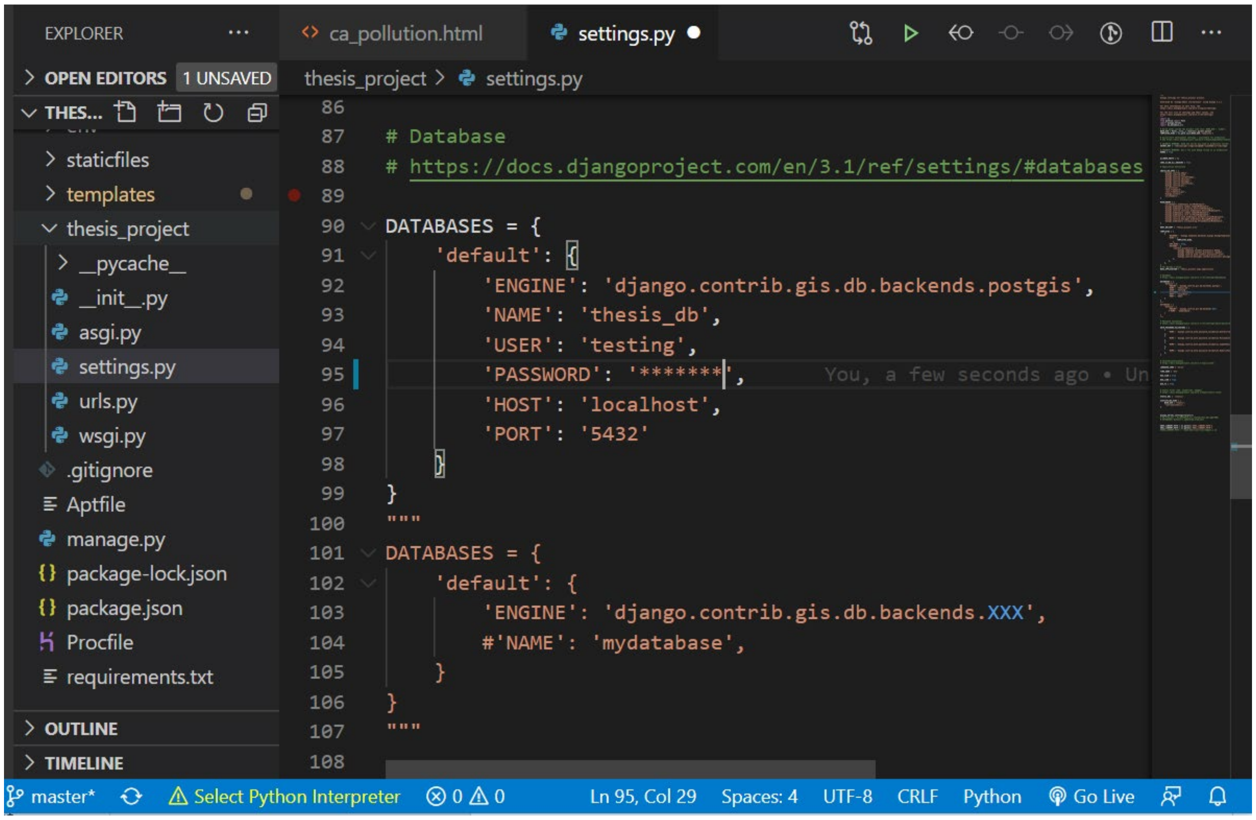

Django/GeoDjango
VIEW
CONTROLLER
MODEL
User
Created view templates to render HTML, JavaScript, and CSS on a webpage
Created API routes to provde the client with data from the database
Python classes represented data schema held in PostgeSQL/PostGIS database
Builds query, selects year
Time Slider
Query Builder
Map
Displays data based on user interaction with widgets
Sends HTTPS request
Uses python query to select entire data set
Passes data for widget manipulation
Returns requested data

Figure 10. Sequence diagram of application MVC workflow and open source tools to develop each component

3.3.2.1. Model Component

The database was created using SQL and the PostGIS interface instead of being hard coded as models in Django. Thus the database was treated as “legacy data” that needed to be integrated in the Django Model framework. Once database settings were configured to link Django to the PostgreSQL database (Figure 12), a simple command was executed to import the table schema from the database as Python classes:

```
> py manage.py inspectdb > pollution_app\models.py`
```



```python

# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'thesis_db',
        'USER': 'testing',
        'PASSWORD': '*******',
        'HOST': 'localhost',
        'PORT': '5432'
    }
}
"""
DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.XXX',
        #'NAME': 'mydatabase',
    }
}
"""

```

Figure 11. Database credentials configured in Django settings folder

Since the tables are represented as GeoDjango objects, Python functions and libraries can be used to manipulate the data. This includes general filtering and also preparation to be passed to the front end web application. Since the two most common data formats passed in HTTP are JSON and XML, the data needed to be serialized, or converted, from the GeoDjango model to a GeoJSON file before being passed to the front end by the Controller.

To maintain a reasonable rendering speed for the application, the number of HTTP requests and responses was reduced by leaving the responsibility for filtering functionality to the front end using Leaflet and JavaScript. This eliminated the need to process HTTP communications between the client and back end every time the user requested different data in the query tool or time slider. Instead, the entire dataset is requested from the back end, cached in the browser as a JavaScript object, then selected data are rendered according to user interaction with the interface. Therefore, the entire dataset needed to be generated in an HTTP-friendly data format and passed from the back end. The first step to this involved selecting all objects from the Django TRI and DMR models (which represents all "rows" in the tables) by using a Python function on the GeoDjango class models called *TriRelease* and *DmrRelease*:

```
>tri_releases = TriRelease.objects.all()

>dmr_releases = DmrRelease.objects.all()
```

All data were used in the environment for the application. The data from the GeoDjango models were serialized to GeoJSON files using a custom-built Python function that was then applied to the *tri_releases* and *dmr_releases* Python objects (Appendix A). Once serialized, the GeoJSON file was indented for readability with the code:

```
> new_tri_json = json.loads(data)
```

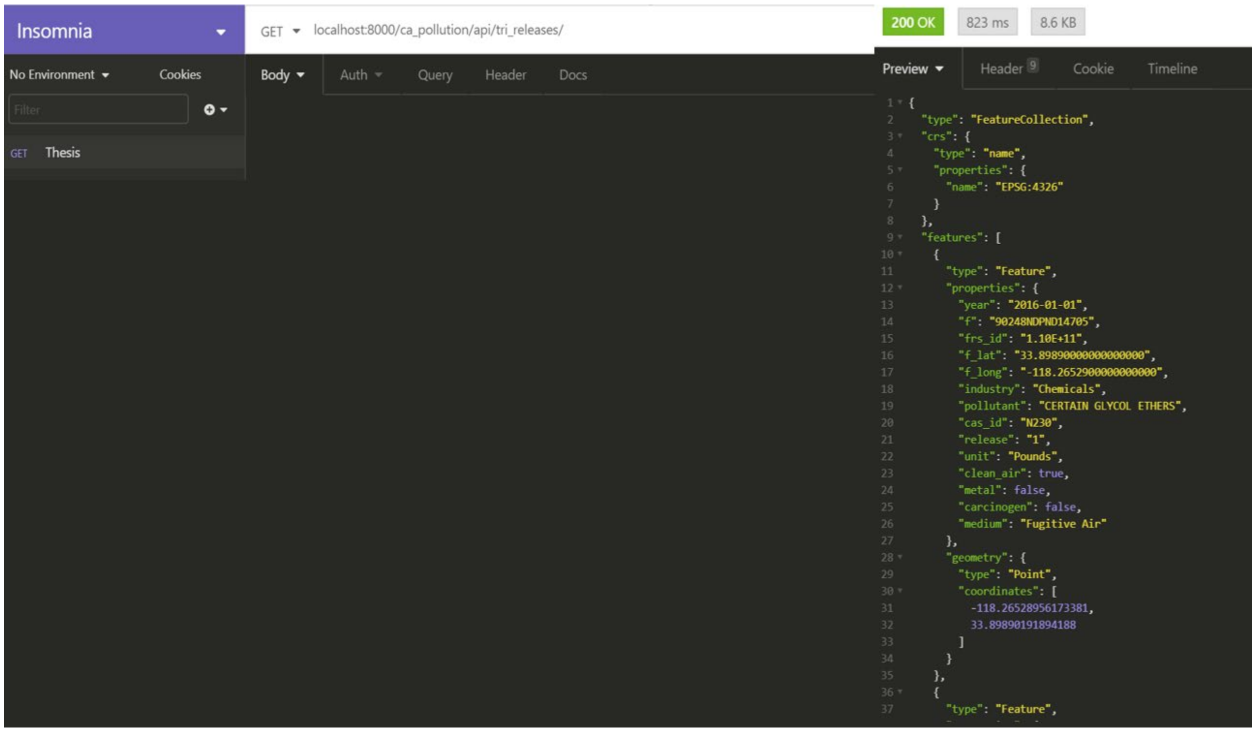
Keeping the data contained in a spatial database and then serializing data in the Django Model component logic allows future application updates to be more feasible than if the data were stored in a static GeoJSON file. Since the pollution datasets are provided on a yearly basis, the application back end needs to support additions of up to thousands of data rows every year. Managing this via a database allows the developer to easily append new data and additionally create SQL queries, relations, and table joins which makes the current and future data much more manageable.

The TRI contains 16,000 rows at 23.2 MB and the DMR dataset is 54,931 rows, for a total size of 9.76 MB. When the entire TRI dataset is loaded onto the map directly from the API it takes about 9 minutes for the data to display. This considered unacceptable for web applications. As a temporary solution while the larger speed issue continues to be investigated, the API model was customized according to stage of the application. During the development stage, all data from the data tables were selected by the controller to render in the application. However, when the application was deployed, the API was customized to only select 50 rows of data for each year between 1987 to 2018 and a Python function automatically wrote the subset of data from each year to a static JSON file. This static JSON file is where the deployed application pulls data from for the time being. By selecting the first 50 rows of the TRI dataset, only water releases were selected for this application. However, the API was completely developed and coded to be available to select all data (instead of the subsets) once the speed issue is resolved.

3.3.2.2. Controller Component and API Creation

The Controller component is the glue between the "front end" and "back end" of the web application. This was accomplished in Django for Cal ToxTrack by creating an application programming interface (API) to act as the software intermediary that passes data from back ends to clients. In Django, the API is created by first defining models with data in the *models.py* file and then generating a *urls.py* file that defines routes for the user to specify the data they want. The responsibility of the *url.py* file is to grab the correct View files according to the route constructed by user interaction with the interface. It then integrates any Model component logic and finally, it shows the user the View.

Another tool, Insomnia, was utilized to test the functionality of the API before attempting to render the GeoJSON in the web browser (Figure 13). Insomnia allows developers to quickly send requests to their back end and returns responses as a browser would. This tool was incredibly useful for debugging the Controller before website deployment.



```
{
  "type": "FeatureCollection",
  "crs": {
    "type": "name",
    "properties": {
      "name": "EPSG:4326"
    }
  },
  "features": [
    {
      "type": "Feature",
      "properties": {
        "year": "2016-01-01",
        "f": "90248NDPND14705",
        "frs_id": "1.10E+11",
        "f_lat": "33.89890000000000000",
        "f_long": "-118.26529000000000000",
        "industry": "Chemicals",
        "pollutant": "CERTAIN GLYCOL ETHERS",
        "cas_id": "N230",
        "release": "1",
        "unit": "Pounds",
        "clean_air": true,
        "metal": false,
        "carcinogen": false,
        "medium": "Fugitive Air"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          -118.26528956173381,
          33.89890191894188
        ]
      }
    },
    {
      "type": "Feature",
```

Figure 12. Insomnia, an API testing tool used throughout development to make JSON requests for the API created in Django using URLs defined in the Django urls.py file. Url in screenshot reads: localhost:3000/ca_pollution/api/tri_releases/

3.3.2.3. View Component

The View component consisted of the most code for this project. An HTML template file was created, and the Leaflet JavaScript framework was imported into the HTML file. The Leaflet framework provided JavaScript and CSS that could be used to generate some geospatial components of the application. However, other components were coded with plain HTML and JavaScript, otherwise known as Vanilla JavaScript, or other APIs because the components were either not geospatial or because they required customizations not already provided by Leaflet. Appendix B contains the full source code used to create each component of the project.

The major components created with Leaflet JavaScript were the map container, the geoJSON to marker layer, map controls (zoom in/out, a dynamic scale, and a legend), and a basemap layer control. In Leaflet, basemaps can be sourced from any tiling service and added to the map container. This is accomplished by 1) passing the source URL with any associated API keys or parameters to the Leaflet tile layer JavaScript function, 2) defining a label for the new basemap variable, and 3) including the basemap variable in the Leaflet function that initially creates the map view:

```
var terrain = L.tileLayer('https://stamen-tiles-{s}.a.ssl.fastly.net/terrain/{z}/{x}/{y}.jpg'
var baseMaps = {“Terrain”: terrain};
var myMap = L.map(‘map’, {center: [35.019073,-118.619628], zoom: 4, layers: [terrain]});
```

Components that were created using Vanilla JavaScript and plain HTML were a query filter left-sidebar, an informational popup modal, and a right-sidebar with the Table of Contents layer switch control. The query filter can currently filter data on the map for industry type and the Table of Contents can toggle specific layers the user wants to explore. The informational modal is coded to display automatically upon the web page loading, which ensures the user views it at least once, however, there is a “Welcome Info” button in the application that reinitializes the modal in case the user needs to view it again. This modal contains information on what the application can accomplish, the date the datasets were last updated, and an update on the application’s latest development status.

The two sidebars and modal are not components covered in Leaflet. However, while Leaflet does integrate an overlay layer toggle control with the basemap control, this control in Leaflet is too small and not customizable, which hinders some requirements for this application. The overlay layer control needs to be clearly visible and each layer will in future work need to include a button prompting an informational popup about the layer’s data source. This type of customization required that these components be coded from scratch.

