SOAP Collaborative Assignment

Stage VII - Final Project Report

Team 1: Bryan Jimenez, Raphael Rezkalla, Ralph Quinto,
Robert Mannuzza, Jan-Lucas Ott, Spencer Viviano

CSC 415 -- Software Engineering

Inception: Team identification

Team Turing: SOAP Heatmaps

- Ralph Quinto: quintor2@tcnj.edu
- Bryan Jimenz: jimeneb1@tcnj.edu
- Jan-Lucas Ott: ottj3@tcnj.edu
- Spencer Viviano: vivians2@tcnj.edu
- Raphael Rezkalla: rezkalr1@tcnj.edu
- Robert Mannuza: mannuzr1@tcnj.edu

Inceptions: Objectives and Overview

One of the key features of SOAP is the data visualization module that can be accessed from the navigation bar. Data visualization allows for users to make sense of a large data set quickly and at a glance. SOAP currently has a Google Maps API implemented that allows users to search for specific facilities based on information like facility name, address, danger level, and longitude/latitude coordinates. Beyond that, the data visualization features of SOAP are highly simplistic and do not allow a user to draw conclusions about New Jersey's pollution from macro perspective.

A major problem in the current implementation of the data visualization module is it assumes that a user knows what he or she is looking for. An objective we hope to accomplish with the module is to provide an overview of the pollution of New Jersey and to bring the attention of the user to areas that are highly affected and need immediate attention. From there, a user can then dive deeper and use the other features on the site to research and draw his or her own conclusions.

Our goal was to create a module that would provide an at a glance view of the level of pollution in New Jersey. Our method of choice was using heatmaps via the Google Maps API. We chose this method because it is the most intuitive way to solve the problem described above. We also saw this as an opportunity to learn about databasing and web development. None of the team members had previous experience with JavaScript or the Google Maps API. The knowledge and skills obtained from this project will definitely prove to be useful moving forward in our software engineering careers and in future development endeavors.

Elaboration: Requirements Modeling and Analysis

Description of the module

The module we are designing is an implementation of a heat map to better show correlation between the data that SOAP has access to. In order to do this we're making a scalable heat map module that allows for different data points to be mapped. Since heat maps are the individual points of a matrix represented by a color it allows us to create a versatile module. So in order to map a specific data tag we have to develop an algorithm to correlate data points into matrixes. Once the data points are grouped together into matrixes they could easily be mapped by our heat map module. An example of a heatmap can be seen in Google Maps heatmap API JS tutorial. The scalability of this Heat Map module is not limited to only current data tags, as the SOAP grows so will this module with very little overhead.

14th St
Alameda St
15th St
15th
16th St
16th
17th St
17th St
Minna St
Folsom St
Bryant St
Potrero Ave
Utah St
S Van Ness
Shotwell St
Alabama St
Florida St
Hampshi
Cap

Statement about the importance and need for the module

The heat map module will give all users of SOAP improved visualization of the data already held within the SOAP database. By having this module it will allow prospective homeowners and parents the ability to visually see where the highest concentration of carcinogenic chemicals are in relation to their future home. Being able to display just carcinogenic chemicals easily gives parents the ability to make sure their children grow up in the healthiest environment. This could be easily changed to also see the concentration of a given chemical, or chemicals all used by a specific facility. Given the nature of heat maps this module would be beneficial for all users of SOAP. For policy makers to see where an abundance of particular chemicals are grouped up. A place for manufacturing companies to find locations already to polluted for residential areas to ensure they don't pollute any or grounds. This could also help with farmers finding which grounds are best for certain crops. This module only gets more use as more data is imported into SOAP.

Other similar systems / approaches

There are other developed systems with functionalities similar to the ones we wish to implement in our project. One is a US Air Quality Gradebook created by Creativemethods.com. On this platform, users are able to view the air quality levels of the different counties in all 50 states, represented by heatmaps based on the "emission" and "ambient" gradesheets provided on the website. The information represented on the maps and gradesheets; however, are based off of 1999 data obtained from the EPA online database and is therefore out of date.

Another developed system is the Air Quality Index (AQI) map created on AirNow.gov. On this platform, users are able to view up-to-date air quality forecasts, provided by the EPA and other state and local monitoring agencies, for the current day or for the following day, depicted through heatmaps. The map and forecast data shown are collected using federal reference or equivalent monitoring techniques or techniques approved by the state, local or tribal monitoring agencies. This map also features the individual pollution ratings of cities all over the country.

These systems are similar to the one we wish to implement in our SOAP module because they provided visual representations of processed data. Much like the map provided by Creativemethods.com, our map aims to distinguish the pollution levels of segmented regions of an area; however, our map would would show pollution levels as more of a gradient depiction. This model also fails to provide information relevant to today's data. The map provided by AirNow.gov uses similar methods of rating areas for pollution that we will be implementing in our project, as every individual city is given an AQI rating that contributes to macro-scale depictions of pollution levels through use of heatmaps.

The purpose of a heat map is to represent data using colors to show the different levels of use. Heat maps are easy to understand because dark colors show little usage and bright colors show high levels of use. Our heat map feature is innovative because it allows for people to analyze an area of land that has not been tested by looking at the surrounding land. A person looking to buy a piece of estate could look at our map and draw a conclusion on its level of pollution. It provides a free alternative to soil testing and would save hundreds of dollars.

