Spatial Visualization of Biodiversity

Ty Skelton, Jasper LaFortune, and Alex Shields

Abstract

The Department of Integrative Biology at Oregon State has collected a large sample of biodiversity data from various sites in the Southwest United States. Handling this data in its raw form requires certain technical knowledge of databases, as well as a bit of patience. This presents a problem for biodiversity researchers and Department of Defense land managers, who need to be able to understand and make decisions about this data easily. Our team addressed this problem by creating a web interface for spatially visualizing this data. We enabled users to easily display useful graphs and maps about areas and species of interest, putting the information they care about most at their fingertips.

The Department of Defense made an investment in collecting all these samples so that it could better manage its land. However, extracting meaning from that information is challenging for land managers and researchers alike. Any good solution to this problem would allow users to easily access the information that is important to them. Our solution provides an interface that allows users to select information of interest and see it in map and graph form. Passerby at Expo will be invited to interact with our system and discover meaningful biodiversity patterns for themselves. Oregon State biodiversity researchers have indicated that our product meets their needs.

CONTENTS

I. Introduction

Professor Dave Lytle, head of the Lytle Lab in the Department of Integrative Biology at Oregon State University, requested this project. His research team had collected five years of insect samples from Department of Defense bases and amalgamated the observations into a large, somewhat messy database. He asked our team for a visualization system to help make sense of the data. Such a system would aid researchers in data analysis, and help the Department of Defense to make informed land management decisions. Dr. Lytle and his research team sponsored the project and acted as clients for the product. We met with them periodically throughout each term to go over progress and goals. They were, of course, involved in design decisions, but left the implementation details primarily to us.

Our team consists of Jasper LaFortune, Alex Shields, and Ty Skelton. We each had a distinct role in development and team leadership. Jasper developed the Filter View and led the team's presentations, including organizing the poster, video reports, and elevator pitch. Alex developed the Map View and led the team's external communications, including contacting our clients and submitting assignments. Ty developed the Graph View and led the team's internal communications, including organizing meetings and individual tasks. Development and leadership responsibilities were shared fairly among all three members.

II. Original Requirements

A. Project Description

This project will deliver a web application that will be used to interactively visualize biodiversity data. This application will be used by researchers and Department of Defense land managers to compare and analyze biodiversity data in areas of interest.

B. Service Need

Professionals working out in the field on government-owned property are tasked with preserving the ecosystems within them. To serve this purpose, theyve requested a tool with which they can quickly assess biodiversity levels of different key areas.

C. Client Requirements

Our client will provide us with the data to visualize and a web server on which to host the application.

- The dataset will provide each individual observation of an insect, including the location, time, and taxa of the species observed.
  - The location will be one of about 700 sites in the Southwestern United States, and will include latitude and longitude. Sites are hierarchically organized in that each site belongs to a basin.
  - The time will be between 2009 and 2013. Samples were collected at individual sites anywhere from only once to twice a year.
  - The taxa will be the fully qualified taxonomy of the insect species observed. These will be hierarchically organized, as every species belongs to a genus, which belongs to a family, and order, and so on.

D. Functional Requirements

We will create a web interface for interacting with and visualizing the dataset provided to us. This interface will provide three views:

- Map View
  - This view will display a map, annotated with markers at the locations of all the sites where data have been collected.
  - Users will be able to pan and zoom in this view.
  - They will also be able to select a site in this view by clicking on it.
  - As a stretch goal, we will display markers with size or color according to biodiversity metrics of that site.
- Statistics View
  - This view will display useful summary statistics, graphs, and charts about the selected information. These will include:
    - A graph of biodiversity over time.
    - The Shannon diversity score.
    - A chart of insect population by taxa or species.
- Filter view
  - This view will allow the user to filter what information is displayed in the Map and Statistics Views. It will contain:
    - Date filter
      - This will allow the user to select data from a range of dates.
    - Taxa filter
      - This will allow the user to select multiple taxa (e.g. species, order).
      - The species will be organized hierarchically within the pane.
    - Site filter
      - This will allow users to select multiple sites.
      - The sites will be organized hierarchically (into their containing basins) within the pane.
- STRETCH GOAL Advanced statistical analysis

- As a stretch goal, this feature will go beyond the data manipulation implemented in the different graphing features of the main app. We will work with David Lytle and his team to generate advanced statistical models on the data theyve gathered. Since this is a stretch goal, it is currently undefined and will only be reached once we complete the 1.0 version of our website.

E. Technical Requirements

- Set up mapping tool
  - Set up a mapping tool (e.g. ArcGIS) to create the map view.
- Test cross-browser support
  - We will test our application to support common browsers such as Chrome, Firefox, Internet Explorer, and Edge.
- Set up web server
  - We will set up a web server to host our application. We will either acquire a server or use a web hosting service.
- Set up website
  - We will create a base website to house our application. We will build this with a web framework to do the heavy lifting for basic needs such as user control, security, database access, and templating.
- Set up database
  - We will get this data in its raw form from our client. We will then convert their data into a form that can be stored in a relational database.
- Create In-depth documentation
  - We will provide sufficient technical documentation of our application for another developer to make extensions and additions to it.

F. Communication Plan

We will use the following for our internal communication:

- Slack
  - We will use this chat client for coordinating meeting times.
- GitHub
  - We will keep version control of our code on GitHub.
- Regular meetings several times a week
  - In order to make regular process towards goals that have been assigned and set by ourselves, we will meet regularly and consistently re-evaluate our progress.

We will use the following for our external communication:

- Email will be our primary means of contact with our client.
- We will meet twice a month with our TA.
- We will meet twice a month with our client.

G. Documentation

Technical Documentation will be provided by the following:

- Microsoft Sharepoint
  - As per our class requirements, weve created a sharepoint site that will be the primary form of providing updates on progress and holding all revisions of our code base and documentation. This is shared amongst the client, the developer team, and the professor/TAs.
- Github
  - While the sharepoint site will be where all code versions will be hosted officially, Github will be the tool the development team will use for version control internally. Documentation will be held here as well, but replicated in the sharepoint site.

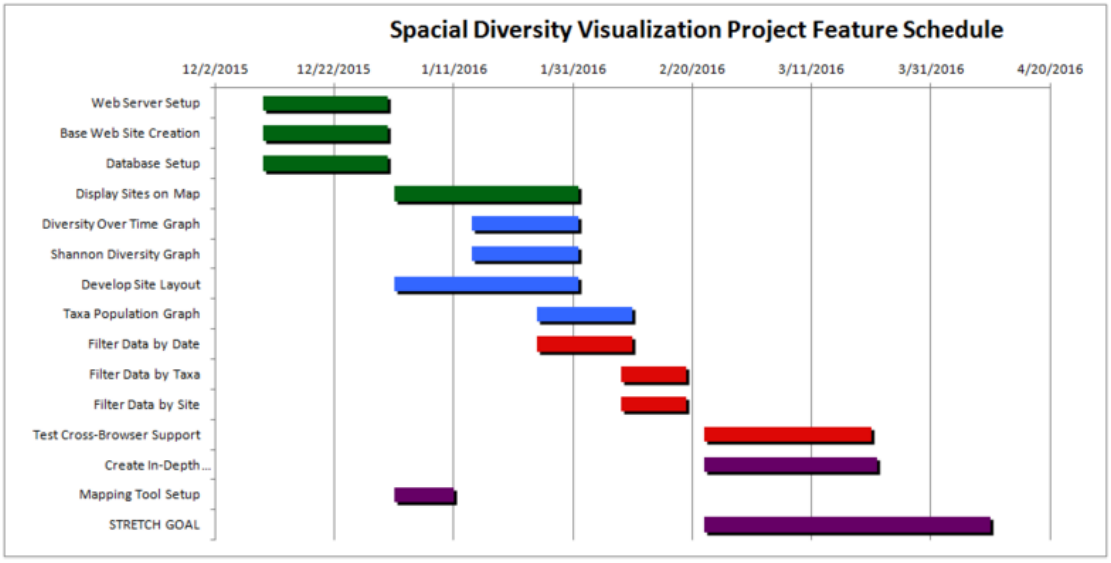

Fig. 1: This is our original speculated gantt chart. It displays the time estimates that we had originally assumed various required tasks would take. As you can see it had a fairly smooth flow to it, which we learned soon in was very different in practice.