An external API was used to create the histogram chart on the right panel. Google Charts API can seamlessly integrate with web applications and is extremely powerful. Thus, it was the chosen tool to help visualize summaries of the pollution data. First, the API source was linked into the main HTML page in the head tag so the code could be accessed. Then, to create a chart, JSN data was passed to the Google Chart histogram function, where parameters like bin count, chart color, axis labels, etc. were also defined. This created a new histogram chart variable. To actually see this chart in the application, the new histogram variable needed to be called in the web page's HTML where it should be displayed, which in Cal ToxTrack's case was in the right sidebar panel. Thus, an HTML div element was created for the histogram chart as shown below:

```
<div id="chart_div"></div>
```

The developer wanted only data being shown on the map to be summarized in the histogram chart. This was accomplished by constructing JavaScript that monitored any time slider input changes. When the input changed, the chart creation script was triggered with new and relevant JSON data for the selected year, updating the cart any time the time slider input changed. Thus the user can see the relevant data in the histogram chart.

3.4. Application Evaluation Process

The application was developed for the general public, and volunteers were invited to participate in testing the application specifications. The sample (*n* = *7*) was a non-random convenience sample and was not intended to be representative of the general population. Potential participants were contacted via email with a Google survey that included a link to the application. They were selected by the researcher based on expert judgement as to who would be most likely to use the application, namely individuals in the health, government, information technology, and environmental fields. 12 individuals were contacted by the primary researcher and provided with background information for the application and its purpose.

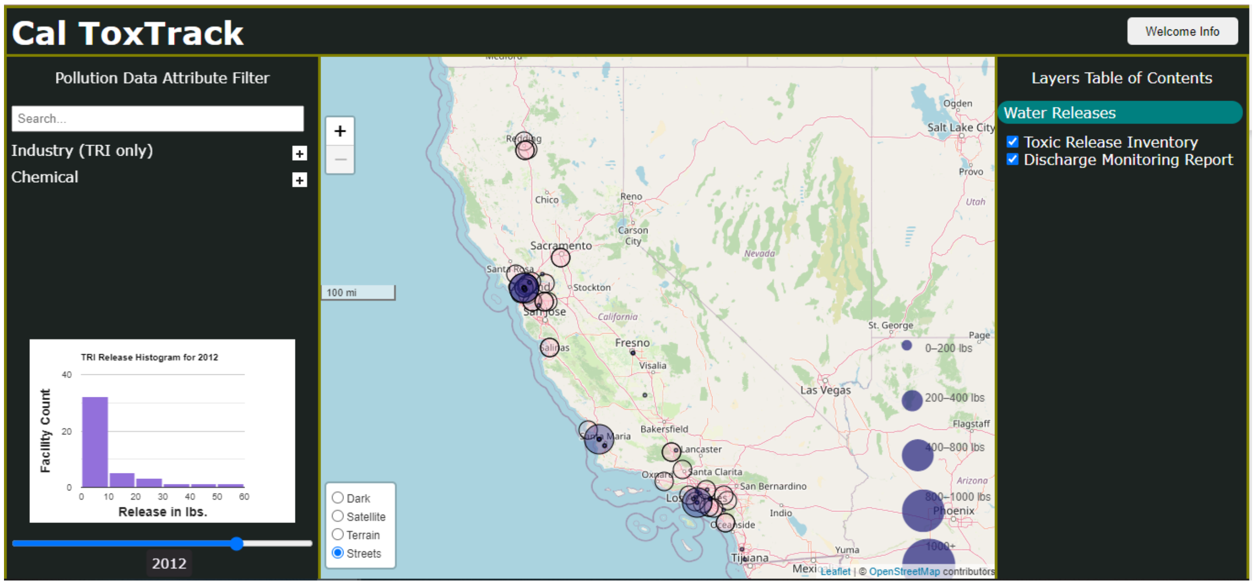
The survey was approved by IRB December 10, 2020. The introduction included guidelines on how to explore data in Cal ToxTrack. It clearly stated that participation was voluntary and that the user could stop participation at any stage in the process. All questions were optional to answer since some questions could be interpreted as sensitive, such as political party identification. The full survey is included in Appendix B.

Chapter 4 Results

This chapter discusses the Cal ToxTrack application that was designed for users to geospatially explore public pollution datasets in California. Cal ToxTrack consists of one webpage that serves as a landing page for users to filter, pan, and analyze charts. This is facilitated with various application specifications summarized in Section 4.1. Section 4.2 discusses the results of user surveys that were conducted in October 2020, emphasizing the survey responses that will help future development of the application.

4.1. Application Specifications

The layout of Cal ToxTrack consists of a map contained within two side panels (Figure 14). The map includes a simple legend, basic controls for zooming and toggling basemaps, and a dynamic scale in imperial units. Analytical tools are included in the left and right panels to facilitate user interaction. This web application's analytical tools consist of a time slider, a basemap selector, a histogram chart, and a filter.



☑ Toxic Release Inventory
☑ Discharge Monitoring Report
[✓ Streets]
2012

Figure 13. Cal ToxTrack layout

4.1.1. Map Container

The Leaflet basemap switch control was kept separate from the Table of Contents and is available through a small, pinned container on the map. By separating the overlay layers from basemap layers into different controls, the basemaps and pollution layers are distinguishable as they both address different areas of concern – the overlay layers provide analysis and the basemap layers provide context.

The basemap sources for Cal ToxTrack were OpenStreetMap, Stamen, and Mapbox. The four available layers are Dark, Satellite, Terrain, and Streets (Figure 15). The default base layer was designated at Streets.

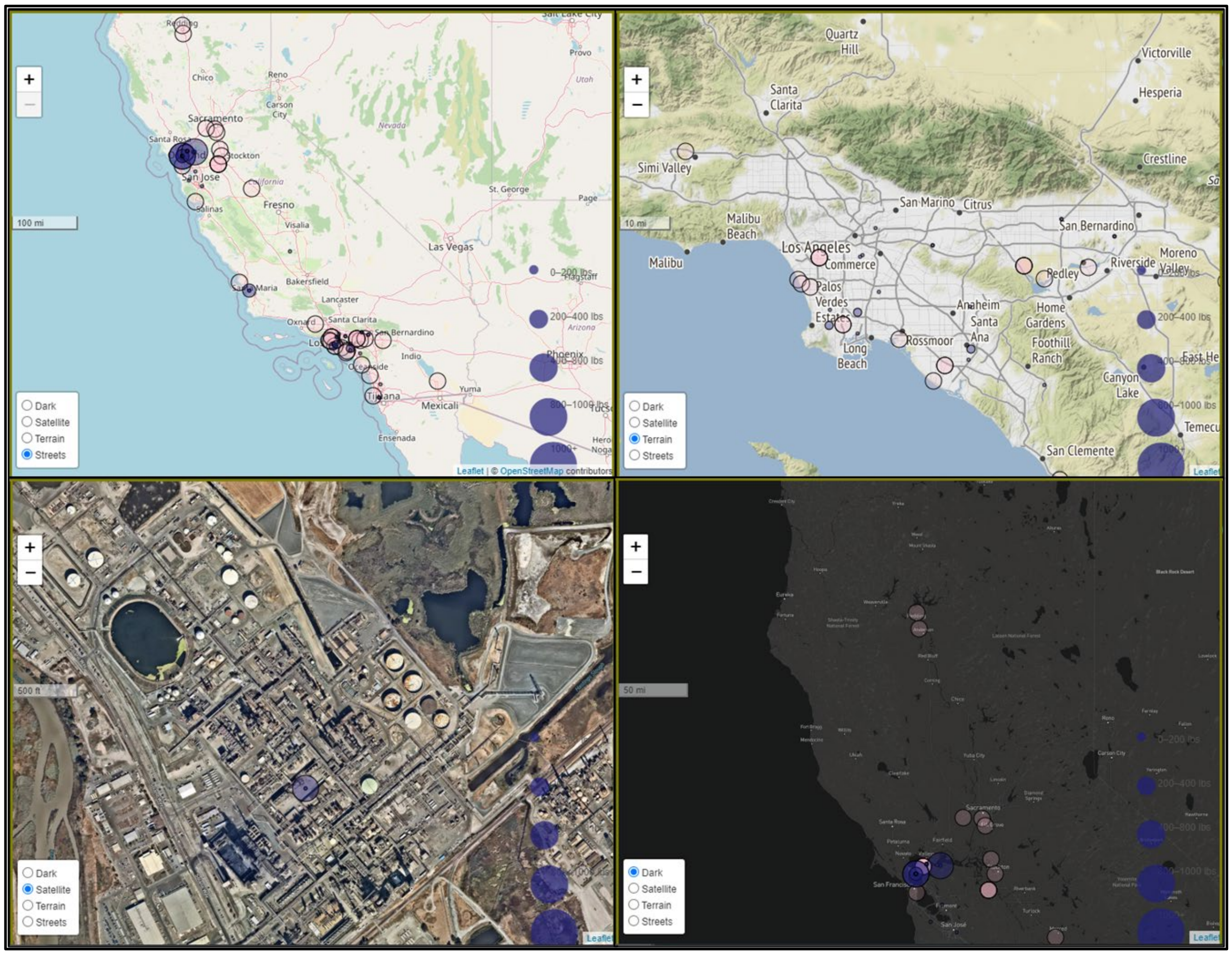


Figure 14. The four different basemap options: Streets, Terrain, Sattellite, Dark (left to right)

4.1.2. Right Sidebar

The right sidebar contains the Layers Table of Contents (Figure 16). Currently, the functionality in this tool is that when the checkboxes are toggled, a JavaScript function either removes the layer from the map, or adds the layer to the map depending on what it was before the checkbox input changed. The data currently only shows toxic chemical releases into waterways, which is represented in the Table of Contents through the Water Releases tab label.

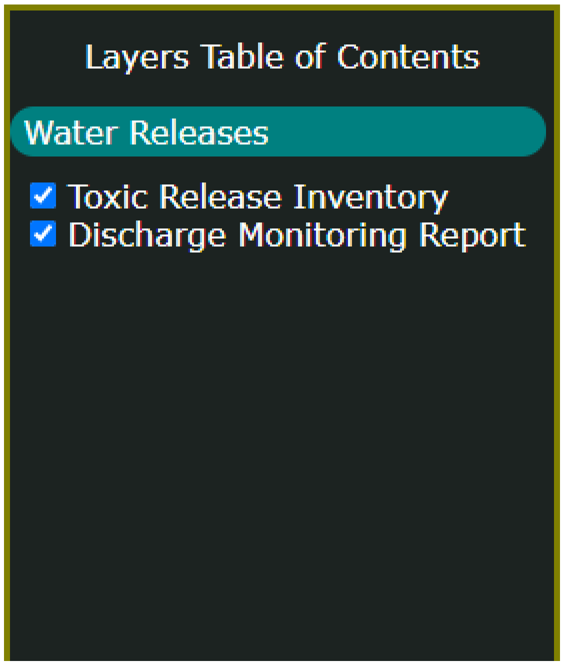


Figure 15. Table of Contents embedded in the right sidebar panel

4.1.3. Left Sidebar

The left sidebar panel includes the query filtering tool, the histogram chart, and the time slider tool (Figure 17). The histogram chart and time slider are included together in one HTML element because they are strongly linked to each other. Both the tools are dynamic. Switching the time slider input not only updates the data shown on the map, but also the data summarized in the histogram chart. All the components contained in the left sidebar panel are described in subsections below.

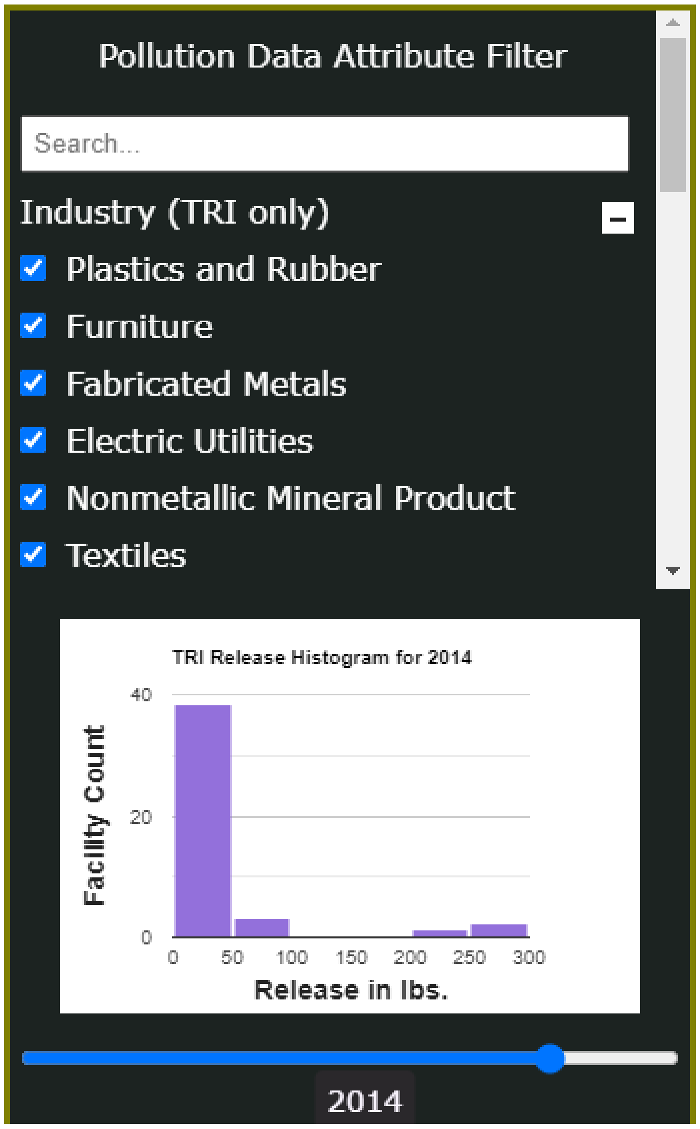


Figure 16. Query dashboard for industry filtering, a histogram chart, and a time slider embedded in the left sidebar panel

4.1.3.1. Query Dashboard

Every time a layer is selected, the data are represented by year with the time slider and are shown on the map accordingly. However, the user can further filter the data by using criteria they define in a query dashboard. The purpose of separating the query constructor dashboard from the table of contents widget is to maintain interface organization and maximum filtering flexibility. Maintaining flexibility means the user can filter the dataset using any field in the associated attribute tables to build their criteria. But as represented in Figure 11, some datasets have at least five fields which would clutter the table of contents with objects if not separated. Having a separate dashboard provides an uncluttered interface for the user to construct their queries.