Other systems have used heat maps to display areas based on emissions and air quality but none have created heat maps based on pollution or chemicals. Our heatmap would be a lot more useful because many different people could benefit from it. The EPA could use the heatmap to know where to focus its cleaning efforts. Real estate developers would also be able to use the map to avoid toxic land. Health workers could also use the map to identify illnesses based on a patient's proximity to a polluted site. All in all our heat map feature would be an addition that many would benefit from.

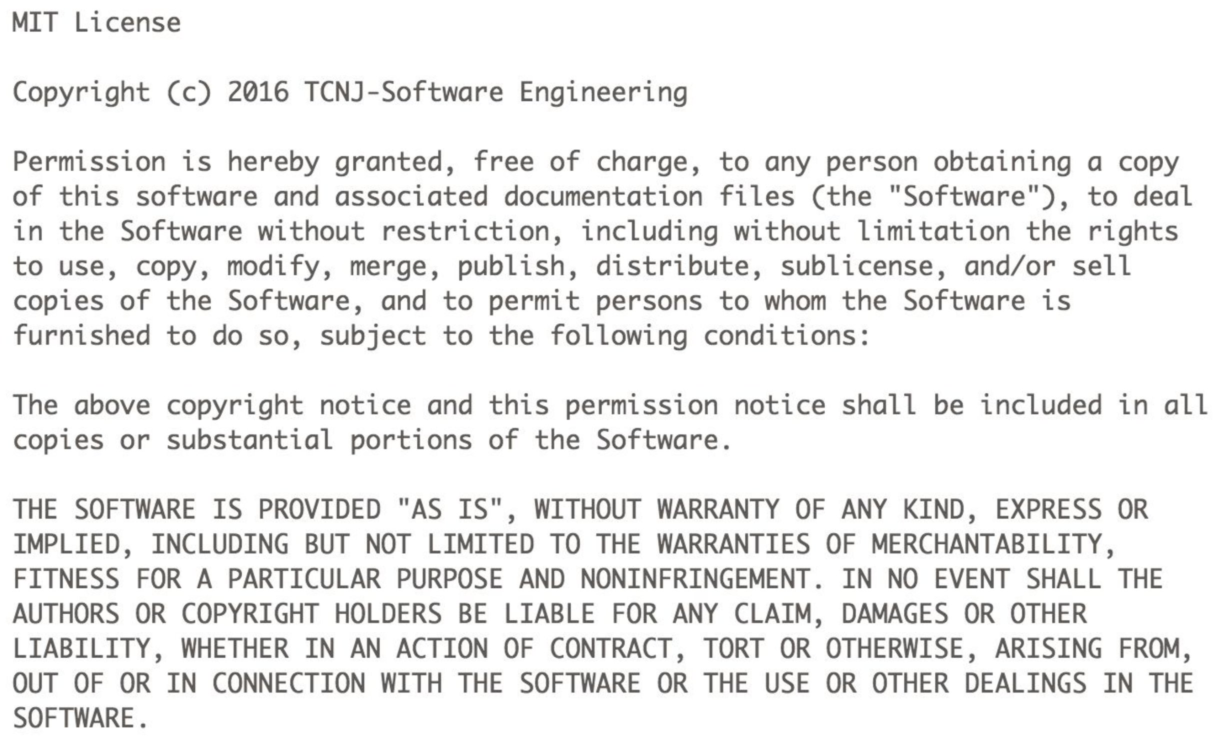
Technologies

This module will rely mainly on the Google Maps JS API to visualize a heatmap layer for different metrics on the SOAP maps page. Google's JavaScript API for Maps has a package that integrates Fusion Tables and Maps. Fusion Tables are similar to a database (like SQL); they are meant for storing large amounts of data in a quickly accessible fashion. One downside of this approach is that the data are already in the SQL database, which means it must be mirrored. However, since the scripts that update the database already pull from a csv file, the same file can be imported directly into Fusion Tables to simplify the process.

Once the data have been imported to the Fusion Table, the Maps API, through the use of JavaScript, allows querying the table for specific columns, displaying a heat map based on a given value. This also allows great scalability and flexibility. Depending on the customer needs or user interest, Fusion Tables can be used to automatically aggregate certain data. For example, we could combine the scores of all carcinogenic chemicals, and construct one heatmap based off that. Or we could combine the amounts found in water and make a one heatmap of that, and another of the chemicals found in the air and make a heatmap of that. Although we do not yet have a full understanding of which of these data are important for the client or for users, the module should be flexible enough to add these options as needed. This interface will be made using HTML, JavaScript, and possibly some PHP. This will allow the user to interact with the website which will pass their requests to the Google Maps JS API.