H. Timeline

Timeline showed in Fig 3.

I. Signature

This looks good to me as a starting document. It doesn't have much about the Obj 2 statistical parts, but we can add those later. Consider this email my signature on the document. Dave+ ********** David A. Lytle Oregon State University (541) 737-1068

III. Changes to Original Requirements

Resulting gantt chart shown in Fig 2

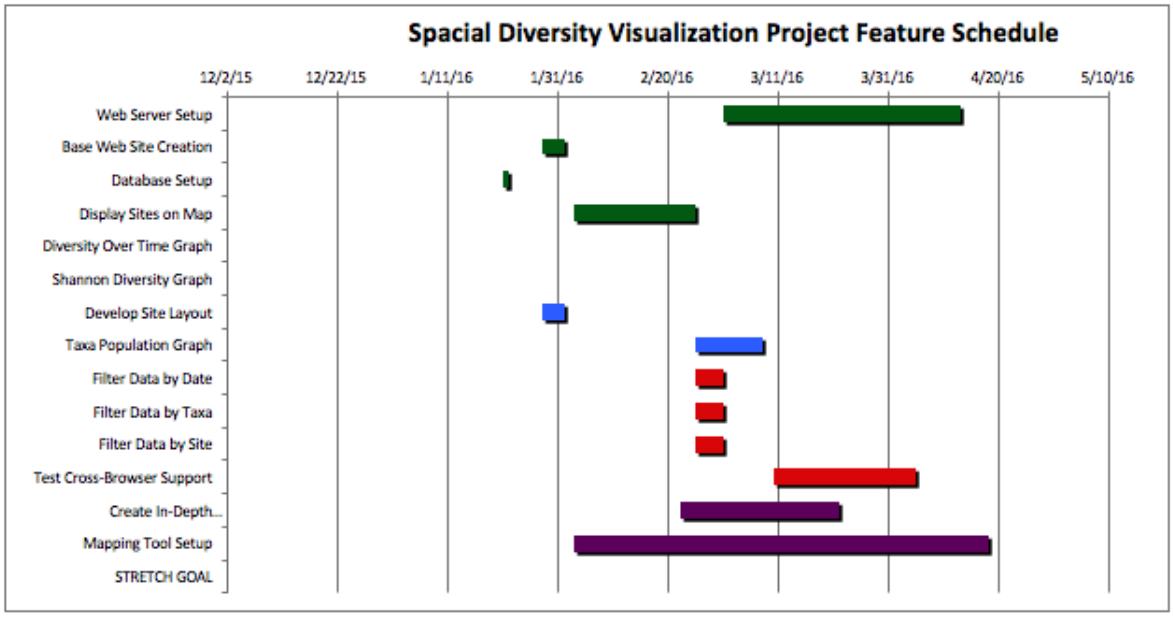

Fig. 2: Our resulting gantt chart after tracking the assign and completion dates for issues in Github. When comparing it to our speculated Gantt chart it is much less waterfall-esque, but taught us a valuable lesson in how sometimes certain aspects have to come before others.

IV. Original Design Document

A. Introduction

The Department of Integrative Biology at Oregon State has collected a large sample of biodiversity data from various sites in the United States. Handling this data in its raw form requires certain technical knowledge of databases, as well as bit of patience. This presents a problem for biodiversity researchers and Department of Defense land managers, who need to be able to understand and make decisions about this data easily. Our team will address this problem by creating a web interface for spatially visualizing this data. We will enable users to easily display useful statistics, graphs, and maps about areas and species of interest, putting the information they care about most at their fingertips.

The Department of Defense made an investment in collecting all these samples so that it could better manage its land. However, extracting meaning from that information is challenging for land managers and researchers alike. A good solution to this problem would allow users to easily access the information that is important to them. Our solution will provide an interactive map overlaid with tools that allow users to easily select information of interest and display useful graphs and statistics about it. Passerby at Expo will be invited to interact with our system and discover meaningful biodiversity information for themselves. Oregon State biodiversity researchers and Department of Defense land managers will provide feedback on how well our product meets their needs.

B. Project Plan

1) Requirements: We will create a web interface for interacting with and visualizing the dataset provided to us. This interface will be comprised of three main views: the map view, the statistics view, and the filter view.

The map view displays the region of interest on a map annotated with markers showing where data has been collected. The user will be able to use this map to select the specific sites and areas they are interested in. The sites that the user selects will be used to populate the charts and graphs so that they can see what is going on in that area. This view will require the use of mapping software and a database in order to create the map itself and to populate it with markers corresponding to each site.

The statistics view takes the data from sites selected in the map view and renders statics, maps, and charts that make it easy to see trends and understand the data. Examples of these are: a graph of biodiversity over time, a Shannon diversity score, and a chart of the insect population by taxa or species. This will be accompanied by any further kinds of statistics and comparisons that they feel would be useful for interpreting the data. We will be creating these graphs and charts using NumPy/SciPy for doing the scientific calculations and using D3 to render the graphs and charts themselves. The statistics view will also require that the data be in an accessible database form.

The filter view will display options that allow users to filter the data in the map and statistics views. There will be several of these filters that can all be used together to give the user the exact data that they want. The date filter allows users to specify what timeframe they are interested in. The taxa filter will allow for viewing data related only to the selected taxa which includes species, order, genus, et cetera. The site filter will allow users to select multiple sites with a unifying factor, such as all being in the same drainage basin.

2) Timeline: The gantt chart above displays the plan for developing this website over the course of the next two terms. We will start by setting up the foundation of the site by configuring the web server, importing the database, and deploying the basic framework. The web server and dataset are provided by our sponsor. The dataset is not immediately in a mysql-import-ready state, but after some massaging we will import and set that up as well.

After establishing the server, our plan is to add on functionality incrementally, increasing the functionality of the website. Functionality will be added to the codebase in a modular fashion, each with their own branch in our version control system (git).

Once our site is feature complete, we will begin our final steps for deployment. Well have to test cross browser support to see how our site is handled by different clients. We know we want to support browsers like edge, chrome, and firefox. However, if we can manage supporting others that would be convenient. After testing clients, we will polish our documentation and fill any gaps we missed along the way. Documentation will be really important, both for usage and maintenance, because we will be handing this off to people who might not be familiar with our technology.

Finally, our stretch goal will be to develop advanced statistical functions to run on the data. This is something that has not been clarified yet and wont be until the primary objective of the website is finished. The general idea is to derive as much meaning from the data as possible, but in ways that still remain unknown.