4.1.3.2. Time Slider and Histogram Chart

A time slider can be used to select or change the year of the data displayed on the map. The time slider default is set to the most recent year of data available when the application is initialized. It can slide to different years to dynamically update the layers selected in the Table of Contents. This provides historical context for the pollution in California and facilitates exploration of data that extends beyond the most recent reporting year. Each time the year is switched, a new query is built. The time slider is essentially replacing the category "year" in the query dashboard to instead be an interactive, aesthetically pleasing, and more dynamic query tool. A user chooses a query via the dashboard and via the time slider in any order. If they select a year before setting their search criteria, all the data will be displayed on the map for that year. Conversely, if a user has already selected their query and then toggles to a new year on the time slider, the entire query will be sent again to the data server and the only difference in the request is a different year. Then each layer is presented by the year selected.

An interactive and dynamic histogram is included on one of the panels summarizes the data symbolized on the map. This enables users to see the raw data in addition to it being visualized on the map.

Summary statistics were calculated for the data TRI set. According to the SQL functions MAX, MIN, AVG, and STDDEV_SAMP, the TRI release dataset has a maximum of 3,687,029,618 pounds, a minimum of 0.0000001 pounds, an average of 29,876.08 pounds, and a dtandard deviation of 9,034,453.39. Because the values range so widely, visualization is easier when the exact amounts are based on individual years. The x-axis of the histogram represents the total pounds released for the selected year, while the y-axis represents the total count of facilities.

Outliers can skew summary results, so histogram buckets were created using a 15% last bucket percentile. This changes the bucket computations to ignore the values that are higher or lower than the percent specified. These values are still included in the histogram chart itself, but this method is meant to prevent outliers from heavily skewing the histogram bucket size calculations.

The histogram chart is updated synchronously with the time slider. The user can hover over the histogram bars to see the total count of facilities, categorized by the amount released in pounds (Figure 18).

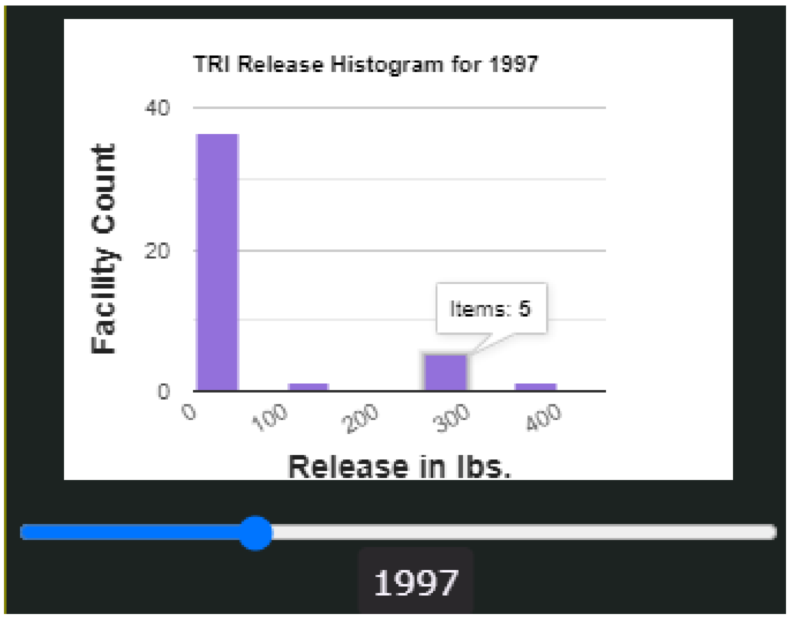


Figure 17. Count of facilities that released between 250 to 300 total pounds of toxic chemicals in 1997

4.2. Application Evaluation Results

Surveys are a fast way to assess the effectiveness of an application in achieving its objectives. This section summarizes the results of the survey question items that are included in Appendix B. The sample was a non-random convenience sample. Emails were sent to 10 individuals during the week of October 12th, 2020 and the survey remained open for two weeks, closing October 26th. Of the 10 individuals contacted, 7 tested the application and completed the survey. Since all questions were optional in an effort to keep the survey as inviting as possible, the results don't represent the entire sample of respondents.

Of the respondents, 43% were male and 57% were female. The ages of the respondents were 72% between 25 to 34 years old, 14% between 35 to 44 years old, and 14% between 19 to 24 years old. Respondents were also asked to provide what field of work they identified themselves with. The pie chart below represents the fields responded.

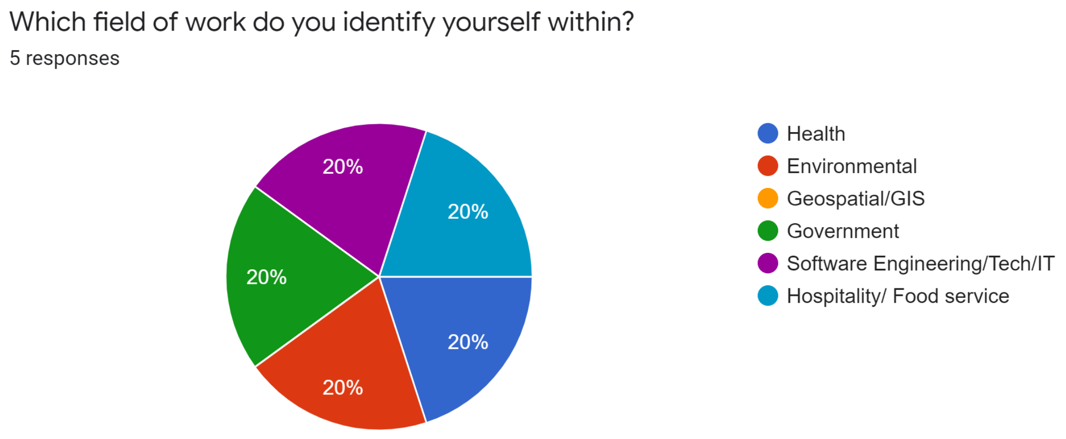


Figure 18. Pie chart representing work fields the respondents identified with

The first question of the survey asked users if they understood the goal of the application. Web applications should be intuitive to use and require minimal effort by the user to understand the goals it attempts to accomplish. This clarity was attempted in Cal ToxTrack with a modal that was built with JavaScript to pop up automatically anytime the HTML body loaded, so that each time the user opened the application, the modal popped up and described what the application was, what data it used, and what users could accomplish. This survey question first re-stated the goal of Cal ToxTrack and followed up by asking if the respondent felt this goal was accomplished. Five respondents stated they did feel this goal was accomplished, while 2 stated it was somewhat accomplished.

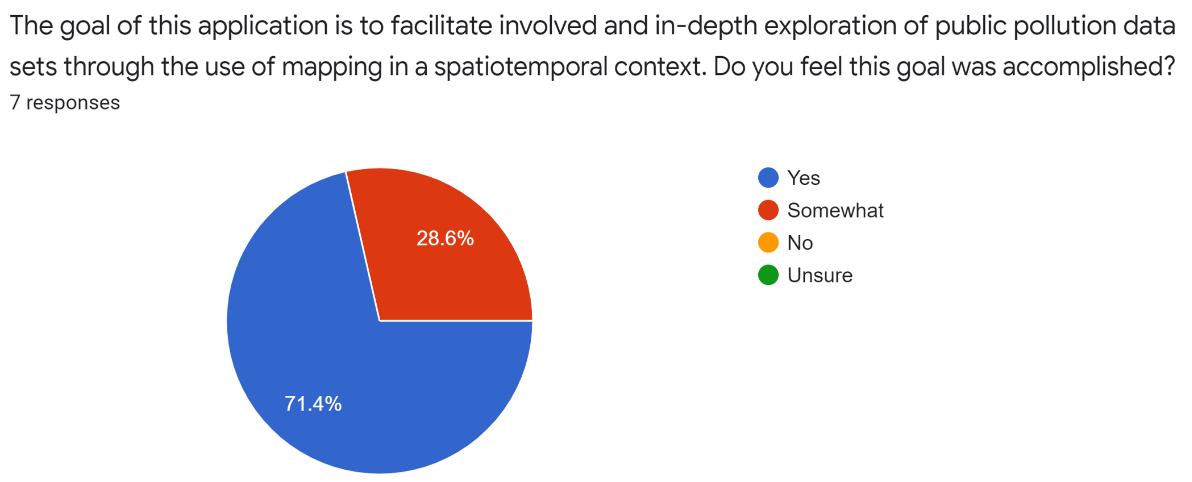


Figure 19. Survey question 2 responses

The second question asked whether the speed of the application was satisfactory with respect to how fast data were rendered on the map. Five of the respondents stated they were very satisfied with the speed, while one responded they were somewhat satisfied, and another responded they were very unsatisfied. Methods to increase application speed will be tested in future work.

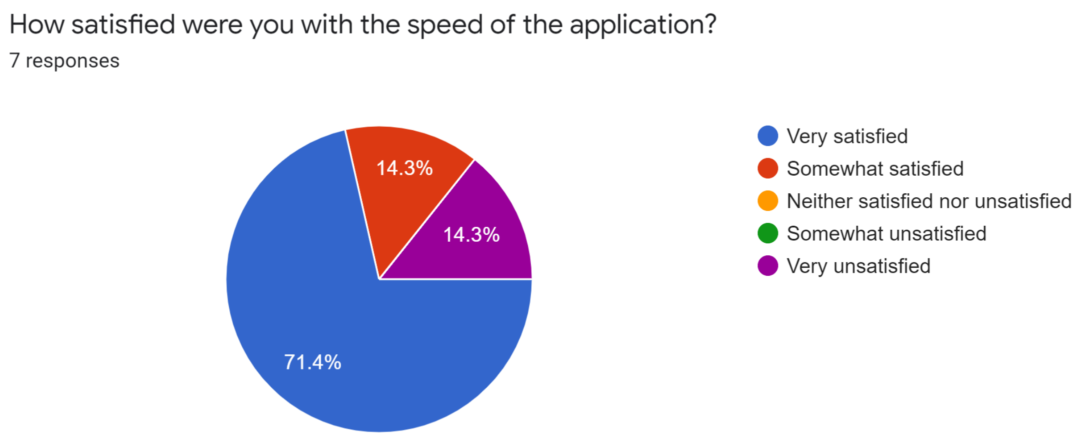


Figure 20. Survey question 3 results

Question 4 asked if the user was satisfied with the aesthetics of Cal ToxTrack. This helped the developer understand if the two-panel layout with the map container displayed in the center was an appealing and useful layout. None of the respondents were unsatisfied with the look of the application and the answers ranged from "neither satisfied nor unsatisfied" to "very satisfied".

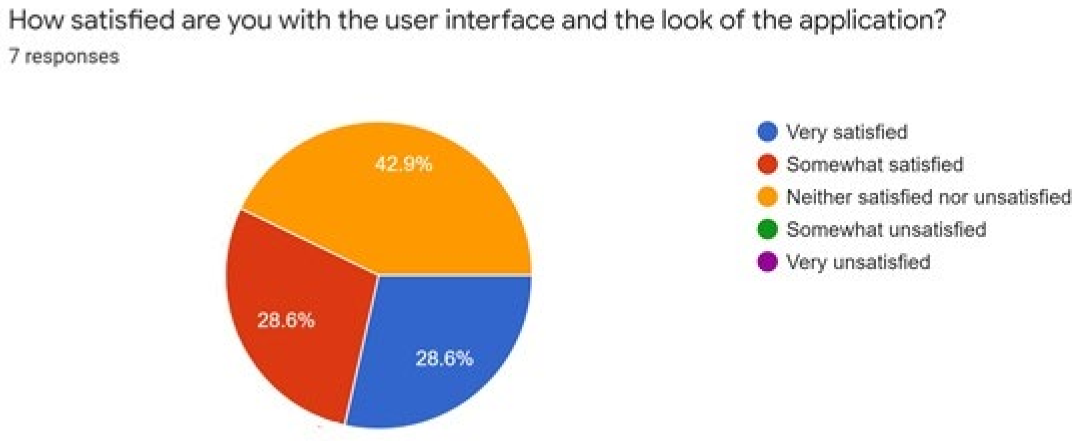


Figure 21. Survey question 4 results

The next question asked what respondents felt about the importance of each feature in the application, namely the time slider, query panel, and data styles. Most users felt all features were either very important or somewhat important to the application.

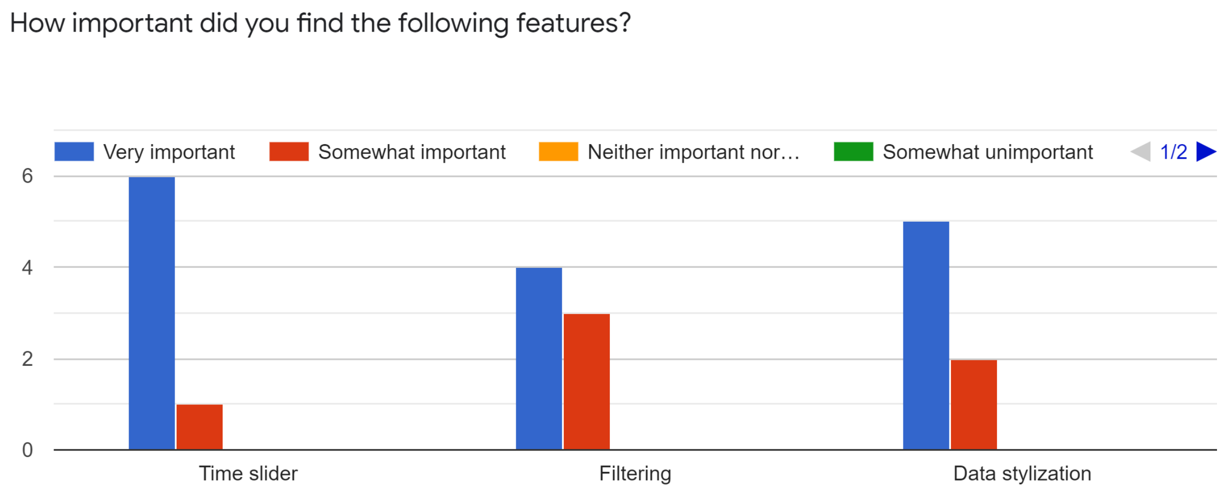


Figure 22. Survey question 6 results

The final question of the survey was left open-ended and asked users what they would like to see incorporated in future versions of the application. One request was for an an additional basemap with a satellite view to see buildings and imagery in relation to the pollutant releases. The responses are listed as below:

- "Different symbols/icons for different kinds of pollutants would be really helpful"
- "Couldn't slide all the way down the screen to adjust the chemical filters"
- "I did not see a point to the search bar as is, the menu is simple enough not to need a search bar. I would like the search button to find information on the map, such as a specific area or city in California where I could find the amount of pollution (specific chemical) I'm researching in a city or county. I think that would be a more useful tool for search function."
- "Phone app"
- "List:
    - Ex/include all of the chemicals and/or industries

- full screen or increase size of chart
- Export data and chart
- Additional to the slider have a calendar input field.
- Year over year comparison of e.g. for total pollution
- Explore pollution development over time for filtered dataset or even single locations”

Chapter 5 Conclusion

This chapter discusses the challenges encountered while developing Cal ToxTrack, as well as plans as to how it will be improved in the future. The application is part of a larger initiative by the developer to create a pollution map with datasets covering other regions in the U.S. This project aimed to establish the foundational programming of this application, with the idea that it is a working product to be continually improved upon. The goals of this application were ultimately accomplished. The geospatial database was set up, an API was created with a back end framework for the data, and this was integrated with a client-side interface— all developed with open source tools. In doing so, the original goals outlined were achieved as the application allows users to geolocate themselves, visualize the location and magnitude of chemical releases with a map and histogram chart, filter data with attributes, and temporally analyze data with a time slider that helps answer questions about whether communities are being overburdened with specific chemicals over multiple years. Functionality goals were also accomplished, as the application can be accessed via browsers across all devices. Section 5.1 discusses the challenges presented in the application development and section 5.2 outlines plans for future work. Section 5.3 contains the concluding comments for this project.