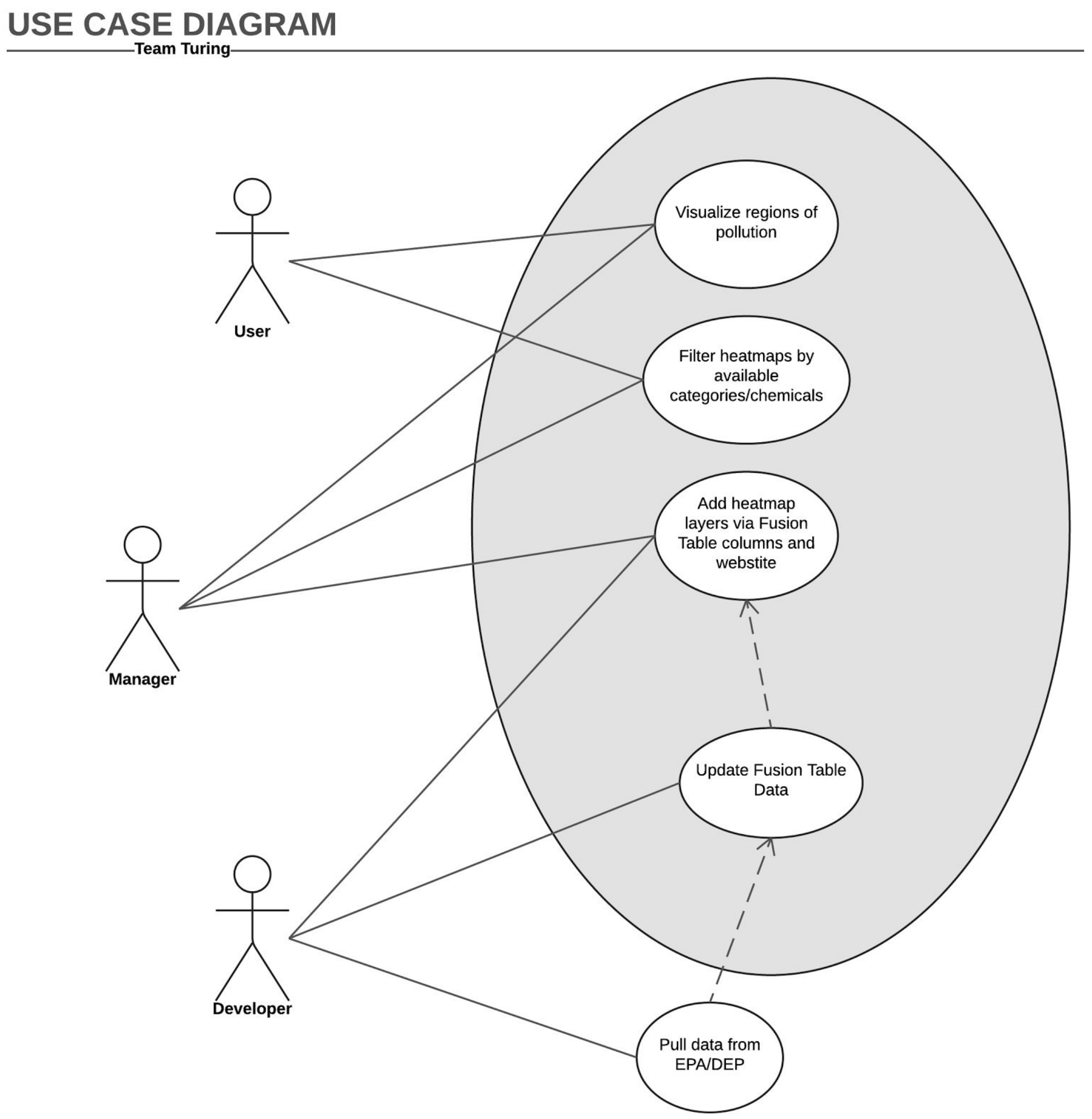
Open source license

For SOAP we are going to use the MIT Open source license. The MIT license is a permissive license that has very little limitations on reuse. It allows for SOAP to be integrated into copyrighted source code, which would allow for a government agency to utilize SOAP. The only requirement of the MIT license is when reusing code that the license is still present.



```
MIT License

Copyright (c) 2016 TCNJ-Software Engineering

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Diagrammatic Representation of the System Boundary


USE CASE DIAGRAM
Team Turing
User
Manager
Developer
Visualize regions of pollution
Filter heatmaps by available categories/chemicals
Add heatmap layers via Fusion Table columns and webstite
Update Fusion Table Data
Pull data from EPA/DEP

Security

Ensuring a secure product should be on the top of the priority list for all developers. For a system like SOAP that contains sensitive information, it is important that only the necessary people has access to the database/code. The heat map's purpose is to provide a visualization of the polluted land in NJ. Should someone decide to input wrong data or make unwanted changes, it would pose a risk to those using the system to look for safe property to purchase. We plan on implementing a robust security protocol to prevent attacks and provide a secure system.

The original design for our heatmap pulled data from the SQL database. However, after some evaluation we have decided to use a fusion table. Using a fusion table is a more secure option as general users will have no access to the data whatsoever. This will prevent any unwanted changes to be pushed through our heatmap. There would be no need to worry about compromises on our SQL accounts or SQL injections affecting our mapping.

Backup and Recovery

SOAP is a system that could have the potential of having millions of views. It is imperative to have a backup of the system should it break when adding new components. Our team has set-up a plan to ensure a copy of a working SOAP system is always at hand. As of right now SOAP is stored in eight different locations. One copy is stored in the team server, another in GitHub, and other six are being hosted in each team member's personal server. When changes are pushed on GitHub it would then be pushed onto the live team server. If we are satisfied with the changes and system still works then we can then push this new copy of SOAP onto the members' servers. If we are unsatisfied with the new version we can revert the changes by adding using a copy of a member's server.

Legal Issues

In order to implement a heat map we must utilize the Google Maps API. The Google Maps API has its own policies and license regarding the legality of using it for a business. The license states that the API may be used free and publicly if there are less than 25,000 requests per day and that no one is charged to use the service. We do not anticipate anything over 25,000 requests per day while testing. In order to avoid any infringements on Google's policies we will refer to https://developers.google.com/maps/terms#8-licenses-from-google-to-you throughout the development process. There are several policies regarding the privacy of users that must be taken into consideration when developing the module. Information that discloses the locations of brownfields and areas of toxicity is public information provided by the EPA and therefore this module will not be subjected to a claim of defamation or anything of the sort.

Possible Applications