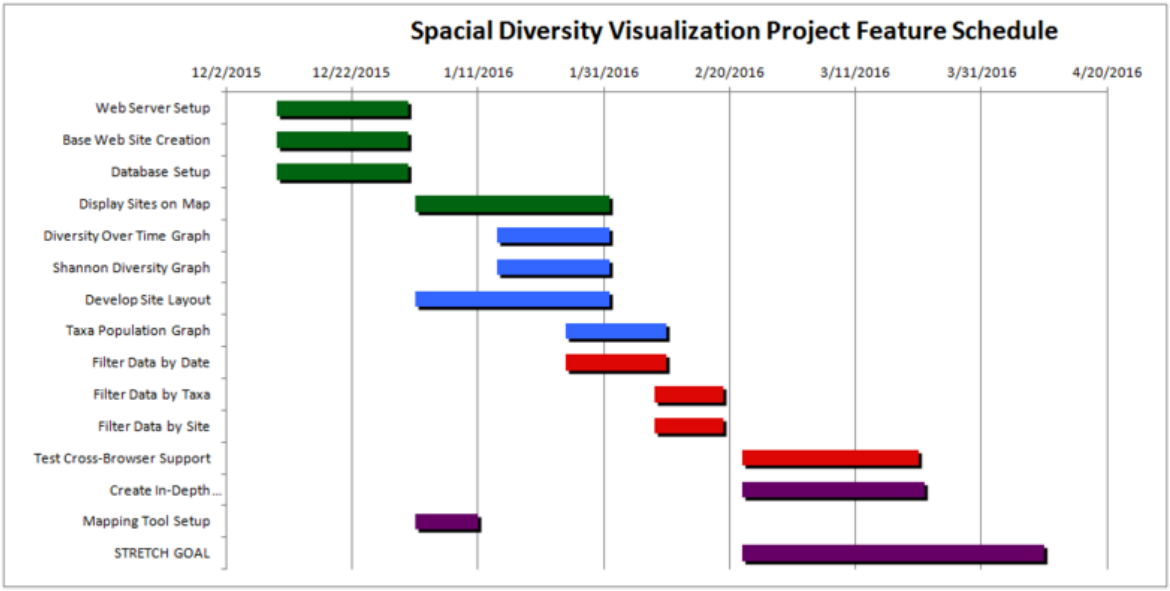

Fig. 3: Gantt chart of schedule

C. Design Plan

1) Database Specification: The biodiversity data was initially recorded on paper, then was transcribed into a Microsoft Access database, and finally exported into an excel spreadsheet and given to us. Needless to say, the data is a bit messy and not in an ideal form for use by our system. We will be creating a database in the model described in the ER diagram in Figure 4. We will need to create a script to map the raw data to our model. The basic idea of the script will be to go through every line and extract all the information into their congruous constructs in the model. This script will also involve some error checking as entering each entry completely by hand is a very error-prone process.

This model takes the simple records of the raw data and puts it in a relational database form. This way information isnt repeated and the records are usable by our systems. In this model each species is broken off into its own table so it isnt repeated for every single sample of that species. This also has the benefit of making it easy to fix a mistake in spelling or change every instance of the species should the name change. We do the same thing with the site, the data for the site is always the same so instead of attaching that data to every sample we simply put it in its own table for any sample in that area to reference.

Each trip to a site is modeled as a sample group that contains the the site that was visited, the date, and all of the various inorganic attributes that were recorded that day such as temperature, pH, and amount of silt in the water. Each individual sample is a part of a group of sample and contains the number of instances that a given species was identified in the overall sample as well what percent it made of that sample.

2) System Specification: The back-end system will be written in a Python framework called Django. This back-end will handle how data is retrieved, manipulated, and sent to the rendered view in the users interface through a standard model-view-controller (MVC) practice. All of this processing will take place on the server side and will respond to requests from the clients browser. The data flow diagram below outlines how the user will interact with the server to manipulate the data displayed.

Upon first visit to the site the interface will fetch all records from the database to display by default. This means that all of the locations on the map where biodiversity data has been collected will be marked, initial statistics will be generated, and the filters will be empty. Once the main view has been rendered the user may begin to explore the map view or interact with the filter options.

Interacting with the filter options should immediately affect rendered data on the page. This means that after a checkbox has been checked/unchecked or a date range has been entered the data should reflect this change. As shown in the data flow diagram above, the users filtration params will be handled by an asynchronous function that will send a request to the server outside of the sites main thread. It is important that this handler is asynchronous,

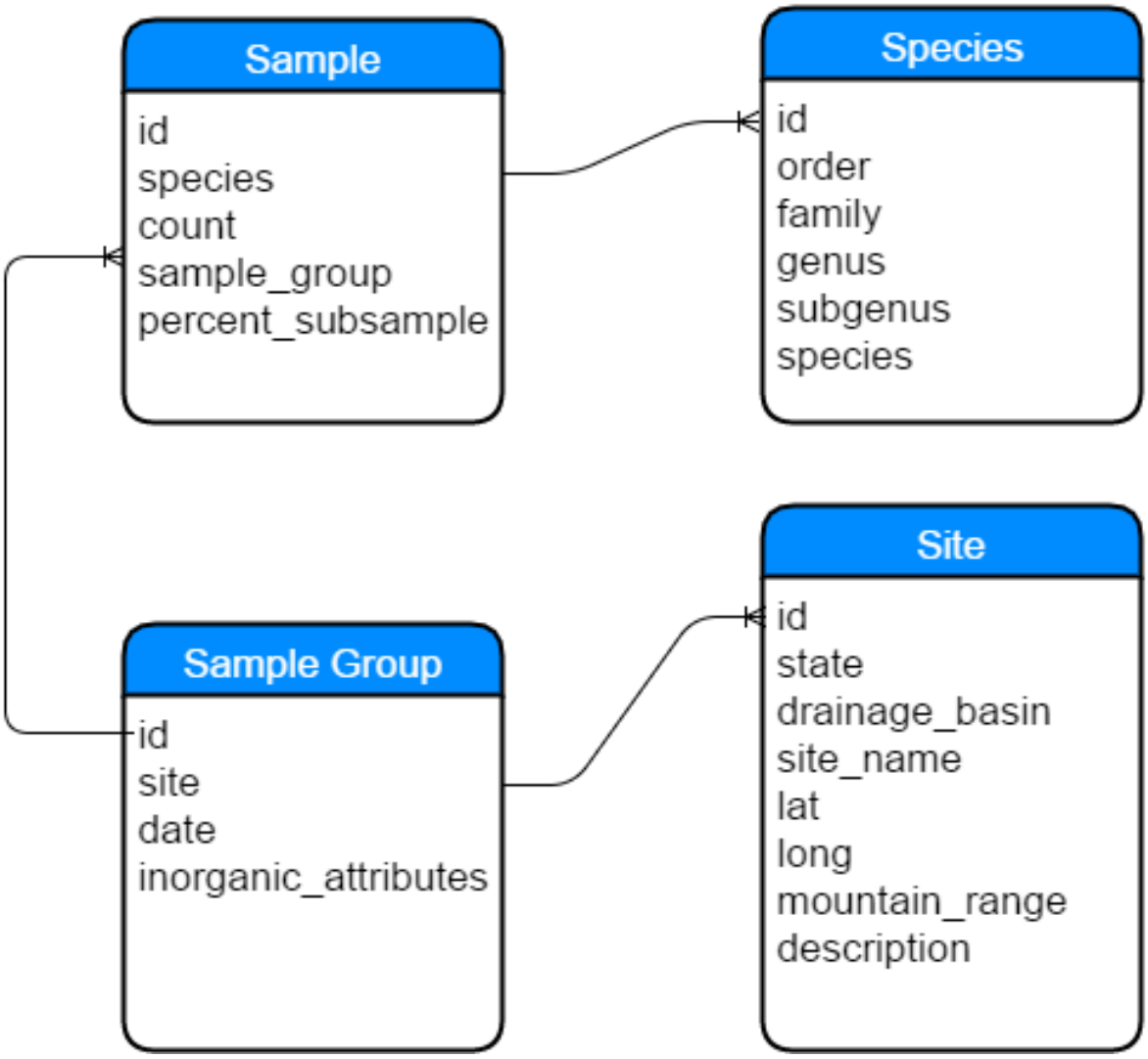

Fig. 4: ER diagram of database

because it means the page will not refresh and upon the return of a filtered dataset from the database the rendered content will be newly generated (i.e., the map shows only those qualifying locations as marked, the statistics are generated from that dataset, etc.).

Controllers: The controller's purpose is to receive specific requests for the application. Routing decides which controller receives which requests. Often, there is more than one route to each controller, and different routes can be served by different actions. Each action's purpose is to collect information to provide it to a view. Whether there will be one primary controller for the sites interface or 3 discrete controllers for each view within the interface is still up to which will be more convenient, since we havent gotten to the implementation phase yet.

Views: A view's purpose is to display this information in a human readable format. The view should just display that information. Our views will be three discrete files that are compiled into one main view for the user using a templating system. The details of how the views will work are part of the interface specification in the next section.

Models: The model is meant to map the objects in the database to code objects manipulatable by the controllers. Each model will have private functions that allow us to access attributes and perform logic depending on the situation.