5.1. Cal ToxTrack Development Challenges

Throughout the development process there were application bugs, which are normal occurrences when coding. These bugs included code typos that needed to be identified and fixed. These typos occurred while typing some of the Django API URL paths and the JavaScript code for creating marker layers with Leaflet.

The most challenging aspect of this application development process was the rendering of such large datasets. While a solution is in place, this solution doesn't adhere to the dynamic data structure this project aimed for, since the application is rendering data from a static JSON file instead of communicating directly with the database. This solution also neglets to represent the entire dataset to the user. All data is displayed in the development environment for the application, but not in the deployed, live version. This application needs to represent all data and needs to be dynamic, thus, there are efforts to fix this issue which are discussed in Section 5.2.

As describesd, the goals that this application first outlined were achieved, but there are some missing components that will be developed in future work. The user should have the option to display both the TRI and DMR values by their raw release values and their toxic weighted equivalent pound equivalent (TWPE). Currently, only the DMR contains information on the TWPE. Additionally, visual issues with the legend are present in the final version of Cal ToxTrack, namely the data symbols to not align correctly with their labels and this also prevented the DMR data from being added to the legend. While this does not hinder functionality, the application would be enhanced and more user friendly if a working legend were included for both datasets on the map. This will be addressed in future work.

5.2. Developing a Project with Open Source

This project benefitted greatly from working with open source tools. Because of how widespread Django is, debugging issues that arose in the tool was greatly aided by the amount of online help provided by other users on forums and blogs. No issues or bugs with the Django source code were identified while developing the tool, however, should there be bugs in the future when new Django releases are rolled out, the developer can flag the bug to be fixed as quickly as possible, thus contributing to the open source community.

5.3. Future Work

Cal ToxTrack is actively under development. The future work addresses both challenges faced during development and integration of survey respondent suggestions. Important aspects that will be added or improved on are included in this section. There are plans to enhance the application with a larger database, more analytical tools, more fluid user interaction, and integration of user assessment recommendations. These aspects can categorized in three ways: data expansion, additional toolsets, and performance enhancement.

Currently, the application only shows pollution data from the TRI and DMR that represent toxic chemical release into water. The database will expand to include toxic chemical releases into the air and onto land. The pollution datasets and their sources are included in Table 2. Of the two pollution datasets currently in Cal ToxTrack, only the DMR contains an attribute that represents the release in the toxic weighted pounds equivalent (TWPE). The TRI dataset will be updated in the database to also include these TWPE values so users can have the option to view chemical releases by raw output or adjusted values based on the individual chemicals toxicity.

Table 2. Data sources to be added to Cal ToxTrack in future work

DATA SOURCE	CONTENTS	DATUM	PURPOSE	SCALE
National Priority List (NPL), EPA Superfund	Sites that are proposed, currently on, and removed from the list of national concern for environmental cleanup	North American Datum of 1983	Represent areas of highest concern in the past and present	USA
Decommissioned Nuclear Site, U.S. Nuclear Regulatory Commission	Locations of the nuclear sites listed as undergoing decommissioning	North American Datum of 1983	Identify areas undergoing decommissioning since some still	USA

			hold nuclear waste on-site	

Data will also be added that are not related to pollution to provide the user with context that can be used for further analysis. This includes demographic data from the American Community Survey and the Census. This additional data can allow additional questions to be explored, such as the identification of over-burdening on specific communities by comparing demographics to the amont and number of years certain toxic chemicals have been released. Each dataset will be categorized in the Table of Contents pane as “Land”, “Air”, “Water”, and “Demographic” layers that will be visually separated and colorized to distinguish the data types and purposes for the user. The Table of Contents will include an “*i”* icon next to all layers that initializes a popup that describes the corresponding dataset source, purpose, and most recent access date.

There are also several tools and functionalities that will be added to the application for a better user experience. Some basic geospatial application functionalities will include a “Print Map” button so users can easily share their findings with others, a bookmarks bar to store views users want to save, an upload button for additional GeoJSON data to be added to the map for a more personalized experience, and an attribute table to view the raw data. Another seemingly basic functionality, which is complicated in development, is the query tool. Currently, it only allows the users to filter the TRI dataset by industry, but the ultimate goal is to have ability to also filter by chemical. This is difficult in development because there are over 300 chemicals listed in the TRI, so a different filtering interface will be required to facilitate this function in the application.

Existing tools will be made more sophisticated. For example, a play button will be added to the time slider that automatically changes the slider input to sift through the years one by one, displaying the data automatically. This will offer a smoother visualization for the year-to-year change by not requiring the user to look at the slider and manually switch the year input. This will make it easier to compare differing years and their releases. Another tool, the histogram chat, will be expanded by including summary information for the DMR layer when it is clicked. A pie chart representing releases based on industry may be added to the chart.

The final component, the performance enhancement, will be carried out by addressing the rendering latency issue currently present in the application. There are two possible reasons for slow rendering. First, the code written in the API generator may be redundant and cause many database requests to be sent to the database when the developer only intended there to be one request to grab all the data. Secondly, the data could be cached on the client side for faster filtering with the time slider and query tools in the application. Each possibility will be explored to identify the cause of latency to appropriately represent the datasets in the application.

5.3 Final Thoughts

Cal ToxTrack is a living application and will continue to be developed for the public. With high demand for products and booming economies comes higher risks from industry waste. Awareness is the first step for keeping communities safe and extends chemical safety responsibility to those at risk by giving agency to the general public. Cal ToxTrack was developed to be a tool that helps facilitate this awareness.

References

Bureau of Economic Analysis (BEA) 2020. *Gross Domestic Product by State, 4th Quarter and Annual 2019.* BEA 20-18. April 7, 2020. https://apps.bea.gov/regional/bearfacts/action.cfm

California Department of Food and Agriculture (CDFA) 2018. “California Agriculture Production Statistics.” CDFA. https://www.cdfa.ca.gov/statistics/

California Environmental Protection Agency 2016. The History of the California Environmental Protection Agency. CAWeb Publishing Services. https://calepa.ca.gov/wp-content/uploads/sites/6/2016/10/About-History01-Report.pdf

Collins, M., S. Pulver, D. Hill, and B. Manski. 2020. Characterizing disproportionality in facility-level toxic releases in US manufacturing, 1998–2012. *Environmental Research Letters* 15 (6): 64002.

Commissariat l'Energie Atomique (CEA), Centre National de la Recherche Scientifique (CNRS), and Institut National de Recherche en Informatique et en Automatique (INRIA). 2006. Ol-ext. Version 1.0. September 5, 2006. https://viglino.github.io/ol-ext/doc/doc-pages/

Exchange Project. 2006. “Real People—Real Stories: Afton, NC (Warren County).” University of North Carolina, Department of Health Behavior and Health Education. September 2006.

Taylor, Dorceta. 2014. Toxic Communities. NYU Press. Kindle Edition.

Fernandez, Mariela. 2018. “Increasing Community Engagement in Latino Residents to Improve Health Outcomes.” *Local Environment* 23, no. 9: 920–33. https://doi.org/10.1080/13549839.2018.1500530.

Franklin, Ben. 1985. “Toxic Cloud Leaks at Carbide Plant in West Virginia.” *New York Times*, August 12, 1985. https://www.nytimes.com/1985/08/12/us/toxic-cloud-leaks-at-carbide-plant-in-west-virginia.html.

Hochstein, Colette, and Marti Szczur. 2006. “Toxmap.” *Medical Reference Services Quarterly* 25, no 3 (January): 13-31. https://doi.org/10.1300/J115v25n03_02.

National Library of Medicine. 2018. “Fact Sheet TOXMAP: Environmental Health Maps.” Accessed June 28, 2020. http://wayback.archive-it.org/org-350/20180312141653/https:/www.nlm.nih.gov/pubs/factsheets/toxmap.html.

Neumann, Catherine M. 2009. “Improving the U.S. EPA toxic release inventory database for environmental health research.” Journal of Toxicology and Environmental Health, Part B Critical Reviews. 1:3, 259-270. https://doi.org/10.1080/10937409809524555

Nodder, Chris. 2013. “UX Foundations: Information Architecture Video Tutorial: LinkedIn Learning, Formerly Lynda.com,” July 31, 2013. https://www.linkedin.com/learning/ux-foundations-information-architecture/the-right-information-architecture-is-crucial-to-your-site?u=76870426.

Office of Land and Emergency Management. 2017. The Emergency Planning and Community Right-to-Know Act. EPA. November 2017. https://www.epa.gov/sites/production/files/2017-08/documents/epcra_fact_sheet_overview_8-2-17.pdf.

Persico, Claudia. 2020. “Can Pollution Cause Poverty? The Effects of Pollution on Educational, Health, and Economic Outcomes.” IZA Discussion Paper No. 12965: 39. Available at: https://www.iza.org/publications/dp/12965/.

Pop, Dragos-Paul, and Adam Altar. 2014. Designing an MVC model for rapid web application development. *Procedia Engineering* 69 : 1172-9, http://www.sciencedirect.com/science/article/pii/S187770581400352X.

Roth, Noelle W. 2014. “The Pollution Next Door.” Duke University Superfund Research Center (blog). January 14, 2014. https://sites.nicholas.duke.edu/superfund/the-pollution-next-door/.

Schulson, Michael. 2019. “Federal Toxmap Shutters, Raising the Ire of Pollution Researchers.” Scientific American, December 17, 2019. https://www.scientificamerican.com/article/federal-toxmap-shutters-raising-the-ire-of-pollution-researchers/.

Sharma, Mayank. 2021. “Microsoft Says Now Is the Time for All Firms to Embrace Open Source,” January 19, 2021. https://www.techradar.com/news/microsoft-says-now-is-the-time-for-all-firms-to-embrace-open-source.

South Coast AQMD. n.d. “The Southland’s War on Smog: Fifty Years of Progress Toward Clean Air.” Accessed July 20, 2020. http://www.aqmd.gov/home/research/publications/50-years-of-progress#The%20Arrival%20of%20Air%20Pollution.

The Debian Free Software Guidelines (DFSG). 2004. “Debian Social Contract”. April 26, 2004. https://www.debian.org/social_contract

U.S. Environmental Protection Agency. 2020. *Toxic Chemical Release Inventory Reporting Forms and Instructions: Section 313 of the Emergency Planning and Community Right-to-Know Act*. EPA 740-B-19-037. January 2020.

U.S. Environmental Protection Agency (EPA). 2018. “TRI-Covered Industry Sectors.” Accessed October 20, 2020. https://www.epa.gov/toxics-release-inventory-tri-program/tri-covered-industry-sectors

U.S. Environmental Protection Agency. 2017. “Addition of Certain Chemicals.” June 13, 2017. https://www.epa.gov/toxics-release-inventory-tri-program/addition-certain-chemicals.

U.S. Environmental Protection Agency. 2016. “TRI Burden Reduction Rule.” July 28, 2016. https://www.epa.gov/toxics-release-inventory-tri-program/tri-burden-reduction-rule.

U.S. Environmental Protection Agency. 2012. *Toxic Weighting Factors Methodology*. EPA-820-R-12-005. March 2012.