The heatmap module would create a visual correlation of plots of land that meets the criteria given by the user. This can be used to show hotspots of areas with high concentrations of toxic chemicals in the soil. Areas that are very concentrated will show the user that surrounding areas may have similar pollutants considering the proximity of high level plots. Alternatively this module can be used to help people looking for a plot of land without any pollutants find areas with low levels of pollutants as well as areas that are not even relatively close to polluted areas. Another possible application would entail using this module to show a growing concern for the amount of pollution to influence policymakers and push to create a movement to clean the environment.

Use Case Description

Visualize regions of pollution

- By analyzing data received by SOAP from the EPA and DEP databases, users will be able to access heat map enabling features on the already existing map display. Under the 'Visualization-> Map' tab on the SOAP website, users will find a heat map filter, much like the 'Satellite' filter on Google Maps, that will allow them to initiate the module. Once the module is initiated, the user will then have access to multiple filters that will allow them to visualize different sets of data based on available categories and chemicals. The actors in this case will be both SOAP users and managers, both of which have access to the equal number of features associated in this use case.

Pull data from EPA/DEP

- The data is pulled manually from the Environmental Protection Agency (EPA) and Department of Environmental Protection (DEP) websites which is then processed through PHP scripts to validate the data. The data is then stored on our team's project server. The actor in this case will only be the developer has the ability to configure which data to pull, as well as how to store it. Note that this is done mostly manually at the moment. The only addition to the existing process will be to upload the csv to the fusion table.

Update Fusion Table Data

- We will utilize the Google Maps and Fusion Tables REST API's to aid in the processing of the data received from the SOAP database. Using the Fusion Tables REST API, the system is able to issue queries to manage these Fusion Tables to create, update, delete specific data that is to be implemented in the heat map visualization. Once the user of the module selects which data to visualize, the Fusion Tables are updated accordingly. The actors in this case will only be the developer who will configure the API to update data in a specified manner.

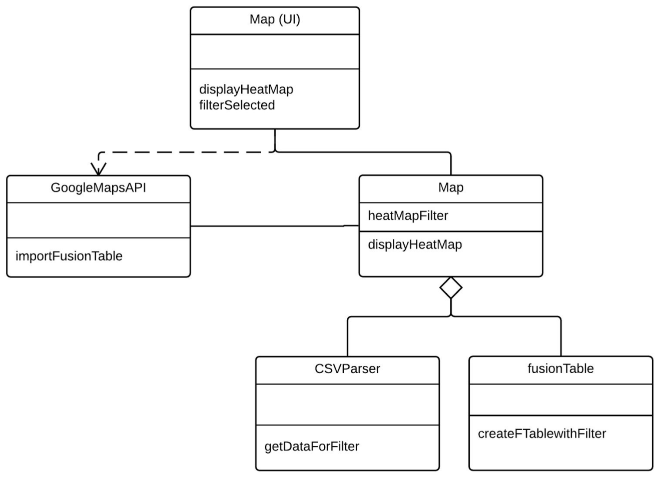
Add heatmap layers via Fusion Table columns and website

- When a user goes to select a new set of categories and/or chemicals to visualize through the heat map, the Fusion Tables that contain the represented data must be modified, either by adding or removing columns of data. This is done automatically by the system once the user has set the new parameters for visualization. The new heat map visualization is layered onto the existing map under the 'Visualization -> Map' tab on the SOAP website, only representing the parameters chosen by the user. The actors in this case will be the developer and the system manager. Developers will configure the functionality of the heat map filters by using the Fusion Tables API to execute certain tasks, such as sending web requests to the SOAP database, when prompted with a new layer request. Managers will be able to monitor and set which parameters are available for visualization.

Filter heatmaps by available categories/chemicals