3) User Interface Specification: The user interface will consist of three views: the Map View, the Filter View, and the Statistics View. These will be tiled on the page as shown below. There will be one full length tile on the left, and two half-length tiles on the right. By default, the Map View will be in the left tile, the Filter View will be

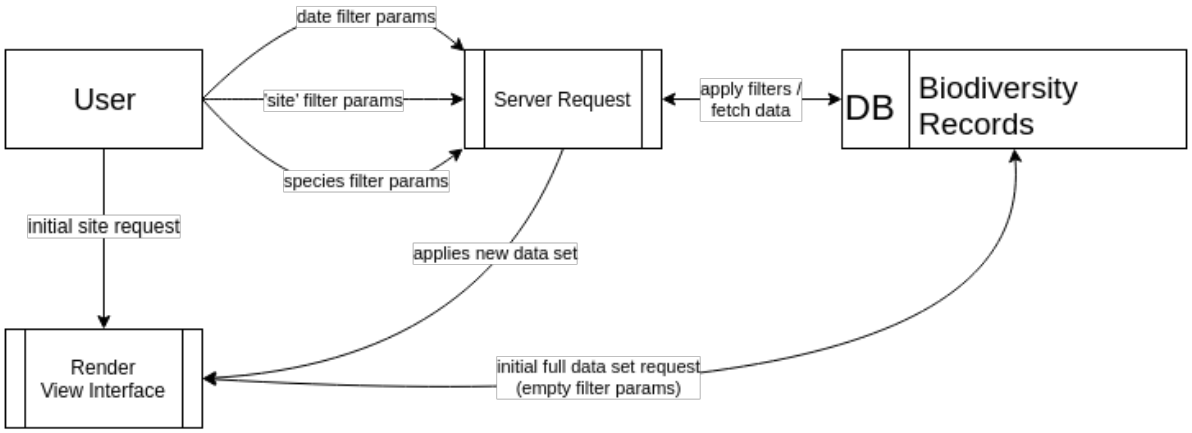

Fig. 5: Gantt chart of schedule

in the top right tile, and the Statistics View will be in the bottom right tile. Each view will have a bar at the top of its tile that identifies the name of the view. Each bar will have its own color.

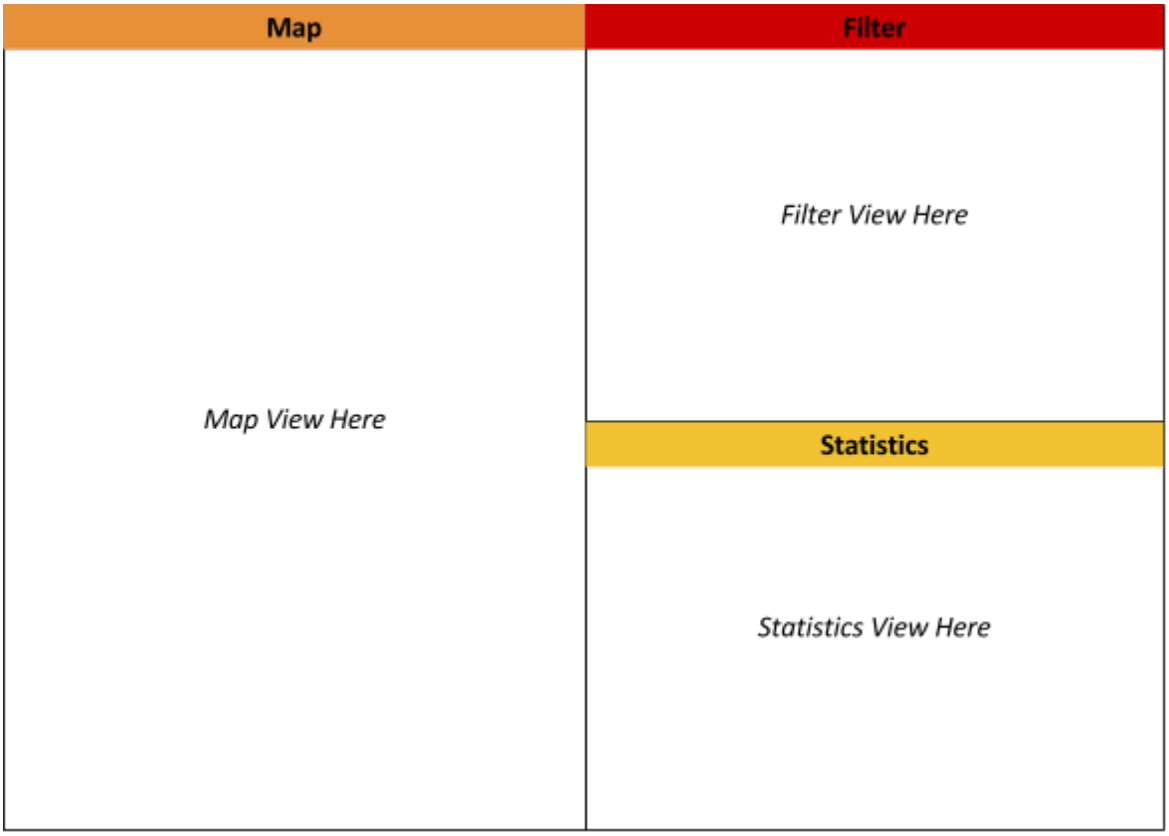

Fig. 6: User Interface Mockup

4) Map View Specification: The Map View will render a map of the area being studied using ArcGis that will be annotated with markers for each site that samples have been collected at. Users will be able to select sites by

clicking on them individual or by dragging a box over the area they are interested in. Once they have selected their sites, the Statistic View will render the current graphs and charts with the new set of data.

Which sites are rendered is dependent on the settings in the Filter View, for instance, if you select a general area only sites in that area will be displayed and if you select a date range only sites that have data for that date range will be displayed. Figure 5 below shows a basic idea of what the map part of the Map View will look like.