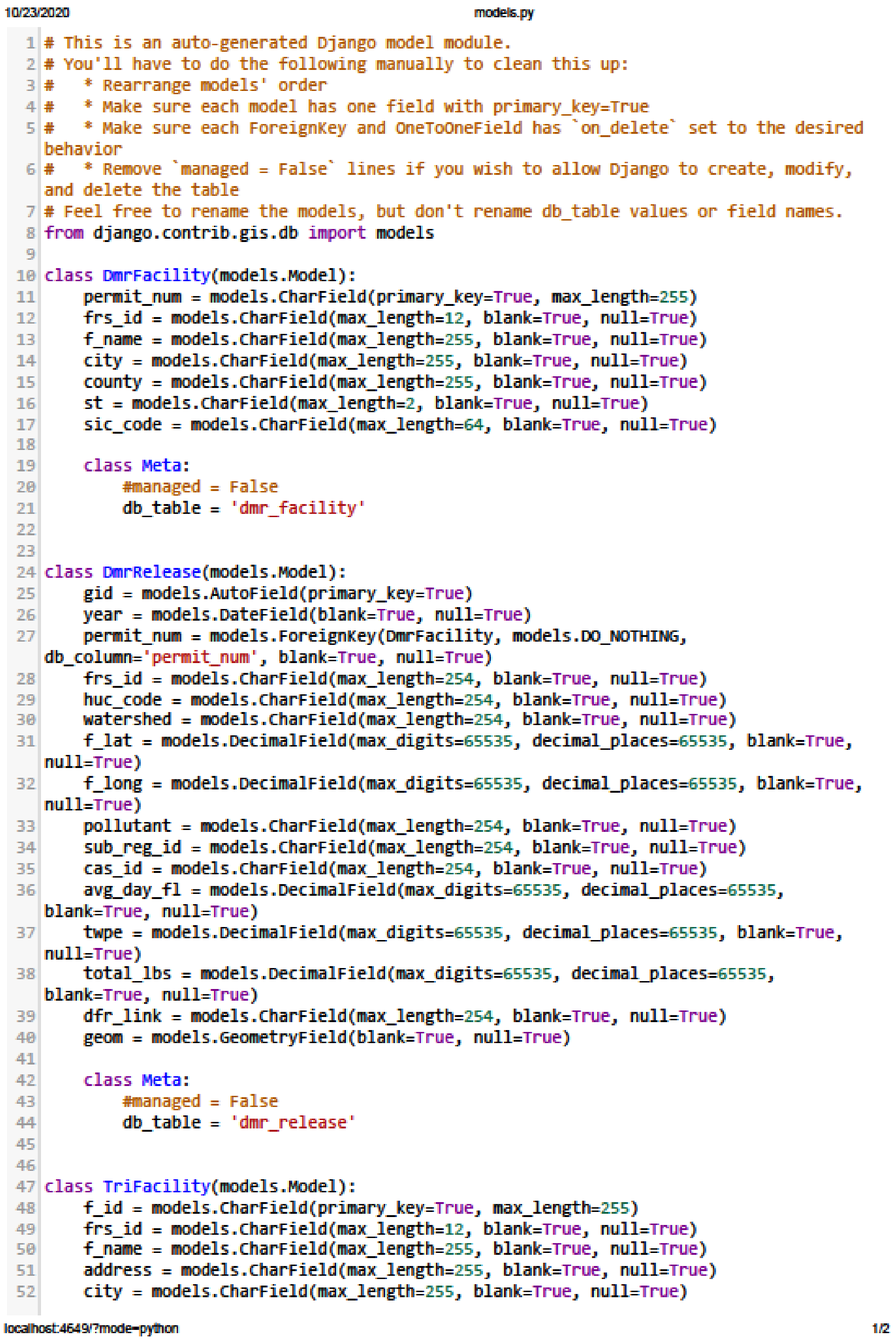
U.S. Environmental Protection Agency. 2010. *NPDES Permit Writers' Manual.* EPA-833-K-10-001. September 2010. https://www.epa.gov/sites/production/files/2015-09/documents/pwm_front.pdf

U.S. Environmental Protection Agency. 1993. *National Emission Standards for Hazardous Air Pollutants for Source Categories: Perchloroethylene Dry Cleaning Facilities*. EPA AD-FRL-4732-9. September 22, 1993.

U.S. Environmental Protection Agency. n.d. *Toxic Release Inventory Search Plus Tool.* Accessed August 9, 2020. https://edap.epa.gov/public/extensions/TRISearchPlus/TRISearchPlus.html

Appendix A – Code for MVC

I. *MODEL: Models.py* file that was auto-generated from the legacy data



10/23/2020 models.py

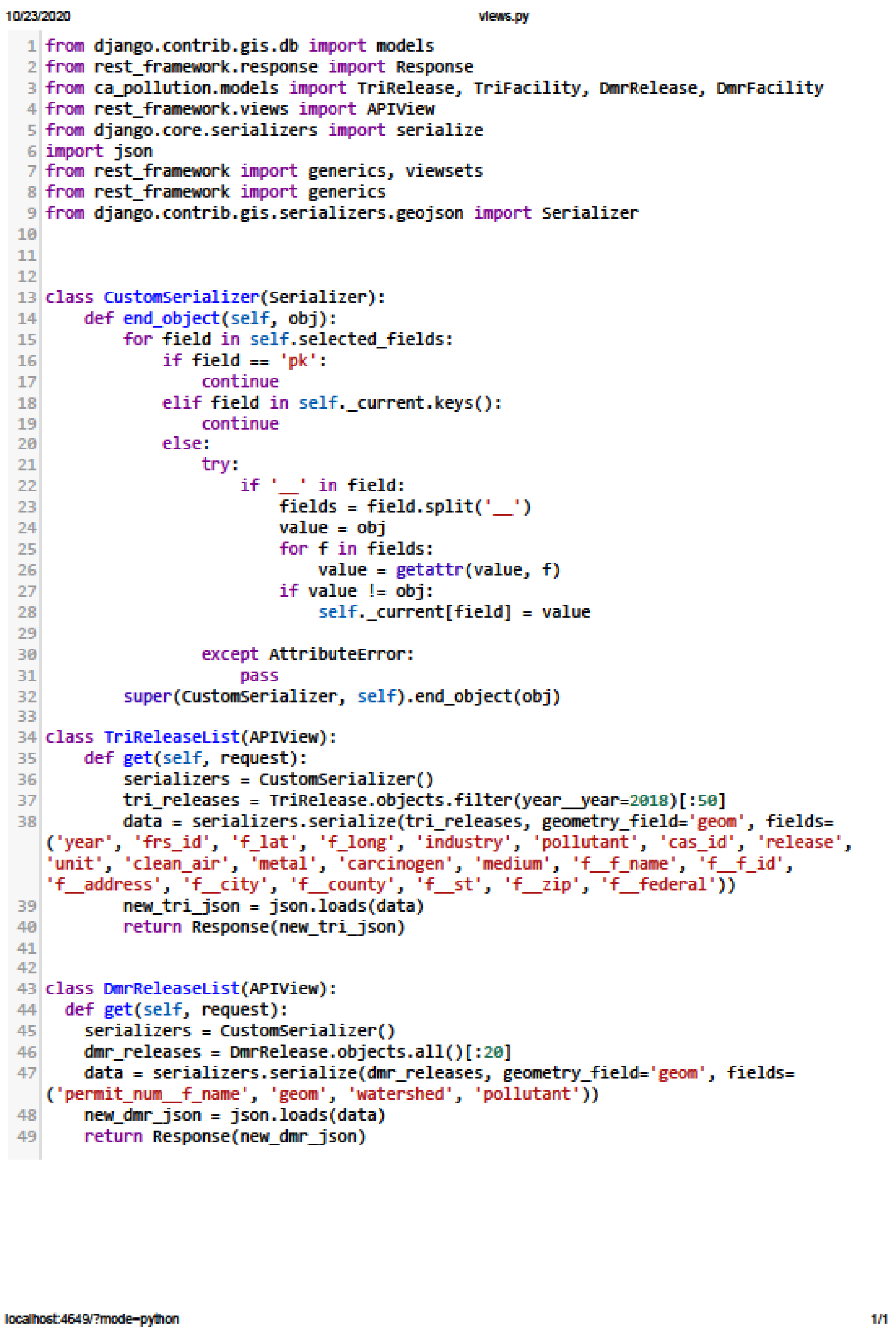
```python
# This is an auto-generated Django model module.
# You'll have to do the following manually to clean this up:
#   * Rearrange models' order
#   * Make sure each model has one field with primary_key=True
#   * Make sure each ForeignKey and OneToOneField has `on_delete` set to the desired behavior
#   * Remove `managed = False` lines if you wish to allow Django to create, modify, and delete the table
# Feel free to rename the models, but don't rename db_table values or field names.
from django.contrib.gis.db import models

class DmrFacility(models.Model):
    permit_num = models.CharField(primary_key=True, max_length=255)
    frs_id = models.CharField(max_length=12, blank=True, null=True)
    f_name = models.CharField(max_length=255, blank=True, null=True)
    city = models.CharField(max_length=255, blank=True, null=True)
    county = models.CharField(max_length=255, blank=True, null=True)
    st = models.CharField(max_length=2, blank=True, null=True)
    sic_code = models.CharField(max_length=64, blank=True, null=True)

    class Meta:
        #managed = False
        db_table = 'dmr_facility'


class DmrRelease(models.Model):
    gid = models.AutoField(primary_key=True)
    year = models.DateField(blank=True, null=True)
    permit_num = models.ForeignKey(DmrFacility, models.DO_NOTHING, db_column='permit_num', blank=True, null=True)
    frs_id = models.CharField(max_length=254, blank=True, null=True)
    huc_code = models.CharField(max_length=254, blank=True, null=True)
    watershed = models.CharField(max_length=254, blank=True, null=True)
    f_lat = models.DecimalField(max_digits=65535, decimal_places=65535, blank=True, null=True)
    f_long = models.DecimalField(max_digits=65535, decimal_places=65535, blank=True, null=True)
    pollutant = models.CharField(max_length=254, blank=True, null=True)
    sub_reg_id = models.CharField(max_length=254, blank=True, null=True)
    cas_id = models.CharField(max_length=254, blank=True, null=True)
    avg_day_fl = models.DecimalField(max_digits=65535, decimal_places=65535, blank=True, null=True)
    twpe = models.DecimalField(max_digits=65535, decimal_places=65535, blank=True, null=True)
    total_lbs = models.DecimalField(max_digits=65535, decimal_places=65535, blank=True, null=True)
    dfr_link = models.CharField(max_length=254, blank=True, null=True)
    geom = models.GeometryField(blank=True, null=True)

    class Meta:
        #managed = False
        db_table = 'dmr_release'


class TriFacility(models.Model):
    f_id = models.CharField(primary_key=True, max_length=255)
    frs_id = models.CharField(max_length=12, blank=True, null=True)
    f_name = models.CharField(max_length=255, blank=True, null=True)
    address = models.CharField(max_length=255, blank=True, null=True)
    city = models.CharField(max_length=255, blank=True, null=True)
```

localhost:4649/?mode=python 1/2

```
    county = models.CharField(max_length=255, blank=True, null=True)
    st = models.CharField(max_length=2, blank=True, null=True)
    zip = models.CharField(max_length=10, blank=True, null=True)
    federal = models.BooleanField(blank=True, null=True)

    class Meta:
        #managed = False
        db_table = 'tri_facility'


class TriRelease(models.Model):
    gid = models.AutoField(primary_key=True)
    year = models.DateField(blank=True, null=True)
    f = models.ForeignKey(TriFacility, models.DO_NOTHING, blank=True, null=True)
    frs_id = models.CharField(max_length=254, blank=True, null=True)
    f_lat = models.DecimalField(max_digits=65535, decimal_places=65535, blank=True,
null=True)
    f_long = models.DecimalField(max_digits=65535, decimal_places=65535, blank=True,
null=True)
    industry_c = models.CharField(max_length=254, blank=True, null=True)
    industry = models.CharField(max_length=254, blank=True, null=True)
    sic = models.CharField(max_length=254, blank=True, null=True)
    naics = models.CharField(max_length=254, blank=True, null=True)
    pollutant = models.CharField(max_length=254, blank=True, null=True)
    cas_id = models.CharField(max_length=254, blank=True, null=True)
    srs_id = models.CharField(max_length=254, blank=True, null=True)
    release = models.DecimalField(max_digits=65535, decimal_places=65535,
blank=True, null=True)
    unit = models.CharField(max_length=254, blank=True, null=True)
    clean_air = models.BooleanField(blank=True, null=True)
    metal = models.BooleanField(blank=True, null=True)
    carcinogen = models.BooleanField(blank=True, null=True)
    medium = models.CharField(max_length=254, blank=True, null=True)
    geom = models.GeometryField(blank=True, null=True)

    class Meta:
        #managed = False
        db_table = 'tri_release'