- On the 'Visualization -> Map' page of the SOAP website, users will have the ability to layer a heat map onto the currently existing map visualization. Once the module is initiated by the user, they will choose a set of parameters (categories/ chemicals) to visualize through the heat map. Multiple options for filters will be available, from chemicals that induce air pollution to chemicals causing water pollution. Once a new filter is chosen, the Fusion Tables must be updated accordingly. The actors in this case are the user and the system manager. Users, in this case, will select which parameters they wish to implement for the heat map filter by selecting categories and/or chemicals which will then manifest on the map. Managers, in this case, will be able to monitor and set which parameters are available for visualization.

Analysis Class Diagram



Architecture of SOAP

SOAP follows the Model-View-Controller (MVC) design paradigm. It is built on the CakePHP web framework which, like Ruby on Rails does for Ruby, provides all the base structures and boilerplate code for doing common web tasks through the MVC pattern. CakePHP interfaces between the front-end, such as web requests and responses, the back-end, such as database access, and the application itself (SOAP), written in the middle via PHP.

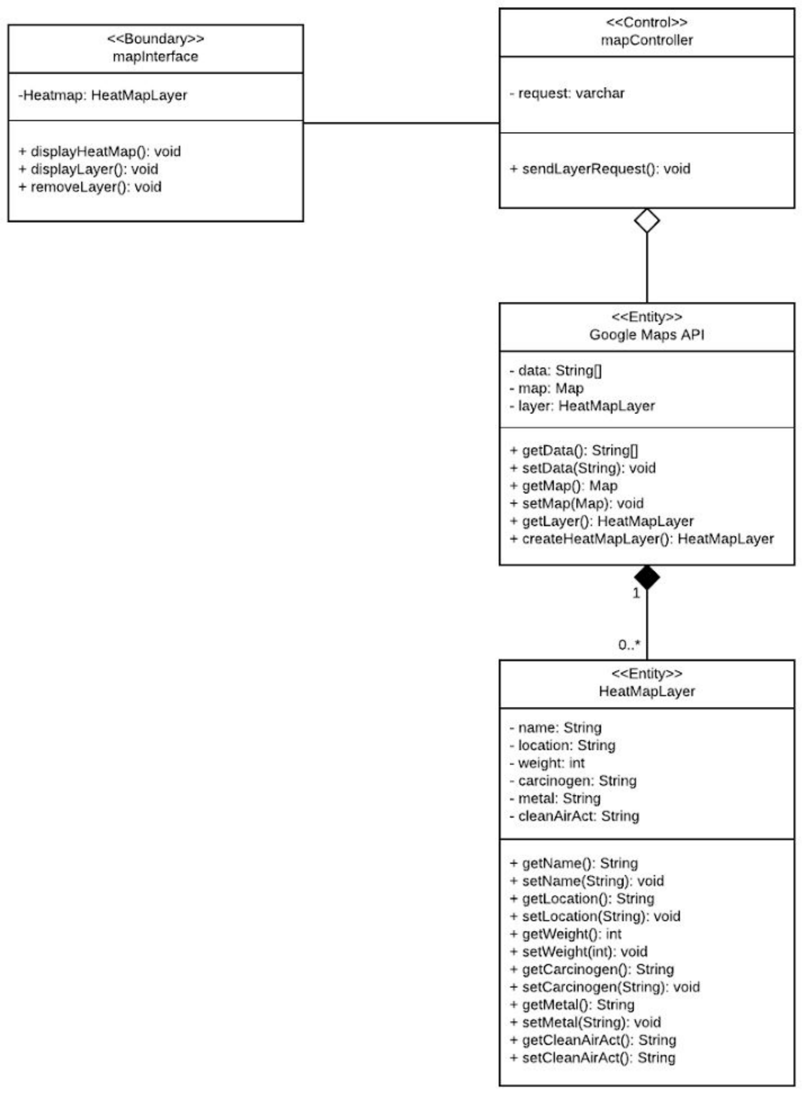
It handles web requests in a RESTful manner via "routing". The routes file (app/Config/routes.php) can be configured to link certain web requests (eg "GET /chemicals") to

Controllers (represented by a PHP file in app/Controller). The controller is responsible for taking the request data and handling it however necessary. It must then return the View (which are defined as subdirectories in app/View). A view consists of one or more ".ctp" files, which are Cake template files that allow HTML (which will be sent as a response to the user's web browser to display). These templates can be modified by the controller before getting sent back as a response. Controllers may also utilize Models (defined in app/Model) as classes which may be used to represent data found in a View.

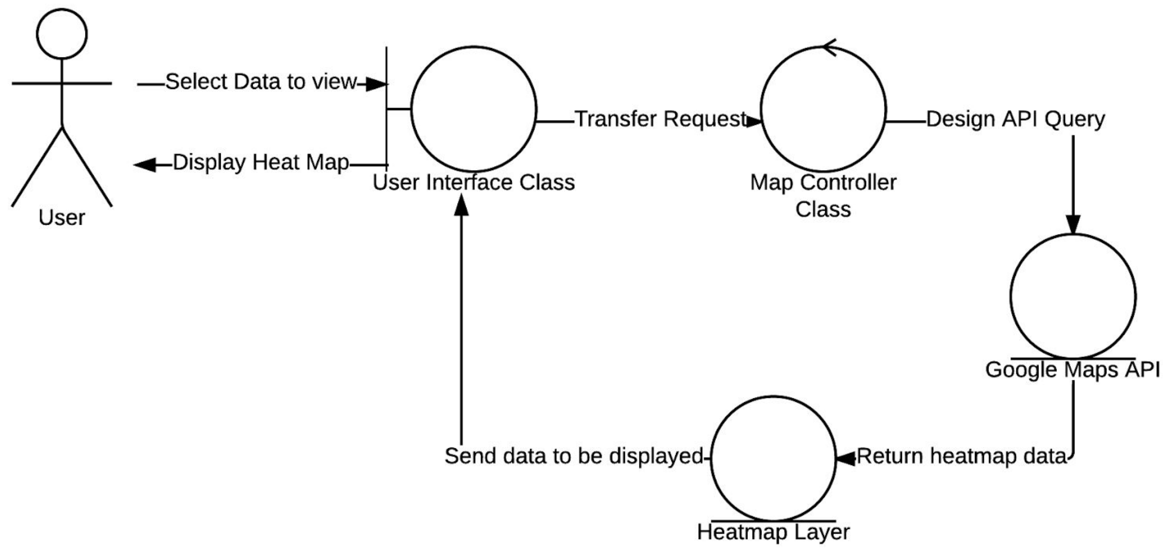
Our module, as seen in the class diagram above, will follow this structure. The user interface class will be the View, while the controller will be responsible for creating the correct JavaScript query for the fusion table and performing any necessary calculations to filter the data. A Model may be used to store the necessary data and parameters for the query and heatmap data.

Elaboration: Design

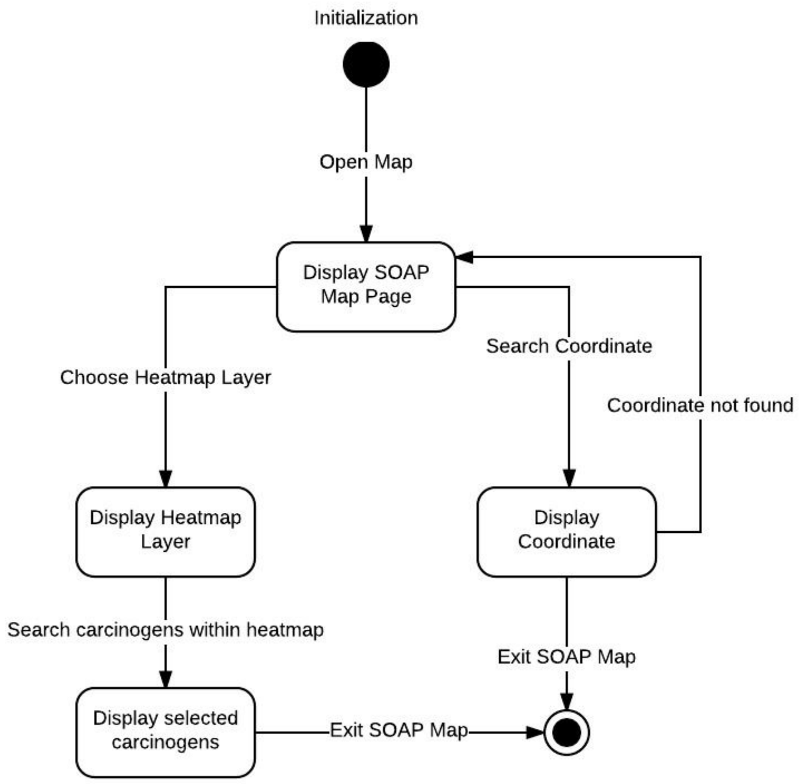
Detailed Design Class Diagram



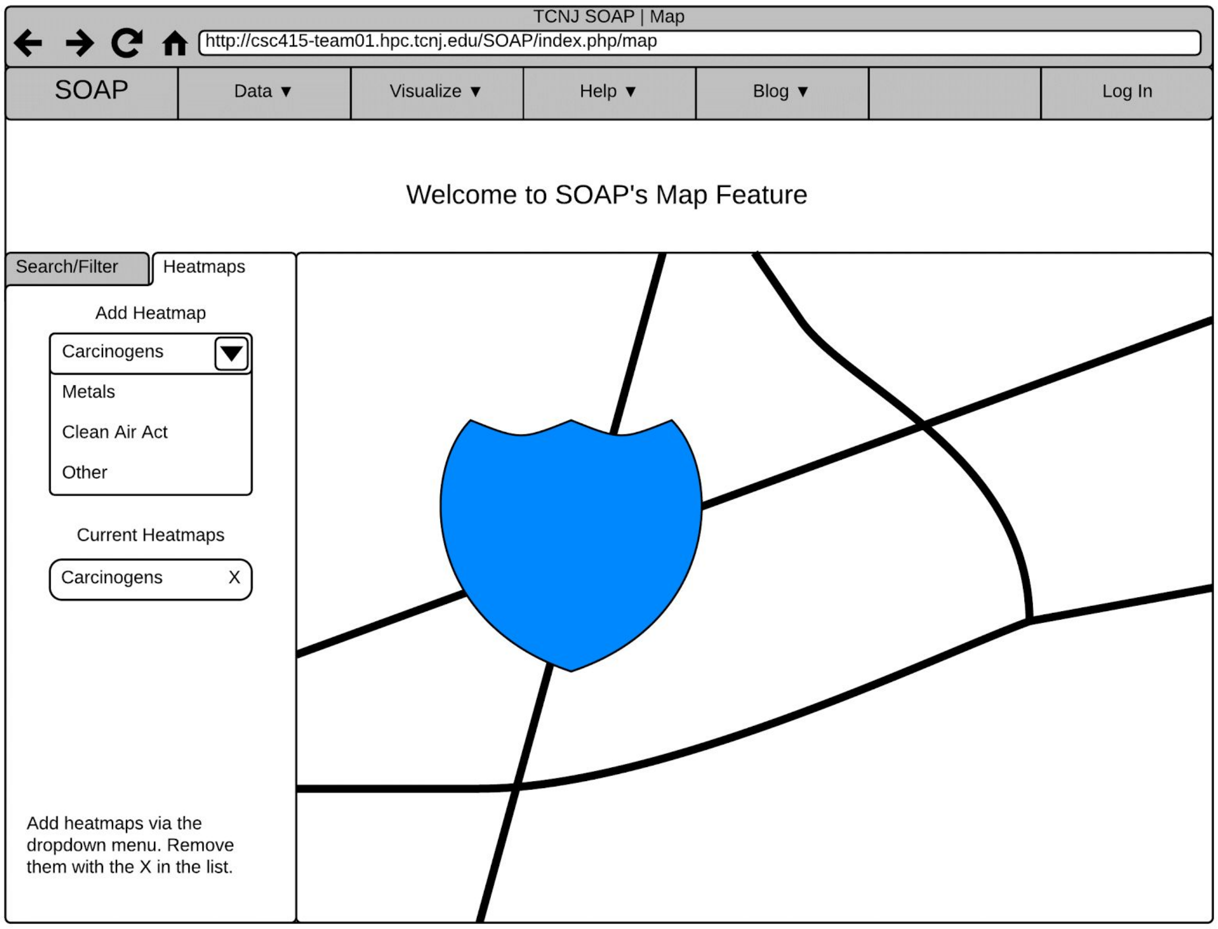
Collaboration Diagram



Detailed Statechart



Interface Design


TCNJ SOAP | Map
http://csc415-team01.hpc.tcnj.edu/SOAP/index.php/map
SOAP
Data ▼
Visualize ▼
Help ▼
Blog ▼
Log In
Welcome to SOAP's Map Feature
Search/Filter
Heatmaps
Add Heatmap
Carcinogens
Metals
Clean Air Act
Other
Current Heatmaps
Carcinogens X
Add heatmaps via the dropdown menu. Remove them with the X in the list.

8 Golden Rules

Strive for Consistency

The first thing we noted when analyzing SOAP and how we intended on implementing our module was how the previous features were implemented on the site. We noted how drop down menus and tabs were used to navigate the map that was embedded into the site. We are striving for consistency by using the same map and adding our heat map to it. Rather than using radio buttons for the various heat maps available, we are using drop down menus to maintain consistency. Additionally, the new feature will be accessed via a tab, which aligns with the original design of the website.

Offer informative feedback

Our module will offer informative feedback through the refreshing of the map after an option has been selected. None of the heat maps will display the same information, so seeing the colors change with the options will be visual feedback to the user informing him/her that his/her input has been accepted. It's important not to overdue this rule via popups or messages because it could detract from the functionality and flow of the site.