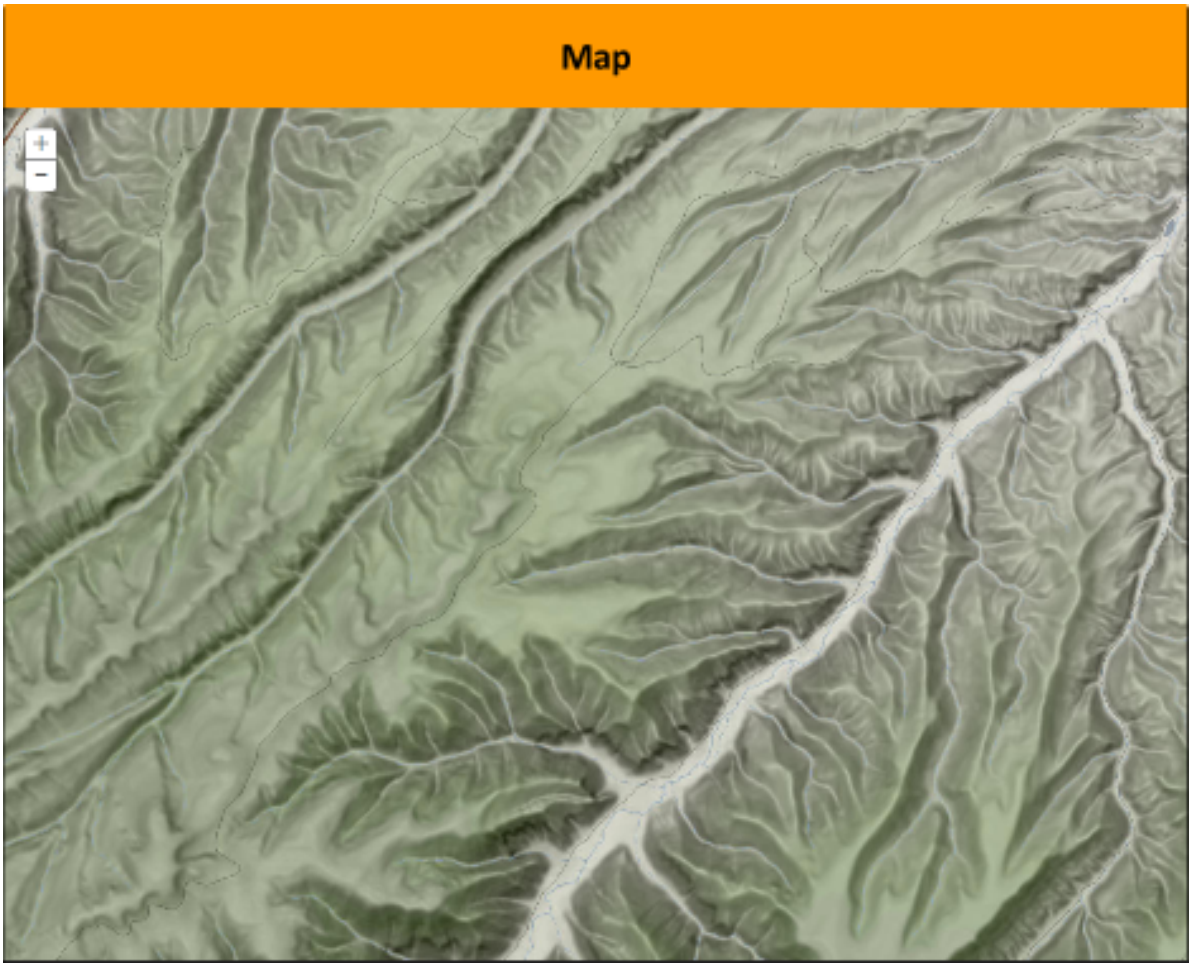

Fig. 7: Map Mockup using ArcGis

Our Map View will look very similar Figure 7, but with colored markers showing the sites. These markers will need to update to show when theyve been selected which we will do by changing the color of selected markers. Almost all of the functionality we need in the Map View will come from the ArcGis API, the markers, the geographical map, and the basic map manipulation features such as zooming and panning.

5) Statistics View Specification: The Statistics View will be generated using the C3.js libraries after being supplied data from our systems controllers. This library was chosen because it has great responsive capabilities, an easy to customize api, and a wide variety of graphs and charts to choose from. As you can see in the mockup below there is a on-hover element in each graph for when a users cursor hovers or clicks a graph point. It will show the exact values for that data point, which is convenient for getting a quick and exact measure. The user can also hide / display data sets by simply clicking their name in the legend for dynamic visualization. The C3.js api allows for complete aesthetic control over the charts along with graphing options for different types of charts.

The statistics view will either scroll upwards or sideways to allow for several meaningful interpretations of the data that we can display. As the user interacts with the filter view the data supplied to the graphs and charts will be re-generated to match the resulting dataset. The exact type of graphs and statistics to be displayed in this view are still being decided by David Lytle and his lab so that will be subject to much change throughout development.

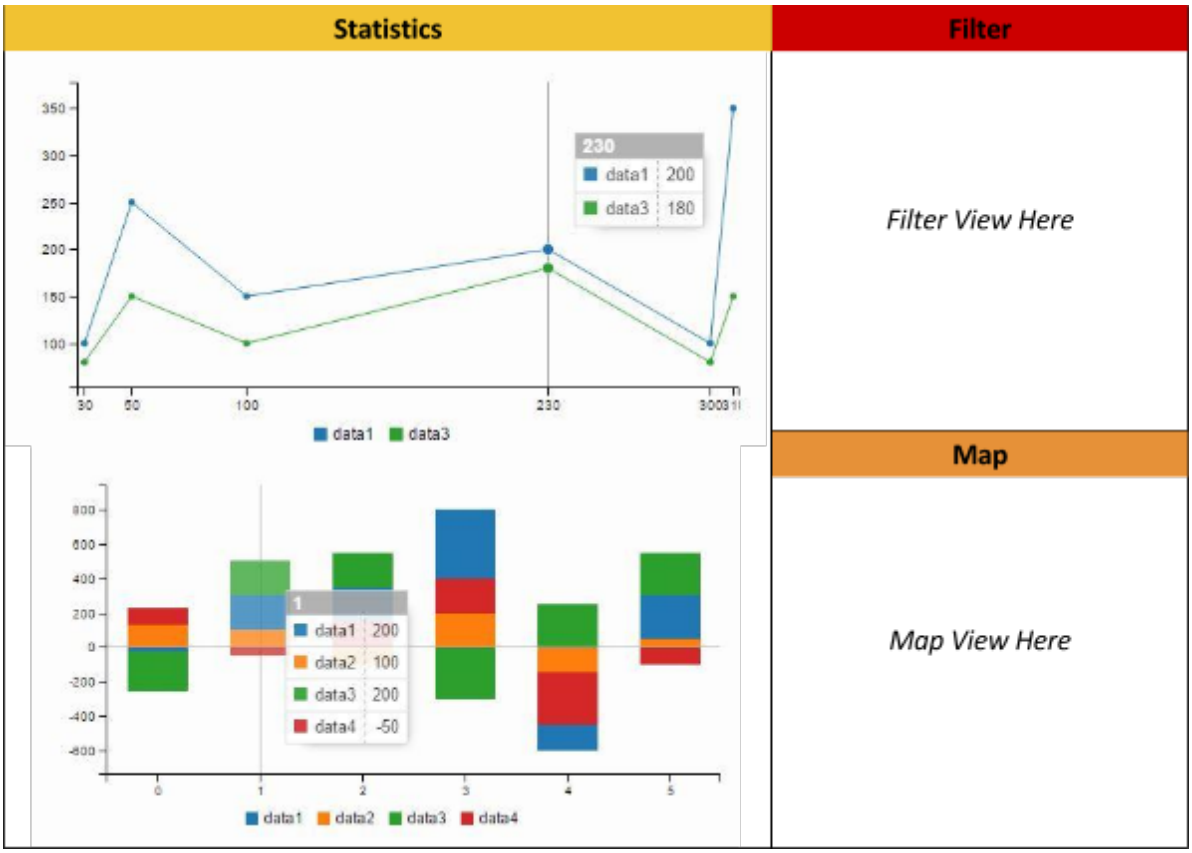

Fig. 8: Statistics Mockup

6) Filter View Specification: The Filter View will allow the user to select information of interest with three filters: the Species Filter, the Site Filter, and the Date Filter.

The Species Filter will allow the user to select biological taxa hierarchically. Each line in the Species Filter will consist of a checkbox, a drilldown, and a name. At the top level will be the phylum Arthropoda. By default, this will be checked and the drilldown will be collapsed, as shown below.

The Site Filter will allow the user to select watersheds hierarchically. Each line in the Site Filter will consist of a checkbox, a drilldown, and a name. At the top level will be the various basins to which sites belong. By default, all basins will be checked and the drilldowns collapsed, as shown below.

The Date Filter will allow the user to select information by date. It will consist of a start date field and an end date field. By default, these will be set to the first and last samples recorded, respectively.

In the Species and Site Filters, clicking on a collapsed drilldown will expand it. This means that underneath that entry, the first level children will appear indented. Clicking on an expanded drilldown will collapse it. Bottom-level children will have no drilldown.

For the Species Filter, the top level will be the phylum Arthropoda. The next level will consist of all the classes of arthropods that appear at least once in the data. The next level will be all the orders that appear at least once in each of the classes, and so on down to the species.

For the Site Filter, the top level will be all the basins represented in the data. The second level will be all the sites at which samples were taken. The Site Filter will only have these two levels.