```

II. *VIEW*

A. *Views.py* file, creating custom GeoJSON serializer and applying it to entire dataset for TRI and DMR tables

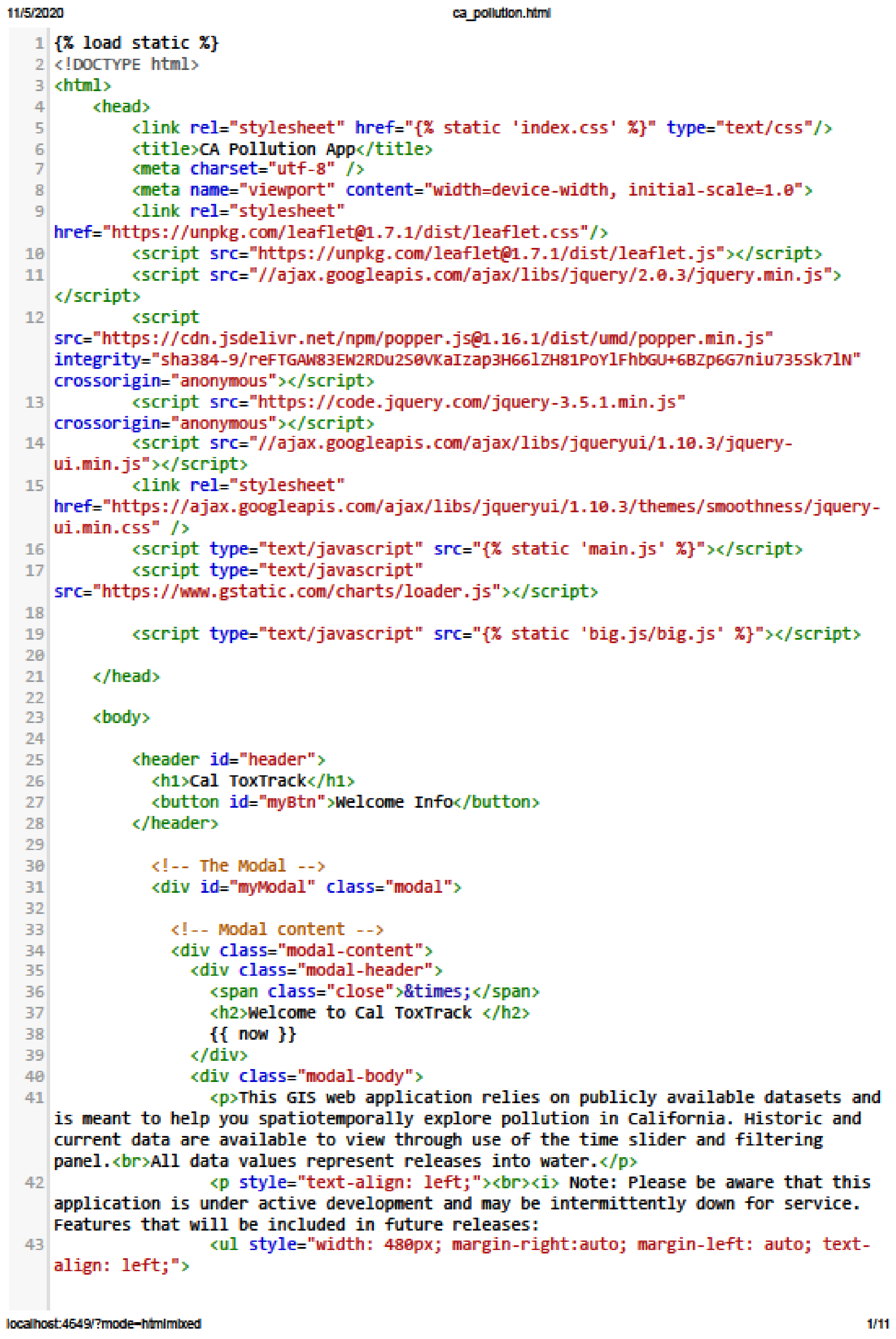


10/23/2020 views.py

```python
from django.contrib.gis.db import models
from rest_framework.response import Response
from ca_pollution.models import TriRelease, TriFacility, DmrRelease, DmrFacility
from rest_framework.views import APIView
from django.core.serializers import serialize
import json
from rest_framework import generics, viewsets
from rest_framework import generics
from django.contrib.gis.serializers.geojson import Serializer



class CustomSerializer(Serializer):
    def end_object(self, obj):
        for field in self.selected_fields:
            if field == 'pk':
                continue
            elif field in self._current.keys():
                continue
            else:
                try:
                    if '__' in field:
                        fields = field.split('__')
                        value = obj
                        for f in fields:
                            value = getattr(value, f)
                        if value != obj:
                            self._current[field] = value

                except AttributeError:
                    pass
        super(CustomSerializer, self).end_object(obj)

class TriReleaseList(APIView):
    def get(self, request):
        serializers = CustomSerializer()
        tri_releases = TriRelease.objects.filter(year__year=2018)[:50]
        data = serializers.serialize(tri_releases, geometry_field='geom', fields=('year', 'frs_id', 'f_lat', 'f_long', 'industry', 'pollutant', 'cas_id', 'release', 'unit', 'clean_air', 'metal', 'carcinogen', 'medium', 'f__f_name', 'f__f_id', 'f__address', 'f__city', 'f__county', 'f__st', 'f__zip', 'f__federal'))
        new_tri_json = json.loads(data)
        return Response(new_tri_json)


class DmrReleaseList(APIView):
  def get(self, request):
    serializers = CustomSerializer()
    dmr_releases = DmrRelease.objects.all()[:20]
    data = serializers.serialize(dmr_releases, geometry_field='geom', fields=('permit_num__f_name', 'geom', 'watershed', 'pollutant'))
    new_dmr_json = json.loads(data)
    return Response(new_dmr_json)
```

localhost:4649/?mode=python 1/1

B. ca_pollution.html file containing all applcication components



11/5/2020 ca_pollution.html

```
{% load static %}
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="{% static 'index.css' %}" type="text/css"/>
        <title>CA Pollution App</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"/>
        <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
        <script src="//ajax.googleapis.com/ajax/libs/jquery/2.0.3/jquery.min.js"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>
        <script src="https://code.jquery.com/jquery-3.5.1.min.js" crossorigin="anonymous"></script>
        <script src="//ajax.googleapis.com/ajax/libs/jqueryui/1.10.3/jquery-ui.min.js"></script>
        <link rel="stylesheet" href="https://ajax.googleapis.com/ajax/libs/jqueryui/1.10.3/themes/smoothness/jquery-ui.min.css" />
        <script type="text/javascript" src="{% static 'main.js' %}"></script>
        <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>

        <script type="text/javascript" src="{% static 'big.js/big.js' %}"></script>

    </head>

    <body>

        <header id="header">
          <h1>Cal ToxTrack</h1>
          <button id="myBtn">Welcome Info</button>
        </header>

          <!-- The Modal -->
          <div id="myModal" class="modal">

            <!-- Modal content -->
            <div class="modal-content">
              <div class="modal-header">
                <span class="close">&times;</span>
                <h2>Welcome to Cal ToxTrack </h2>
                {{ now }}
              </div>
              <div class="modal-body">
                <p>This GIS web application relies on publicly available datasets and is meant to help you spatiotemporally explore pollution in California. Historic and current data are available to view through use of the time slider and filtering panel.<br>All data values represent releases into water.</p>
                <p style="text-align: left;"><br><i> Note: Please be aware that this application is under active development and may be intermittently down for service. Features that will be included in future releases:
                <ul style="width: 480px; margin-right:auto; margin-left: auto; text-align: left;">
```

localhost:4649/?mode=htmlmixed 1/11

```
         <li>A play button for the time slider for smoother year transitions</li>
         <li>A "Select/Deselect All" button for the filter sections</li>
         <li>A filter section for chemicals</li>
         <li>Additional charts to summarize data</li>
         <li>Pollution data from additional sources</li>
        </i></ul></p>
       </div>
       <div class="modal-footer">
        <h3>Datasets last updated September 30, 2020.</h3>
       </div>
      </div>

     </div>
        <!-- Modal  -->
    <script>
     // Get the modal
     var modal = document.getElementById("myModal");

     // Get the button that opens the modal
     var btn = document.getElementById("myBtn");

     // Get the <span> element that closes the modal
     var span = document.getElementsByClassName("close")[0];

     // When the user loads the page, open the modal
     onload = function() {
      modal.style.display = "block";
     }

     // When the user clicks the button, open the modal
     btn.onclick = function() {
       modal.style.display = "block";
      }

     // When the user clicks on <span> (x), close the modal
     span.onclick = function() {
      modal.style.display = "none";
     }

     // When the user clicks anywhere outside of the modal, close it
     window.onclick = function(event) {
      if (event.target == modal) {
       modal.style.display = "none";
      }
     }
    </script>

    <div id="mapcontainer">
     <div id="map"></div>
    </div>
     <div id="sidenavopen" class="sidenavbutton" href="javascript:void(0)" onclick="openNav()">&#9776;</div>
     <div id="sidenavclose" class="sidenavbutton closebtn" href="javascript:void(0)" onclick="closeNav()">&times;</div>
    <!-- Right Panel   -->
     <div class="sidenavright">
      <p>Layers Table of Contents</p>
      <div class="pollutionlabel">
       <p>Water Releases</p>
      </div>
      <div id="layerswitch">
       <label><input type="checkbox" checked=true value="geojsonLayer" > Toxic Release Inventory</label><br>
       <label><input type="checkbox" checked=true value="dmrGeojsonLayer" > Discharge Monitoring Report</label>
       </div>
     </div>

    <!-- Left Panel -->
     <div id="sidenav" class="sidenav" style="color: rgb(250, 250, 250);">
      <div id="filterblock">
       <p>Pollution Data Attribute Filter</p>
      <input id="filter" type="text" placeholder="Search..."/>
       <ul class="level1">
        <li class="listtitle expander">Industry (TRI only)<span class="ui-icon ui-icon-plus" style="display:inline-block"></span></li>
         <ul class="level2 condense" class="searchable">
          <li><input type="checkbox" class="event-type" name="plastics-and-rubber" value="Plastics and Rubber" checked="true">Plastics and Rubber</input></li>
          <li><input type="checkbox" class="event-type" name="furniture" value="Furniture"  checked="true">Furniture</input></li>
          <li><input type="checkbox" class="event-type" name="fabricated-metals" value="Fabricated Metals" checked="true">Fabricated Metals</input></li>
          <li><input type="checkbox" class="event-type" name="electric-utilities" value="Electric Utilities"  checked="true">Electric Utilities</input></li>
          <li><input type="checkbox" class="event-type" name="nonmetallic-mineral-product" value="Nonmetallic Mineral Product" checked="true">Nonmetallic Mineral Product</input></li>
          <li><input type="checkbox" class="event-type" name="textiles" value="Textiles" checked="true">Textiles</input></li>
          <li><input type="checkbox" class="event-type" name="metal-mining" value="Metal Mining" checked="true">Metal Mining</input></li>
          <li><input type="checkbox" class="event-type" name="tobacco" value="Tobacco" checked="true">Tobacco</input></li>
          <li><input type="checkbox" class="event-type" name="machinery" value="Machinery" checked="true">Machinery</input></li>
          <li><input type="checkbox" class="event-type" name="leather" value="Leather" checked="true">Leather</input></li>
          <li><input type="checkbox" class="event-type" name="paper" value="Paper" checked="true">Paper</input></li>
          <li><input type="checkbox" class="event-type" name="publishing" value="Publishing" checked="true">Publishing</input></li>
          <li><input type="checkbox" class="event-type" name="printing" value="Printing" checked="true">Printing</input></li>
          <li><input type="checkbox" class="event-type" name="electrical-equipment" value="Electrical Equipment" checked="true">Electrical Equipment</input></li>
          <li><input type="checkbox" class="event-type" name="chemicals" value="Chemicals" checked="true">Chemicals</input></li>
          <li><input type="checkbox" class="event-type" name="petroleum-bulk-terminals" value="Petroleum Bulk Terminals" checked="true">Petroleum Bulk Terminals</input></li>
          <li><input type="checkbox" class="event-type" name="beverages" value="Beverages" checked="true">Beverages</input></li>
          <li><input type="checkbox" class="event-type" name="hazardous-waste" value="Hazardous Waste" checked="true">Hazardous Waste</input></li>
          <li><input type="checkbox" class="event-type" name="wood-products" value="Wood Products" checked="true">Wood Products</input></li>
```