Design dialog to yield closure
To meet this rule of interface design, we intend on highlighting the option that was selected. Our form of showing design dialog to yield closure is having the option selected after the heat map changes. We don't believe a dialog box is appropriate for this module because it would interfere with the functionality. A huge asset of this module is it being highly responsive and easily accessible, a dialog box would only interfere with this flow.

Offer simple error handling
Our module combats errors by limiting what the user can input into the module. Our options are closed ended, so the user cannot input something that would cause the system to break or cause an error. Regarding the closed ended options, they will be tested extensively to ensure that they are working properly and interface with SOAP as intended.

Permit easy reversal of actions
Because the options are available via a drop down menu, easy reversal of actions is possible. If a user selects the wrong option by mistake, he or she can easily reverse this action by picking the option they originally intended. Additionally, once a heatmap layer is added, it can be removed again easily.

Support internal locus of control
Our design supports a user's internal locus of control because it presents all functionality as closed-ended options. The design is very intuitive and simple to learn. Every input in the module is mapped to a single output, so there are no surprises. As a result, the user feels like he or she is always in control.

Reduce short-term memory load
The choice to use drop down menus and tabs to navigate the module ensures that a user never has to remember which option was selected or what options are available in the module. All information in the module is presented as-is, front and centered.

Enable Frequent Users to Use Shortcuts
Due to the simplicity of the design, there aren't necessarily explicit shortcuts that a user can use to navigate the module. However, as a user becomes acquainted with the module and the options that are available, he or she will use the module at a faster pace. For example, a user may know that the option he or she is looking for is the second option in the drop down menu, so they can use keyboard inputs to quickly navigate to that option.

Construction: Implementation

Link to our github team repository: https://github.com/TCNJSEteam1/SOAP

Construction: Testing

Code Review

Elisa Idrobo:

File: App/webroot/js/map.js

- When creating heatmaps variable documentation should specify what is being queried since it is not querying soap.
- threeIndexRox function documentation should specify what the parameter row is.
- Error handling is implemented with informative messages
- Good use of outside API's

Overall the code looks good-there are appropriate error messages and there are no obvious inefficiencies. However, some of the documentation could be more detailed and the code could be more modular. Currently there is a lot going on in the Map.js file. The heatmap portions could be separated from the code that creates the map and sets markers.

Felipe Mardones:

- Some comments are missing, making understanding of portions a little difficult.
- Separate the heatlayer information into a separate class (low coupling, high cohesion approach).
- Include error-handling somewhere. Currently, there is no error handling, which could cause problems.
- Use SOAP's database for information instead of csv files.

Keenan Sayers

Reviewing file: /SOAP/app/webroot/js/map.js

- Well commented and properly documented

- More heat maps should be added for better visualization of all environmental contamination