In the Species and Site Filters, checkboxes will have three states: checked, unchecked, and partially checked. If a taxon is checked, then all of its children are checked. If it is unchecked, then all of its children are unchecked. If it is partially checked, then some of its children are checked and some are unchecked. Clicking on a checkbox will recursively invert its selection. That is, clicking on a checked checkbox will uncheck it, all of its children, all of its childrens children, and so on. Clicking on an unchecked checkbox will check it, its children, its childrens children, and so on. Clicking on a partially selected checkbox will check all of its unchecked children and uncheck

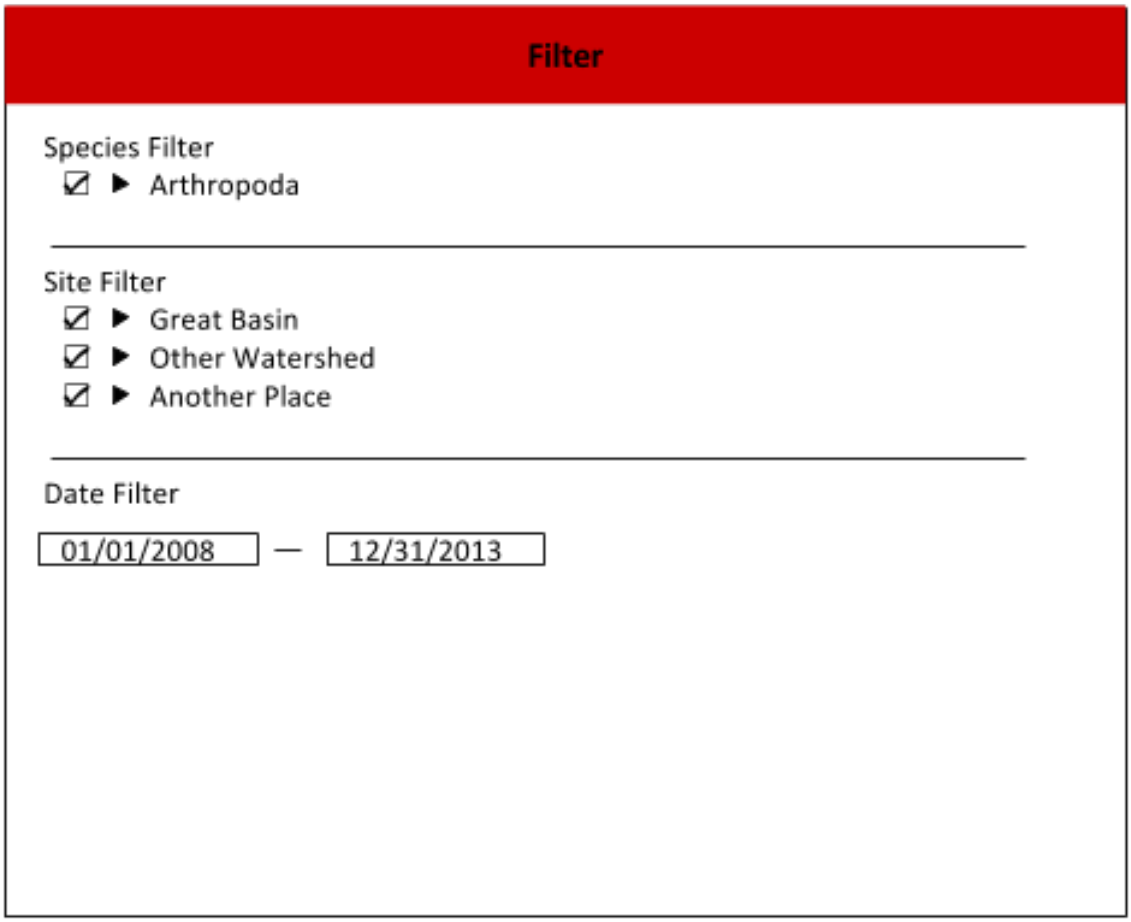

Fig. 9: Filter View Collapsed

all of its checked children, and do the same for its childrens children, and so on.

As an example, the user below clicked on the drilldowns for Arthropoda, Insecta, Hemiptera, Hymentoptera, Apidae, Apis, and Orthoptera in the Species filter and for Another Place in the Site Filter. She then clicked on the checkbox for Orthoptera, unchecked all its children. She then unchecked some of Hemipteras children, then collapsed its drilldown. She also changed the dates in each field of the Date Filter.

D. Changes

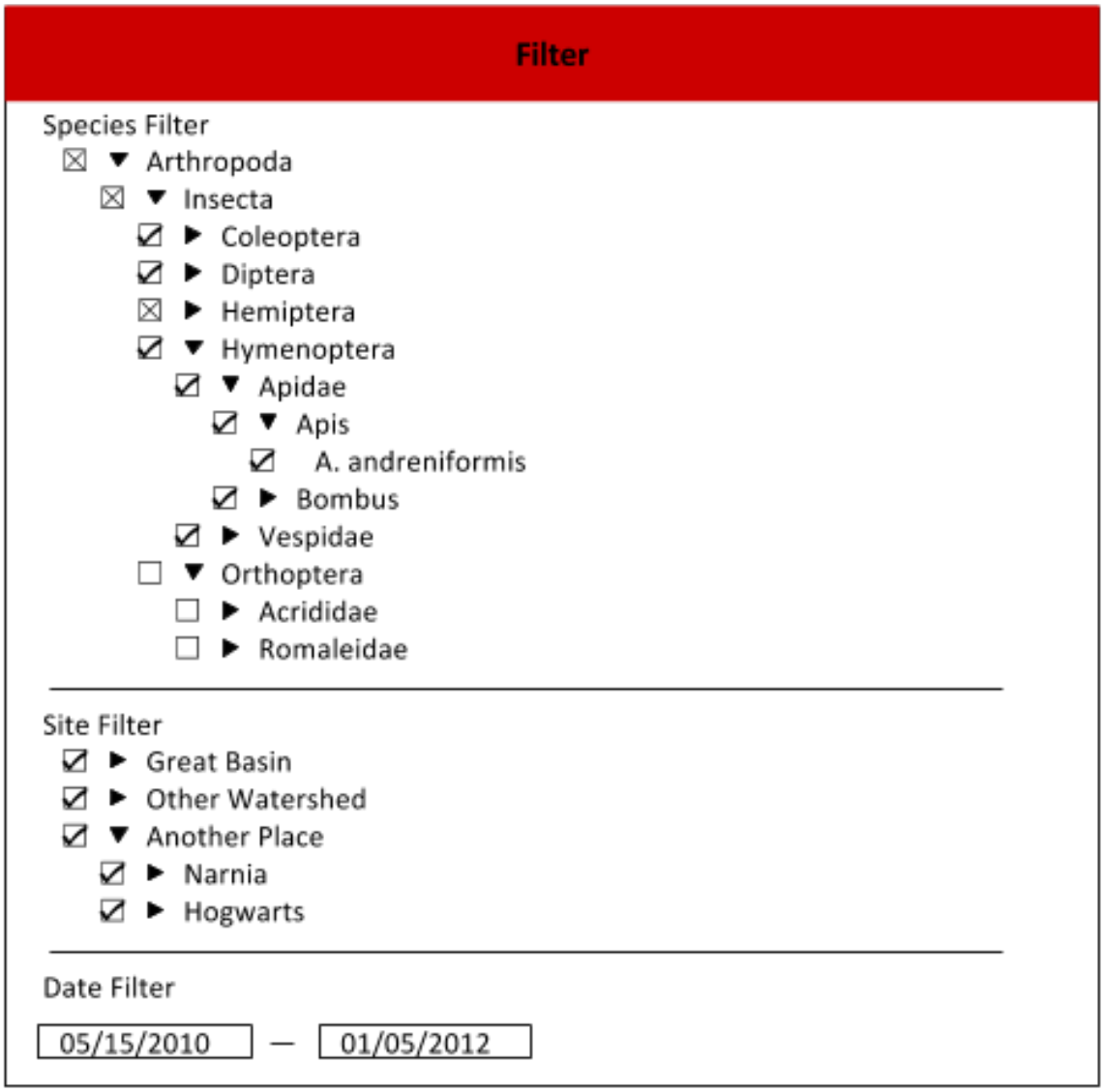

Fig. 10: Filter view expanded

V. Tech Review

Our original Technology Review is included below, along with notes about changes made over the course of the year.

A. Relational Database

1) MySQL: MySQL is an open-source relational database management system (RDBMS). It is the most widely used open-source client-server model RDBMS. The source code is available under the GNU General Public License. The members of our team have the most experience using this database technology and its quite easy to use, which means the initial learning curve will be quite low. MySQL is also compatible with almost every operating system, which makes it quite portable in the event of an environment change.

While the benefits to using MySQL appear to be substantial, members of the tech community have voiced various complaints. Among these are issues with scalability, continuation of development, and limitations. The scalability of MySQL seems to take a hit with increase write/read ratio. This isnt enough to steer large companies and start-ups from using this technology, so we will take that with a grain of salt. The lagging continuation of development

and limitations stems from an issue with its duality of being open-source and owned by Oracle. The code base as stagnated slightly and as a result MySQL hasnt grown much for some time. This isnt an immediate issue for our team, however, because the features it does have are exactly what we need. We will still compare this tool against the other potential database technologies to make sure we make the right choice.