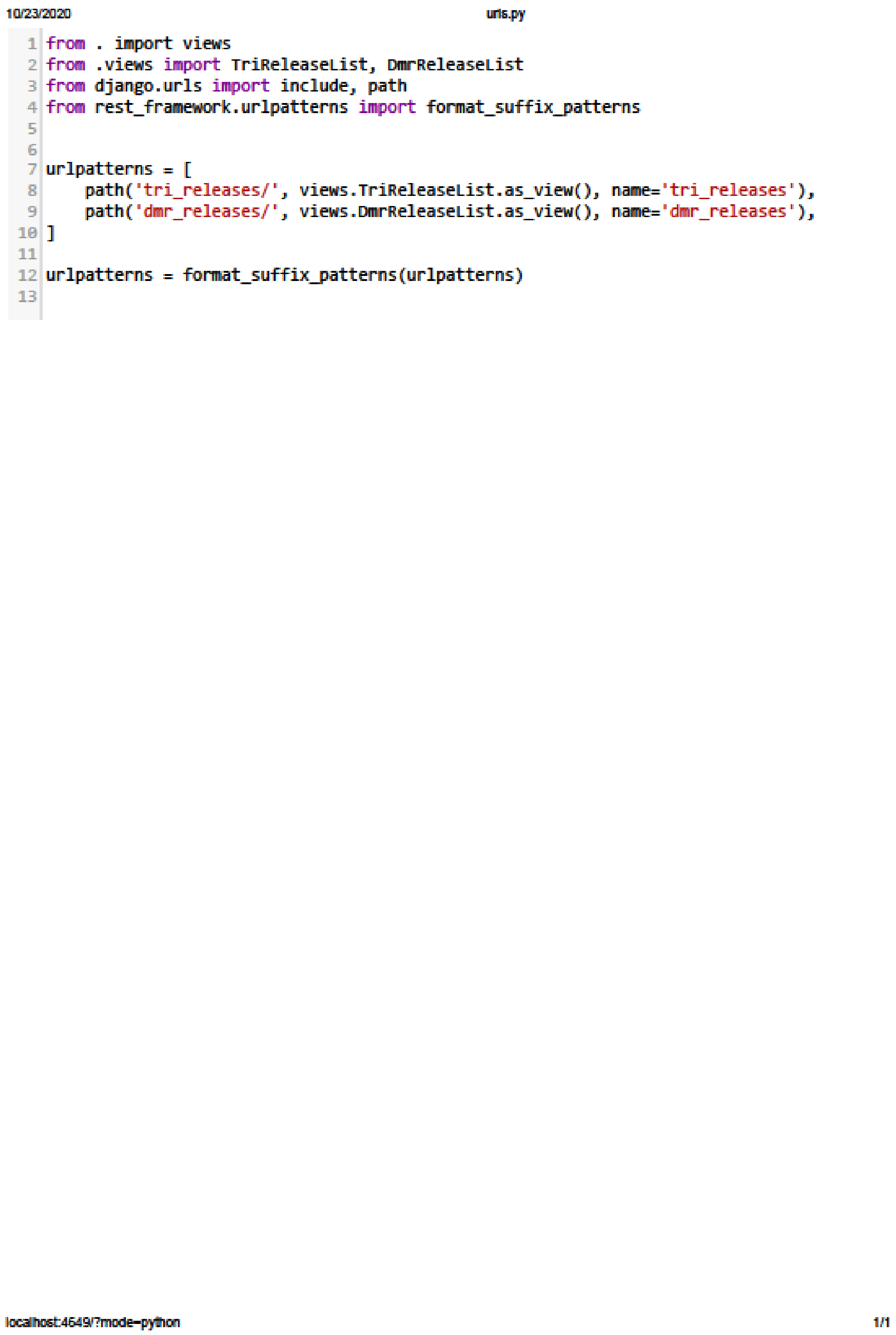
```
                    <li><input type="checkbox" class="event-type" name="textile-product" value="Textile Product" checked="true">Textile Product</input></li>
                    <li><input type="checkbox" class="event-type" name="petroleum" value="Petroleum" checked="true">Petroluem</input></li>
                    <li><input type="checkbox" class="event-type" name="computers-and-electronic-products" value="Computers and Electronic Products" checked="true">Computers and Electronic Products</input></li>
                    <li><input type="checkbox" class="event-type" name="apparel" value="Apparel" checked="true">Apparel</input></li>
                    <li><input type="checkbox" class="event-type" name="food" value="Food" checked="true">Food</input></li>
                    <li><input type="checkbox" class="event-type" name="chemical-wholesalers" value="Chemical Wholesalers" checked="true">Chemical Wholesalers</input></li>
                    <li><input type="checkbox" class="event-type" name="transportation-equipment" value="Transportation Equipment" checked="true">Transportation Equipment</input></li>
                    <li><input type="checkbox" class="event-type" name="primary-metals" value="Primary Metals" checked="true">Primary Metals</input></input>
                    <li><input type="checkbox" class="event-type" name="miscellaneous-manufacturing" value="Miscellaneous Manufacturing" checked="true">Miscellaneous Manufacturing</input></li>
                    <li><input type="checkbox" class="event-type" name="other" value="Other" checked="true">Other</input></li>
                  </ul>
                <li class="listtitle expander">Chemical<span class="ui-icon ui-icon-plus" style="display:inline-block"></span></li>
                    <ul class="level2 condense" class="searchable">
                      <li><input type="checkbox" class="event-type" name="oxygen" value="Oxygen" checked="true">Oxygen</input></li>
                      <li><input type="checkbox" class="event-type" name="helium" value="Helium" checked="true">Helium</input></li>
                  </ul>
              </ul>
            </div>
              <div class="main" class="inputs" id="years">
                <div id="chart_div"></div>
                  <input type="range" class="year" min="1987" max="2020" value=2018 id="slider">
                  <div id="selector">
                    <div class="SelectBtn"></div>
                    <div id="SelectValue"></div>
                  </div>
            </div>
        </div>

              </script>



              <script>

                var slider = document.getElementById("slider");
                var selector = document.getElementById("selector");

                SelectValue.innerHTML = slider.value;

                slider.oninput = function() {
                  SelectValue.innerHTML = this.value;
                  selector.style.left = this.value;
                }
              </script>

              <script src="https://code.jquery.com/jquery-1.9.1.min.js"></script>


              <script>
                // Setup map
                var terrain = L.tileLayer('https://stamen-tiles-
{s}.a.ssl.fastly.net/terrain/{z}/{x}/{y}.jpg'),
                    streets =
 L.tileLayer('https://{s}.tile.osm.org/{z}/{x}/{y}.png', {attribution: '&copy; <a
href="https://osm.org/copyright">OpenStreetMap</a> contributors'
                      }),
                    satellite =
L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/satellite-streets-
v11/tiles/{z}/{x}/{y}?access_token=' + '{{ mapbox }}')
                    dark = L.tileLayer('https://api.mapbox.com/styles/v1/mapbox/dark-
v10/tiles/{z}/{x}/{y}?access_token=' + '{{ mapbox }}')
                    ;

                var baseMaps = {
                    "Dark": dark,
                    "Satellite": satellite,
                    "Terrain": terrain,
                    "Streets": streets,
                  };

                var myMap = L.map('map', {
                  center: [35.01907305286242, -118.61962890625001],
                  zoom: 4,
                  minZoom: 6,
                  maxZoom: 17,
                  layers: [terrain, streets, satellite, dark]
                  });

                  L.control.layers(baseMaps,null, {collapsed: false, position:
'bottomleft'}).addTo(myMap);

                  function getPixel(p) {
                  return p > 1000 ? 5:
                         p > 800 ? 4:
                         p > 400 ? 3:
                         p > 200 ? 2:
                                  1;
                     }
              </script>




              <script>

                // Get initial checkbox states from HTML
                var checkboxStates;
                function updateCheckboxStates() {
                  checkboxStates = {
                    years: [],
                    eventTypes: []
                  }
                  for (let input of document.querySelectorAll('.event-type')) {
                    if(input.checked) {
                      switch (input.className) {
                        case 'event-type':
checkboxStates.eventTypes.push(input.value); break
                      }
                    }
                  }
                  for (let input of document.querySelectorAll('input')) {
                    if (input.className === "year") {
                      checkboxStates.years.push(input.value);
                    }
                  }
                }
                updateCheckboxStates();

                // Define symbology variation functions
                function getSize(r) {
                  return r > 1000 ? 16:
                         r > 800 ? 10:
                         r > 400 ? 8:
                         r > 200 ? 5:
                                  2;
                }


                function getOpacity(o) {
                  return o > 1000 ? 0.4:
                         o > 800 ? 0.4:
                         o > 400 ? 0.4:
                         o > 200 ? 0.4:
                                  0.4;
                }

                // Define geojson layer (without any data yet)
                var geojsonLayer = L.geoJson(null, {
                  filter: function(feature) {
                    var isYearChecked =
checkboxStates.years.includes(feature.properties.year.substr(0,4));
                    var isEventTypeChecked =
checkboxStates.eventTypes.includes(feature.properties.industry);
                    return isYearChecked && isEventTypeChecked;
                  },
                  onEachFeature: function(feature, layer) {
                    if (feature.properties) {
                      var year = feature.properties.year.substr(0, 4);
                      var year = feature.properties.year.substr(0, 4);
                      var pollutant = feature.properties.pollutant;
                      var cas_id = feature.properties.cas_id;
                      var release = new
Big(feature.properties.release).toPrecision(5);
                      var unit = feature.properties.unit;
                      var facility = feature.properties.f__f_name;
                      var facility_id = feature.properties.f__f_id;
                      var industry = feature.properties.industry;
                      var facility_address = feature.properties.f__address;
                      var facility_city = feature.properties.f__city;
                      var facility_county = feature.properties.f__county;
                      var facility_st = feature.properties.f__st;
                      var facility_zip = feature.properties.f__zip;
                      var federal = feature.properties.f__federal;
                      layer.bindPopup("<div class='popuptitle'>TRI RELEASE</div><br>" +
                        "<div class='popuptitle'>YEAR: </div><div class='popupdata'>" + year +
                          "</div><div class='popuptitle'>POLLUTANT: </div><div class='popupdata'>" + pollutant +
                          "</div><div class='popuptitle'>CAS ID: </div><div class='popupdata'>" + cas_id +
                          "</div><div class='popuptitle'>RELEASE:</div><div class='popupdata'>" + release + " " + unit +
                          "</div><div class='popuptitle'>FACILITY:</div><div class='popupdata'>" + facility +
                          "</div><div class='popuptitle'>INDUSTRY:</div><div class='popupdata'>" + industry +
                          "</div><div class='popuptitle'>FACILITY ID:</div><div class='popupdata'>" + facility_id +
                          "</div><div class='popuptitle'>ADDRESS:</div><div class='popupdata'>" + facility_address + ", " + facility_city + ", " + facility_city + ", " + facility_zip + " " + facility_st +
                          "</div><div class='popuptitle'>FEDERAL:</div><div class='popupdata'>" + federal + "</div><br>" + "<div class='popuptitle'>Data last downloaded from <a href='https://www.epa.gov/enviro/tri-search' target='_blank'>source</a> on 09/30/2020.</div>",
                            //  { maxWidth: 500,
                              {maxHeight: 200,
                              autoPan: true,
                              closeButton: true,
                              closeOnClick: true,
                        });
                  }
                },
                  pointToLayer: function (feature, latlng) {
                    var geojsonMarkerOptions = {
                    radius: getSize(Big(feature.properties.release).toPrecision(5)),
                    fillColor: "rgb(32, 32, 124)",
                    color: "#000",
                    weight: 1,
                    opacity: 1,
                    fillOpacity: getOpacity(Big(feature.properties.release).toPrecision(5)),
                    };
                    return L.circleMarker(latlng, geojsonMarkerOptions);
                  }
                }).addTo(myMap);


                // DMR data
                var dmrGeojsonLayer = L.geoJson(null, {
                  filter: function(feature) {
                    var isYearChecked = checkboxStates.years.includes(feature.properties.year.substr(0,4));
                    // var isEventTypeChecked = checkboxStates.eventTypes.includes(feature.properties.industry);
                    return isYearChecked;
                  },
                  onEachFeature: function(feature, layer) {
                    if (feature.properties) {
                      var year = feature.properties.year.substr(0, 4);
                      var pollutant = feature.properties.pollutant;
                      var cas_id = feature.properties.cas_id;
                      var release = feature.properties.total_lbs;
                      var twpe = feature.properties.twpe;
                      var dfr_link = feature.properties.dfr_link;
                      layer.bindPopup("<div class='popuptitle'>DMR RELEASE</div><br>" +
                        "<div class='popuptitle'>YEAR: </div><div class='popupdata'>" + year +
                          "</div><div class='popuptitle'>POLLUTANT: </div><div class='popupdata'>" + pollutant +
                          "</div><div class='popuptitle'>CAS ID: </div><div class='popupdata'>" + cas_id +
                          "</div><div class='popuptitle'>RELEASE:</div><div class='popupdata'>" + release + " lbs" +
                          "</div><div class='popuptitle'>TWPE:</div><div class='popupdata'>" + twpe +
                           "</div><div class='popuptitle'>DATA SOURCE: </div><div class='popupdata'>" + "<a href=" + dfr_link+ "target='_blank>" + 'Link to DMR page' +"</a>",
                            //  { maxWidth: 500,
                              {maxHeight: 200,
                              autoPan: true,
                              closeButton: true,
                              closeOnClick: true,
                        });
                  }
                },
                  pointToLayer: function (feature, latlng) {
                    var geojsonMarkerOptions = {
                    // radius: getSize(release),
                    fillColor: "pink",
                    color: "#000",
                    weight: 1,
                    opacity: 1,
                    // fillOpacity: getOpacity(Big(feature.properties.release).toPrecision(5)),
                    };
                    return L.circleMarker(latlng, geojsonMarkerOptions);
                  }
                }).addTo(myMap);

                // Get JSON from web request
                // $.getJSON('https://localhost:8000/ca_pollution/api/tri_releases/', function(json) {
                  // Update geojson layer with data
                  // console.log(json);
                  // geojsonLayer.addData(json);


                // Get JSON from your test file
                $.getJSON("static/tri_data.json", function(json) {
                // Update geojson layer with data
                  console.log(json);
                  geojsonLayer.addData(json);

                  // Listen to 'change' event of all inputs
                  for (let input of document.querySelectorAll('input')) {
                    input.onchange = (e) => {
                      geojsonLayer.clearLayers()
                      updateCheckboxStates();
```

```
           geojsonLayer.addData(json);
           console.log(checkboxStates.years);
           drawChart();
          }
         }
        });

        $.getJSON("static/dmr_data.json", function(json) {
        // Update geojson layer with data
         console.log(json);
         dmrGeojsonLayer.addData(json);

         // Listen to 'change' event of all inputs
         document.getElementById('slider').oninput = (e) => {
           console.log("DMR CHANGE");
           dmrGeojsonLayer.clearLayers()
           updateCheckboxStates();
           dmrGeojsonLayer.addData(json);
           console.log(checkboxStates.years);
           SelectValue.innerHTML = slider.value;
           console.log(checkboxStates.years);
           SelectValue.innerHTML = checkboxStates.years;
           selector.style.left = checkboxStates.year;
          }
         }
        );

       // Use jQuery to listen for Table of Contents event changes to
add/remove layers
       $( '#layerswitch input[type=checkbox]' ).click(function( event ) {
         layerClicked = window[event.target.value];
         console.log(layerClicked);
         if (myMap.hasLayer(layerClicked)) {
          console.log(layerClicked);
          myMap.removeLayer(layerClicked);
         }
         else{
          console.log("added!");
          myMap.addLayer(layerClicked);
         };
        });


       // Legend Control

       var legend = new L.Control({position: 'bottomright'});
       legend.onAdd = function (map) {

        var div = L.DomUtil.create('div', 'info legend'),
          grades = [0, 200, 400, 800, 1000],
          labels = [];

        for (var i = 0; i < grades.length; i++) {
         div.innerHTML +=
         '<i class="fas fa-circle fa-' + getPixel(grades[i] + 1) +'x"
style="color: rgb(32, 32, 124);"></i> ' +
         grades[i] + (grades[i + 1] ? '&ndash;' + grades[i + 1] + " lbs"+
'<br>' + '<br>' + '<br>' : '+');
        }
        return div;
```

```
                }
                legend.addTo(myMap);




            L.control.scale({imperial: true, metric: false, position:
"topleft"}).addTo(myMap);

      </script>
  <!-- style="color: #f111; font-size:' + getSize(grades[i] + 1) + '" -->
  <script src="https://kit.fontawesome.com/1448dc8b97.js" crossorigin="anonymous">
</script>

  <script type="text/javascript">
    // Load the Visualization API and the corechart package.
      google.charts.load('current', {'packages':['corechart']});

      // Set a callback to run when the Google Visualization API is loaded.
      google.charts.setOnLoadCallback(drawChart);

      function drawChart() {
        var jsonData = $.ajax({
          url: "static/tri_data.json",
          dataType: "json",
          //async: false,
          success: function (jsonData){
            console.log(jsonData);
            var data = new google.visualization.DataTable();

            data.addColumn('string', 'industry');
            data.addColumn('number', 'release');
            console.log(jsonData.features.length);
            console.log(checkboxStates.years[0]);
            for (var i=0; i < jsonData.features.length; i++) {
              if (jsonData.features[i].properties["year"].substr(0,4) ===
checkboxStates.years[0]) {
                var industry = jsonData.features[i].properties["industry"];
                var release = parseFloat(jsonData.features[i].properties["release"]);
                console.log(typeof release);
                data.addRow([industry, release]);
              }
            }

            var options = {
                title: 'TRI Release Histogram for ' + checkboxStates.years[0],
                histogram: {lastBucketPercentile: 15},
                vAxis: { scaleType: 'null', title: 'Facility Count', titleTextStyle:
{fontSize: 14, bold: true, italic: false} },
                colors: ['mediumpurple'],
                hAxis: { title: 'Release in lbs.', titleTextStyle: {fontSize: 14,
bold: true, italic: false}},
                legend: {position: 'none',},
                series: {labelInLegend: true, visibleInLegend: true,},
              };

              var chart = new
google.visualization.Histogram(document.getElementById('chart_div'));
              chart.draw(data, options);
            }
        });
      }


    </script>

  </body>
</html>
```