-No user interface features added, functions must be initialized through console

-Current user interface features are clunky, and should be more fluid.

Testing Reports

Elisa Idrobo:

Functionality Tested	**Inputs**	**Expected Output**	**Actual Output**
Displaying carcinogen heatmap layer	Press Carcinogen button	Carcinogen heatmap displayed successfully on Google Map	There is a heatmap displayed
Removing carcinogen layer	Press Carcinogen button	Carcinogen heatmap removed successfully from Google Map	Heatmap disappears
Displaying lead heatmap layer	Press Lead button	Lead heatmap displayed successfully on Google Map	Heatmap is displayed
Removing lead layer	Press lead button	Lead heatmap removed successfully from Google Map	Heatmap disappears
Displaying release into water	Press Release into water button	Release to water heatmap successfully displayed on Google Map	Heatmap is displayed
Removing release into water	Press release into water button	Release to water heatmap removed successfully from Google Map	Heatmap disappears

Display more than one layer at a time	Press multiple heatmap buttons	The heatmaps should overlay each other	Multiple heatmaps are shown

Comments: UI works following specifications. The color changes showing selection vs deselection is unintuitive. The heatmaps are created and displayed with no errors.

Keenan Sayers:

Functionality Tested	**Inputs**	**Expected Output**	**Actual Output**
Displaying carcinogen heatmap layer	Press Carcinogen button	Carcinogen heatmap displayed successfully on Google Map	Works as expected.
Removing carcinogen layer	Press Carcinogen button	Carcinogen heatmap removed successfully from Google Map	Works as expected.
Displaying lead heatmap layer	Press Lead button	Lead heatmap displayed successfully on Google Map	Works as expected.
Removing lead layer	Press lead button	Lead heatmap removed successfully from Google Map	Works as expected.
Displaying release into water	Press Release into water button	Release to water heatmap successfully displayed on Google Map	Works as expected.

Removing release into water	Press release into water button	Release to water heatmap removed successfully from Google Map	Works as expected.
Display more than one layer at a time	Press multiple heatmap buttons	The heatmaps should overlay each other	Works as expected.

Patrick Roderman:

Functionality Tested	**Inputs**	**Expected Output**	**Actual Output**
Displaying carcinogen heatmap layer	Press Carcinogen button	Carcinogen heatmap displayed successfully on Google Map	Carcinogen heatmap displays as overlay
Removing carcinogen layer	Press Carcinogen button	Carcinogen heatmap removed successfully from Google Map	Carcinogen heatmap overlay no longer is visible
Displaying lead heatmap layer	Press Lead button	Lead heatmap displayed successfully on Google Map	Lead heatmap displays as overlay
Removing lead layer	Press lead button	Lead heatmap removed successfully from Google Map	Lead heatmap overlay no longer is visible
Displaying release into water	Press Release into water button	Release to water heatmap successfully	Release heatmap displays as overlay

		displayed on Google Map	
Removing release into water	Press release into water button	Release to water heatmap removed successfully from Google Map	Release heatmap no longer is visible
Display more than one layer at a time	Press multiple heatmap buttons	The heatmaps should overlay each other	Heatmaps are displayed and overlayed on top of each other

Felipe Mardones:

Functionality Tested	**Inputs**	**Expected Output**	**Actual Output**
Displaying carcinogen heatmap layer	Press Carcinogen button	Carcinogen heatmap displayed successfully on Google Map	Displays carcinogen heatmap; unknown if correct