2) Postgres: Postgres is also an open-source database technology and is an object-relational database management system (ORDBMS). Unlike MySQL, the source code is available only under the PostgreSQL License. No members of our team have used this technology yet, but from skimming the documentation it does not look too foreign and could most likely be learned while developing the site. Postgres operations have been said to be faster than that of MySQL.

While Postgres can be faster at times and scale better than MySQL when using it concurrently, it still has its disadvantages. When executing many read-heavy operations, Postgres can be overkill and actually less performant than MySQL. Postgres has a strong community behind it, but its not as large as MySQL and is far less popular.

3) MongoDB: MongoDB is also an open-source database, but rather than being a variation of a RDBMS database its a document-oriented database. Document-oriented databases are also known as NoSQL databases, which casts aside the traditional relational-database structures supported by MySQL and Postgres and insteads supports JSON-esque documents (called BSON). MongoDB is available under the GNU Affero General Public License and the Apache License. If implemented correctly using the document format, MongoDB can be much faster than its RDBMS counterparts. The most advertised benefit to using a NoSQL database like MongoDB is the fact that it is very flexible, due to the fact any entity can have any attributes.

Being flexible is a very nice feature for a database technology, especially when youre unsure of the data types coming in. However, we have a fairly static data pool already and we know exactly where things will be and the structure probably will never change. This means that MongoDB might be fun to consider if the performance boost outweighs the benefits of the other database technologies, but its probably not the right tool for the job.

4) MariaDB: MariaDB is a fork off of the MySQL database. The fork off of the major code repository was in response to the Oracle acquisition. MariaDB was designed for drop-in compatibility when replacing MySQL in existing systems. It is almost identical to MySQL except for a few minor differences. The most prominent of these is the fact that its got a larger open-source community behind it. Secondly, it requires a bit more memory due to its default enabling of the Aria storage engine that handles internal temporary tables. Outside of those, the differences are very technical and minor.

5) Selection: After careful consideration, weve chosen MySQL as our database technology for this project. Its offers the same level of verbosity as the rest of the tools, while being supported in an open-source and proprietary context. The relational structure of MySQL caters to our data set and will be easy to manipulate and query. MySQLs portability across all major operating systems is very enticing and will prove to be very helpful in the event that we have to change hosting services or move it to a remote server. Finally, our teams base familiarity with the tool was a convenient plus and will ideally remove whatever potential overhead for learning a database tool during our development.

B. Backend Web Framework

1) Django: Django is a free and open-source web framework written in Python. Its major design philosophy is that you should be able to focus on developing your application without needing to focus on the intimate details. It seeks to create this environment by offering a clean and easy to use framework with various components encapsulate the different functionalities that you need to run a web application. It comes out of the box with various components that make it simple to start working on your application right away such as: a standalone web server for development, an interface for administration, and built-in protection from various kinds of web attacks. These are only part of the default modules, there are many third-party packages that can be plugged into Django that add even more powerful features such as site-wide search or content management. Overall, the framework seems to be about taking your mind of the web side of things so that you can focus on making your application.

2) Symfony: Symfony is a free PHP web application framework. It is focused on creating a platform that can be set up easily for a small application but is robust enough to support large and complex applications as well. It also seeks to give developers full control over the configuration and customization of their project. Another principle it operates on is DRY, Dont Repeat Yourself, and it implements this by the way of bundles. Everything in Symfony is wrapped in a bundle that can be reused in other applications. For instance if you make everything you need to interact with a certain database you can keep that in its own bundle that you could use in two different applications that needed that database. Even the core of symfony is essentially a collection of bundles that you can mix, match,

and modify as you see fit. There is also a large and active community of developers creating and collaborating on open source bundles. These can greatly reduce development time on a project because you often dont have to create generic functionality from scratch.

3) Laravel: Laravel is a free and open source PHP web application framework. Similar to Symfony is it based on a modular package based system. In fact, Laravel has a very similar approach to Symfony, it uses many of the same tools and technologies and even uses parts of symfony itself. Laravel differentiates itself from Symfony in two major ways: being very simple and easy to setup and the focus on elegant and clear syntax. Even the Symfony website talks about how Laravel takes the pain out of common tasks associated with web development. Laravel is in my mind the PHP equivalent of Django in that it seeks to create an elegant and simple way to create web application without having to deal with the dirty details.

4) Selection: We have decided to use Django for our project. Even though our group members have experience using the other two frameworks, we have gone with Django for several compelling reasons. The operating system was a significant factor. Python is a very clean and elegant language that had thought and foresight put into its design. PHP, on the other hand, has somewhat of a reputation as a language that it very ugly, messy, and unguided. Cleanliness is not the other reason for preferring Python, though. Many of our mathematical and statistical modules will be in Python so it will be preferable to work in the same language as much as we can. Another factor is the ease of starting a project with Django. It is simply a handful of commands and you have your fresh project ready with a development webserver all set up. With Symfony setup can be a long arduous process as you set up all the individual pieces and try to understand how to get them all to work together properly. Symfony definitely offers more customization and power, but our application is small project that wont need to take advantage of that power. In the end it was closest between Django and Laravel but Django wins due to being Python instead of PHP.

C. Numerical Analysis

1) MATLAB: MATLAB deserves to be mentioned here, as it is the most commonly used scientific computing language. It provides a tremendous amount of functionality for scientific computing needs. If the numerical analysis portion of our project were completely standalone, MATLAB would be an excellent choice. However, we cant run it on a backend server, especially one that talks to our web framework.

2) Numpy/Scipy: Numerical Python, or NumPy, combined with Scientific Python, or SciPy, is the authoritative set of Python packages for scientific computing. NumPy is organized around the ndarray object, which is essentially a more Pythonic version of an array. They are distinct from Python lists in that they do not hold arbitrary objects (unless you explicitly ask them to, in which case you are probably using them wrong). While Python lists make no assumptions about the homogeneity of the data they are storing, NumPy ndarrays optimize based on the assumptions that each entry contains the same fields of the same types, and the entries are organized into a square n-dimensional matrix. These simplifications turn out to cover the vast majority of scientific applications, and allow NumPy ndarray objects to use C structures under the hood. This makes them efficient in memory usage and fast in processing compared to plain Python objects. On top of the ndarray object, the NumPy and SciPy modules provide functions for many common mathematical, statistical, and scientific routines. The libraries cover everything from element-wise arithmetic to linear algebra in single function calls, such that users will rarely have to explicitly iterate through an ndarray object element by element. More advanced specific capabilities can be offered by modules such as OpenCV, but many of these make use of NumPy ndarrays. Finally, NumPy and SciPy have excellent documentation and support. In short, NumPy and SciPy are Pythons answer to MATLAB. They should be more than adequate for our project.

3) SciRuby: SciRuby happened because some people went, Hey, wheres the scientific computing for Ruby? and started a GitHub repo. The project is still young, but provides an impressive smattering of scientific computing tools. In addition to providing most of the same basic array manipulation functionality as NumPy and MATLAB, SciRuby has built in functionality for some advanced routines, including several machine learning methods. It even includes some visualization tools, which would be of use to us if it were on the frontend. However, SciRuby has scattered, inconsistent documentation and support. Many of the modules it offers are not currently stable. One cannot simply assume that a SciRuby package just works.

4) PEAR: PEAR, or the PHP Extension and Application Repository, is a collection of PHP snippets that developers have built and which seem useful. It includes everything from user authentication to some scientific computing routines. Each extension must be found and installed separately, and there is not necessarily any consistency between extensions. It is about all PHP has to offer for numerical analysis.

5) Selection: We have chosen to use NumPy/SciPy for the numerical analysis piece of our project. This option should provide us with the greatest ease of development for several reasons. First, NumPy and SciPy are consistent. They are well-documented, well-tested, and well-supported. In general, if think NumPy should be able to do it, it already does, and one can guess what the function is called and how to use it. If not, it can reliably be found in the documentation. The only other technology which provides this ease of development is MATLAB, which is not suitable for a backend server. NumPy and SciPy also enjoy the advantage of being able to talk easily with our website backend, which will use Django.