III. *CONTROLLER:*

Urls.py file that created the routes for the API

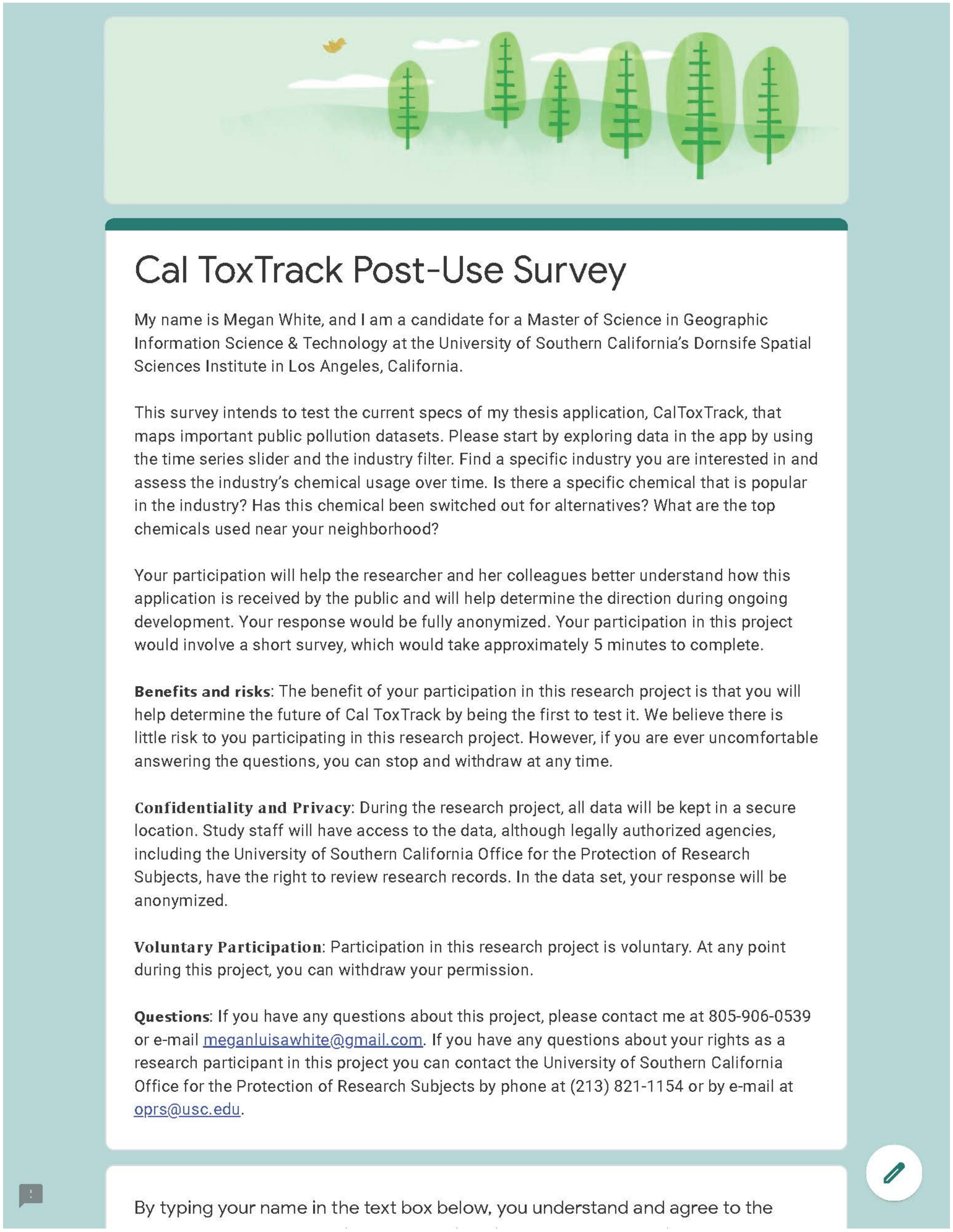


10/23/2020 urls.py

```python
from . import views
from .views import TriReleaseList, DmrReleaseList
from django.urls import include, path
from rest_framework.urlpatterns import format_suffix_patterns


urlpatterns = [
    path('tri_releases/', views.TriReleaseList.as_view(), name='tri_releases'),
    path('dmr_releases/', views.DmrReleaseList.as_view(), name='dmr_releases'),
]

urlpatterns = format_suffix_patterns(urlpatterns)

```

localhost:4649/?mode=python 1/1

Appendix B – Cal ToxTrack Evaluation Survey Questions



Cal ToxTrack Post-Use Survey

My name is Megan White, and I am a candidate for a Master of Science in Geographic Information Science & Technology at the University of Southern California's Dornsife Spatial Sciences Institute in Los Angeles, California.

This survey intends to test the current specs of my thesis application, CalToxTrack, that maps important public pollution datasets. Please start by exploring data in the app by using the time series slider and the industry filter. Find a specific industry you are interested in and assess the industry's chemical usage over time. Is there a specific chemical that is popular in the industry? Has this chemical been switched out for alternatives? What are the top chemicals used near your neighborhood?

Your participation will help the researcher and her colleagues better understand how this application is received by the public and will help determine the direction during ongoing development. Your response would be fully anonymized. Your participation in this project would involve a short survey, which would take approximately 5 minutes to complete.

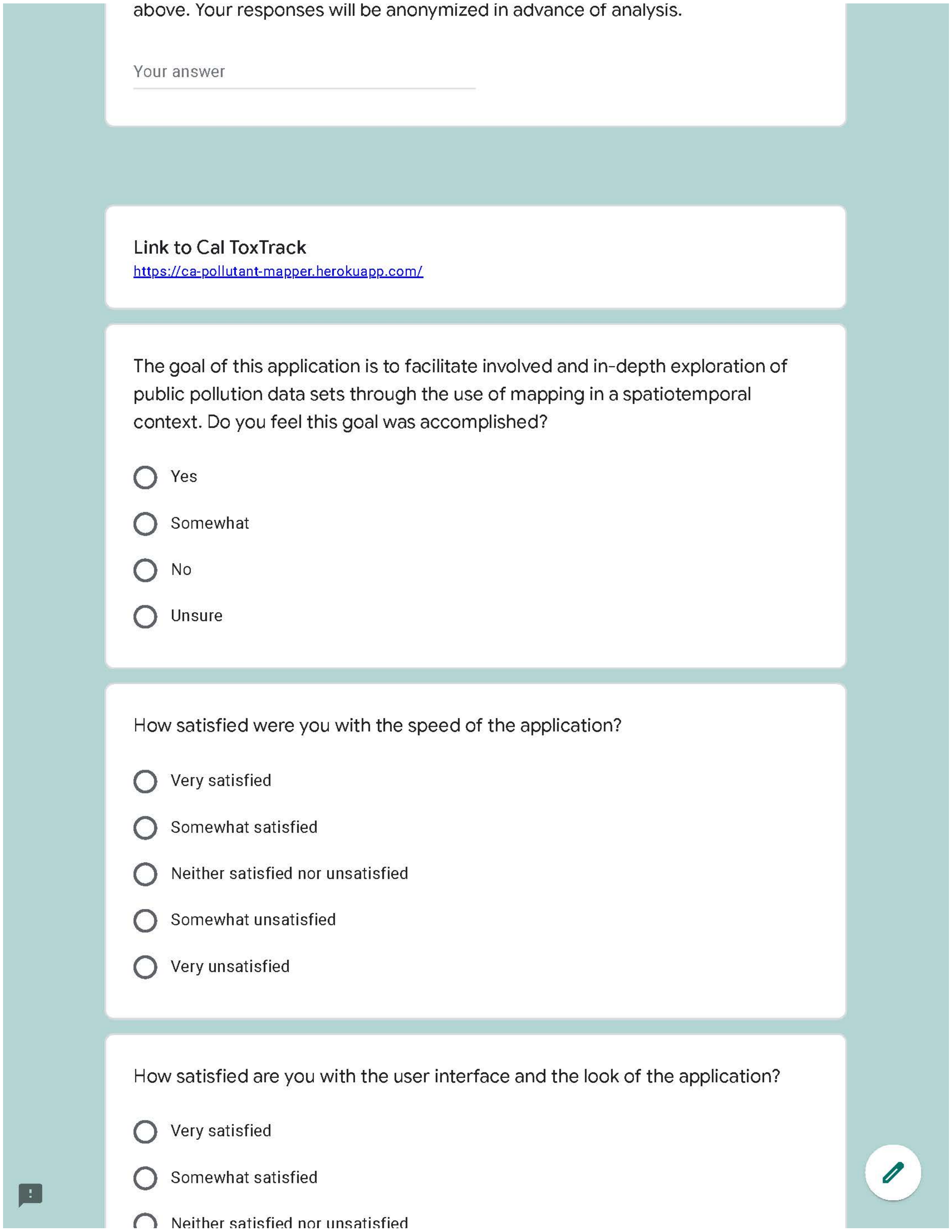
Benefits and risks: The benefit of your participation in this research project is that you will help determine the future of Cal ToxTrack by being the first to test it. We believe there is little risk to you participating in this research project. However, if you are ever uncomfortable answering the questions, you can stop and withdraw at any time.

Confidentiality and Privacy: During the research project, all data will be kept in a secure location. Study staff will have access to the data, although legally authorized agencies, including the University of Southern California Office for the Protection of Research Subjects, have the right to review research records. In the data set, your response will be anonymized.

Voluntary Participation: Participation in this research project is voluntary. At any point during this project, you can withdraw your permission.

Questions: If you have any questions about this project, please contact me at 805-906-0539 or e-mail meganluisawhite@gmail.com. If you have any questions about your rights as a research participant in this project you can contact the University of Southern California Office for the Protection of Research Subjects by phone at (213) 821-1154 or by e-mail at oprs@usc.edu.

By typing your name in the text box below, you understand and agree to the


above. Your responses will be anonymized in advance of analysis.

Your answer

Link to Cal ToxTrack
https://ca-pollutant-mapper.herokuapp.com/

The goal of this application is to facilitate involved and in-depth exploration of public pollution data sets through the use of mapping in a spatiotemporal context. Do you feel this goal was accomplished?

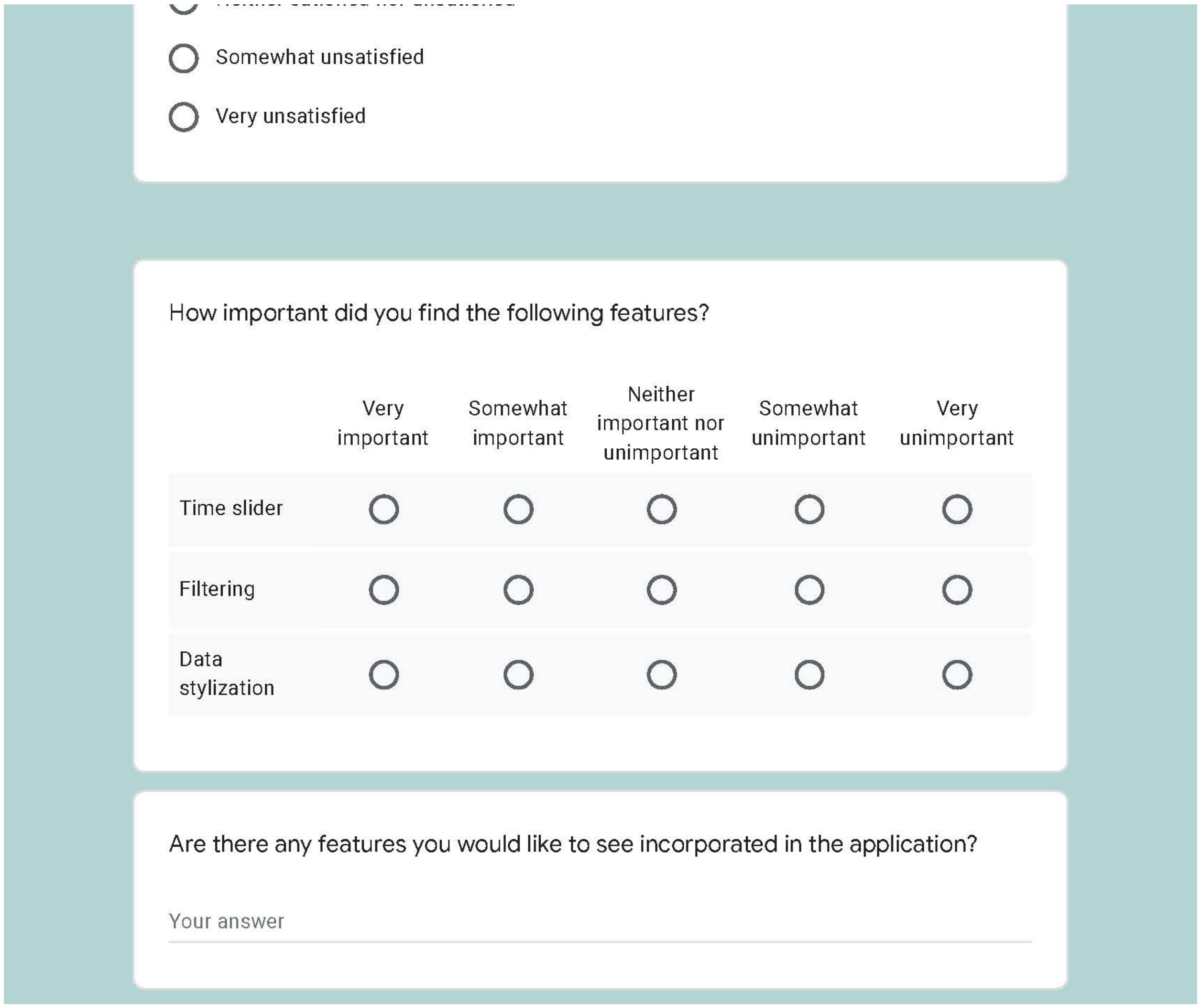
- ○ Yes
- ○ Somewhat
- ○ No
- ○ Unsure

How satisfied were you with the speed of the application?

- ○ Very satisfied
- ○ Somewhat satisfied
- ○ Neither satisfied nor unsatisfied
- ○ Somewhat unsatisfied
- ○ Very unsatisfied

How satisfied are you with the user interface and the look of the application?

- ○ Very satisfied
- ○ Somewhat satisfied
- ○ Neither satisfied nor unsatisfied



Somewhat unsatisfied

Very unsatisfied

How important did you find the following features?

	Very important	Somewhat important	Neither important nor unimportant	Somewhat unimportant	Very unimportant
Time slider	○	○	○	○	○
Filtering	○	○	○	○	○
Data stylization	○	○	○	○	○

Are there any features you would like to see incorporated in the application?

Your answer