TCNJ
THE COLLEGE OF
NEW JERSEY

SOAP Collaborative Assignment
Stage III - Requirements Modeling
Team 1: Bryan Jimenez, Raphael Rezkalla, Ralph Quinto,
Robert Mannuzza, Jan-Lucas Ott, Spencer Viviano
CSC 415 -- Software Engineering

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 first step is to have different levels of permissions when accessing the database. Every user in SQL is added to the public database role by default. These users will only have basic access and will not be able to read restricted data. The six members on our team will be granted higher levels of access with each assigned different roles. It would be more convenient to be granted all levels of access for all the team members but one account being compromised would put the entire system at risk. The next step is preventing attacks such as SQL injections. There is a multitude of ways in preventing SQL injections with limiting database privileges being one of them. We also plan to employ data sanitization to filter all input. Characters would be limited to the context of the form. For example the inputs for the login page should only allow number and alphabet characters. Ideally we would want to remove any SQL queries that involve user input as data sanitization is not perfect.

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

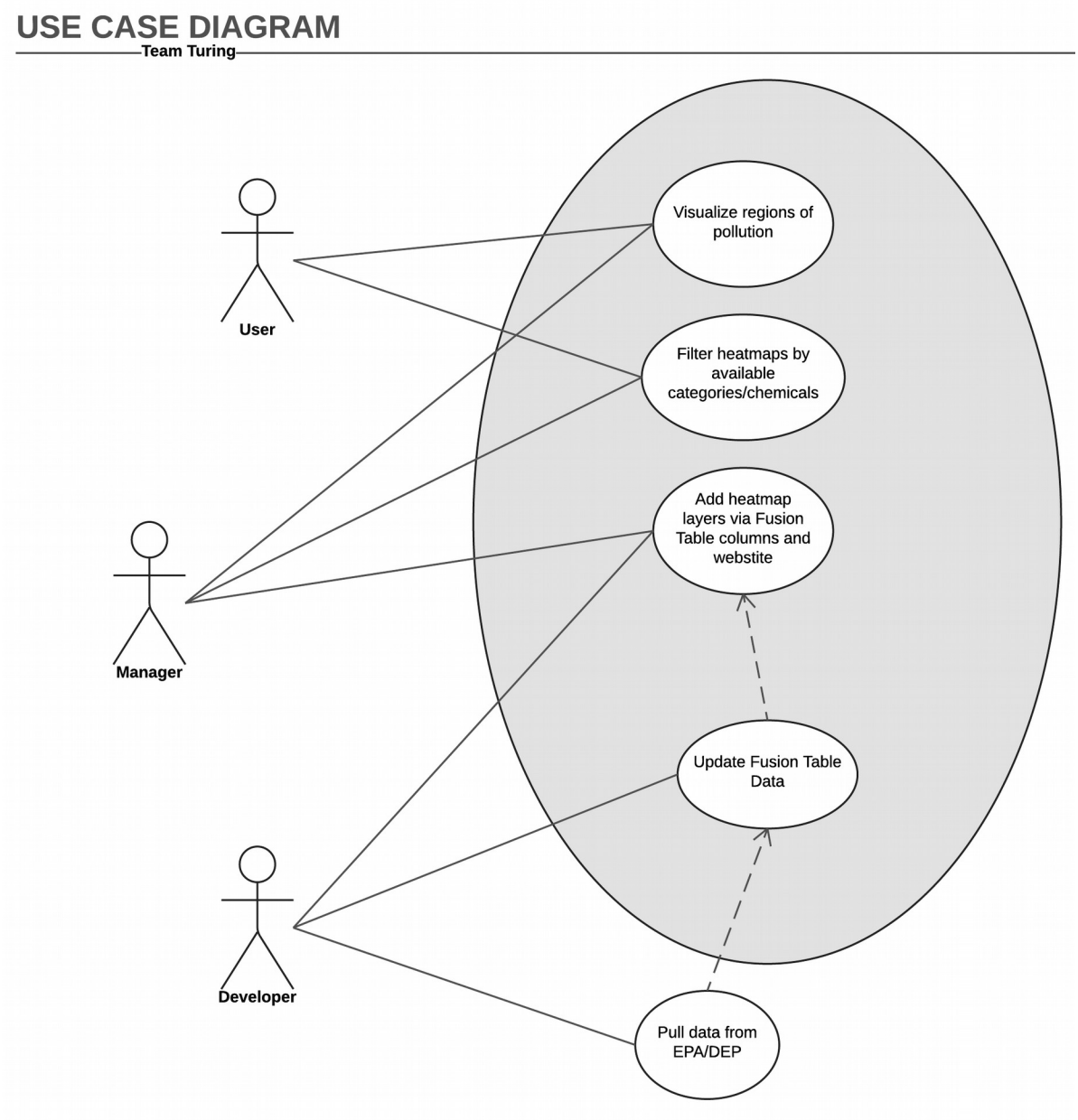
In order to implement a heat map we must utilize the Google Maps API. The Google Maps API has its own policies and license regarding the legality of using it for a business. The license states that the API may be used free and publicly if there are less than 2,500 requests per day and that no one is charged to use the service. 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 Diagram



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.

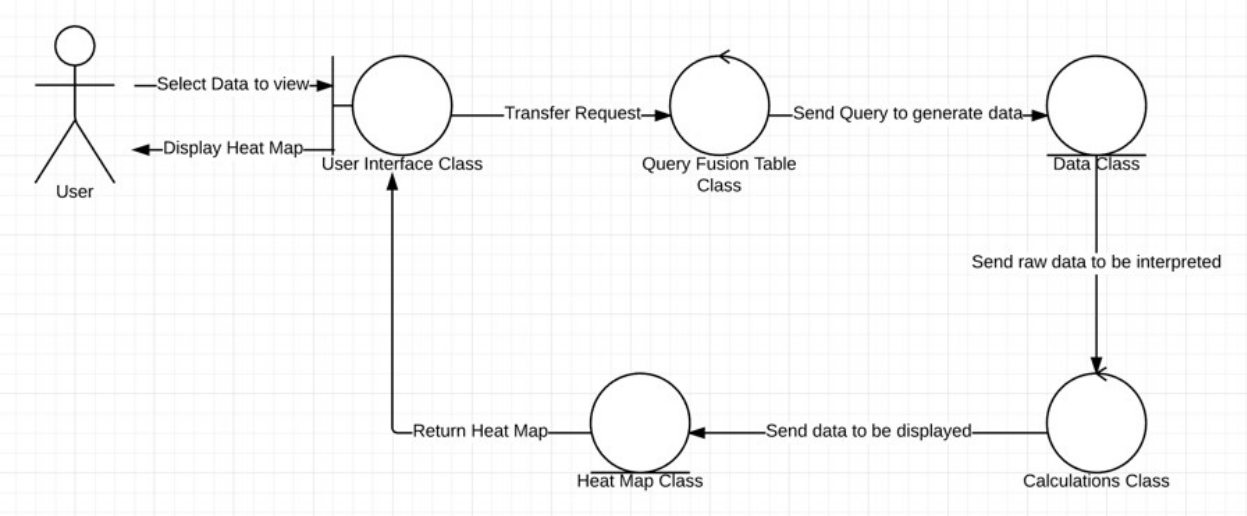
Add heatmap layers via Fusion Table columns and website

- 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.

Documentation for the Main Module

Documentation has been added to the GitHub repository.