Removing carcinogen layer	Press Carcinogen button	Carcinogen heatmap removed successfully from Google Map	Successfully removed carcinogen heatmap
Displaying lead heatmap layer	Press Lead button	Lead heatmap displayed successfully on Google Map	Displays lead heatmap; unknown if correct
Removing lead layer	Press lead button	Lead heatmap removed successfully from Google Map	Successfully removed lead heatmap
Displaying release	Press Release into	Release to water	Displays released to

into water	water button	heatmap successfully displayed on Google Map	water; unknown if correct information
Removing release into water	Press release into water button	Release to water heatmap removed successfully from Google Map	Successfully removed information
Display more than one layer at a time	Press multiple heatmap buttons	The heatmaps should overlay each other	Heatmaps overlay on each other with same colors; no differentiation between layers

Summary:

Functionality functioned functionally. The test cases provided were appropriate and ran as expected. For future work, check the database information to see if heatmap is accurate. Additionally, if database does not have all of the necessary info, make sure that the heatmap function doesn't crash. If possible, try heat maps for different chemicals individually.

Edward Kennedy

Functionality Tested	**Inputs**	**Expected Output**	**Actual Output**
Displaying carcinogen heatmap layer	**Press Carcinogen button**	**Carcinogen heatmap displayed successfully on Google Map**	**Heatmap displayed**
Removing carcinogen layer	**Press Carcinogen button**	**Carcinogen heatmap removed successfully from Google Map**	**Heatmap removed**
Displaying lead heatmap layer	**Press Lead button**	**Lead heatmap displayed successfully on Google Map**	**Heatmap displayed**
Removing lead layer	**Press lead button**	**Lead heatmap removed successfully from Google Map**	**Heatmap removed**
Displaying release into water	**Press Release into water button**	**Release to water heatmap successfully displayed on Google Map**	**Heatmap displayed**
Removing release into water	**Press release into water button**	**Release to water heatmap removed successfully from Google Map**	**Heatmap removed**

Display more than one layer at a time	**Press multiple heatmap buttons**	**The heatmaps should overlay each other**	**Displays multiple layers, one for each selected heatmap**

Transition: Maintenance

All of our source code is uploaded to our Github team01 page. The methods we added onto the different files contains comments that explains each functionality.

Github Page: https://github.com/TCNJSEteam1/SOAP

Transition: Product Hand Over

Manual

To access our heat map the user would go to the SOAP homepage and click on 'Visualize' tab located at the top of the webpage. From that drop down menu the user will select the 'Map' and it will load the Maps page. By default there will be no heat map layers enabled. To display the different heatmap layers, the user would go the side tab and click on the Heatmap Overlays button. Doing so would the expand the panel and display 3 buttons that for the Carcinogen, Lead, and Release into water heatmaps. In order to enable the Carcinogen heatmap the user would have click on the Carcinogen button. To disable the heatmap layer for carcinogens all the user would have to do is click the Carcinogen button again. The process for toggling the other two heatmaps is the same as the Carcinogens. The system allows for multiple layers to be overlayed at once so if the user chooses to do so he or she could. Should a future group decide to add more heatmap layers to the system, all they would have to do is go to the map.js file. From there the user would then create an instance of the heatmap variable and define it with the desired pollutant data.