D. Mapping API

1) Google Maps: The Google Maps API provides all the functionality of the Google Maps app as an API that is (pretty much) free to developers. This means that developers can, for example, easily and simply create a map, scale and transform it, place markers on it, and make those markers display further information. Common use cases such as these are simple to implement. Furthermore, they are simple and easy for users to understand. Customization, for both developers and users, is a greater challenge. The Google Maps API follows Googles principles in general: for 99% of use cases, it is simple, easy, and elegant, and for the other 1%, it is useless.

2) ArcGIS: ArcGIS is a powerful mapping tool commonly used for geospatial applications and research. It provides a JavaScript API for developers. The API is well-documented and includes understandable code samples. Advanced tools for visualization and analysis are built in, as well as nice tools for information visualization in separate windows or panels. It even includes a tool for visualizing watersheds, which could be of direct use to our application, and which would be very challenging to implement ourselves. It does require a license, but that license is available free to Oregon State faculty and students.

3) CartoDB: CartoDB is a powerful map-based visualization tool, heavily centered around its JavaScript API. The API is not as well-documented as that of ArcGIS, but it does include useful sample code. It provides tools for viewing maps and visualizing and analyzing data. It is designed for big datasets. CartoDB provides similar functionality base functionality to ArcGIS, but with less advanced functionality. The free plan is reasonable, but premium plans are expensive, and do not come with Oregon State tuition.

4) Selection: We have decided to use ArcGIS for our mapping API. It implements many of the features we already know we want, which sets it apart from the Google Maps API. Additionally, we will already have the license to use it at no financial cost, which sets it apart from the CartoDB API. Furthermore, ArcGIS will be familiar to researchers in the field, which is important for our project. Finally, the API is well-documented and easy to navigate, which will improve ease of development.

E. Graphing Utility

1) D3: D3 allows you to bind arbitrary data to a Document Object Model (DOM), and then apply data-driven transformations to the document. D3 provides powerful data-driven visualization. This means that we can perform minor statistical manipulation to the data pulled from the database in real-time via a view on our website. This is incredibly useful, because users on the site will be able to make selections on what data theyre interested in and the analysis should change with those selections. D3 uses HTML, SVG, and CSS. This technology is able to be implemented with any web framework and only requires either a reference to a remote CDN or a local install of the source files. D3 can also work with any data size, even performing while in the Gigabits range. The main issue with D3 is its not al charting tool. This means that while youre able to perform data manipulation and graph out a trend, you wont be able to provide quantitative metrics like bar charts and pie charts. This could be solved, however, by co-implementing it with C3 (a similar, but charting-focused tool).

2) C3: C3 is based on the D3 codebase. whereas D3 is primarily a plotting tool, C3 is a charting tool. Developers tend to use one or the other and then compensating for their differences. C3 proves to be very customizable with the ability to provide custom classes to all chart types. Where D3 shines in power C3 provides ease and is much lighter. Users interacting with a C3 chart can easily show/hide series, update data, select data points, and focus on series. This is unprecedented in other charting libraries and makes for a very smooth user experience. The built-in charts are very aesthetically pleasing and have extensive properties available for customizing their format. D3 has been out for several years now, but C3 has only just surfaced. Consequently, C3 is fairly lacking in documentation. One or two of our team members are already semi-familiar with this tool and can implement basic graphs, but there will be a learning curve during development. Since C3 is based off of D3, D3 is much more powerful with the amount of functionality it offers. Even though we are comparing these two technologies for use over the other, there is a possibility we could try to implement both if the situation calls for it.

3) Google Charts: Google Charts is an API created by Google for creating charts in HTML and SVG through javascript. It generates a static PNG image of the chart after data processing and embeds it in the web view. This api is very verbose and supports many different kinds of charts- including unconventional ones like bubble charts,diff charts, gantt charts, geo charts, sankey diagrams, and tree map charts. The DataTable class that populates the charts can be populated from a web page, database, or any data provider supporting the Chart Tools Datasource Protocol (SQL-like language implemented by Google sheets, Google Fusion, and even Salesforce.) Google charts offers more functionality than C3 and is simpler to learn/implement than D3.

Despite being easier to use than D3, it actually is less customizable. D3 provides much more in the ways of control for generating plots and charts. Google charts also do not support as much data as C3 and D3 can handle. With this technology, exporting to a png file is very easy, but the graph view on the web page is hardly as dynamic and interactable as C3 or D3. This is a major problem when catering to a user group that needs to do real time sorting and filtering on data sets, like ours.

4) Selection: We selected C3 for this tool, because its the most aesthetically pleasing of the charting tools. It has all the graphs that we are interested in displaying for our primary objective and is easy to implement. While D3 and Google Charts are more powerful, they are also more time consuming for the same result. C3 provides quick access to the exact features we need, all while being simple in usage. Its important to us that we are able to achieve the desired look while providing meaningful metrics along with some level of user-interaction and this technology satisfies all those conditions the most completely.

F. Frontend Web Framework

1) Bootstrap: Bootstrap is a free and open-source front-end framework for creating web applications. It is basically a collection of a bunch of different tools and assets to quickly and simply create the applications interface. The assets it contains are things like forms, button, navbars, and similar css that gives your website a clean and professional look. It also contains a bunch of javascript components to handle things such as dropdowns, tabs and modals. Basically, these breathe life into your application making it more dynamic and interactive. Another excellent part of Bootstrap is its grid system that makes creating the layout of the website very clean and intuitive. Bootstrap is an industry standard for a reason, it makes creating a professional looking website as easy as including some dependencies.

2) Materialize: Materialize is a CSS framework that seeks to implement Googles Material Design. Material Design is a design language that seeks to create a unified user experience using the metaphor of material, bold graphics, and meaningful motion. Basically, Material Design links the paper and ink roots to our current technology and design in order to create a striking and clean interface. On terms of functionality, Materialize offer similar functionality to Bootstrap but following Material Design guidelines that Google has laid out. Due to this is has slightly less functionality that Bootstrap but it has the same core functionality.

3) React: React is an open-source Javascript library used to create user interfaces. It specifically focuses on tackling single-page applications. Its development is led by Facebook as well as the community that uses the library. Its strength is in large applications where the data changes frequently. It makes sense then that companies such as Facebook and Instagram have a hand in its development. React is built on the idea of simplicity and reusability, you tell it how you want something to look and it updates your application as the data changes. Everything built with React is a component. This means that by design everything produced in React is separated, reusable, and testable. While the other frameworks are built around the idea of various kinds of websites with branching paths and many pages, React does one thing but does it well: offer a clean and intuitive way to display that changes in a complex application on a single page.

4) Selection: We have selected Bootstrap for our project. After researching React it was clear that it wasnt what we needed for our project. The data in our project is not changing. It is also a small project so it likely wont benefit from all the features React is focused on. So the main options we were looking at were Bootstrap and Materialize. Functionality-wise these are very similar to each other. They both offer a grid system, css elements, and javascript components. The deciding factor is what kind of design we want for project. Material Design is focused on consistent user experience across different devices which isnt really a concern for our project. This isnt a commercial app that needs a fancy or elegant user interface, we need an interface that is clean and displays our applications information in a manner that is easy to read and use. For this reason Bootstrap is a good choice. It is standard and offers a simple interface that will meet our needs and present a good stage for our data.

G. Changes

The decisions we made in our technology review have remained fairly constant. There are only two significant changes. First, we did not end up needing to use NumPy/SciPy, as our numerical analysis needs were less than we initially imagined. Second, we additionally used a Javascript library called FancyTree for the hierarchies in the Filter View.

VI. Weekly Blog Posts

A. Fall

1) Week 3 Update: Accomplished last week:

Met each other, we're all really cool dudes. We then met with our client, David Lytle, and went over high-level project details to gain better understanding. Created a rough draft of our project abstract.

Working on this Week:

Setting up touchpoint site and blog (yay). Finish final draft of abstract along with problem statement, solutions, etc.

Goals for next week:

Meet with client for more discussion and updates. Work on whatever is assigned that week.

2) Week 4 Update: This week, we did not make progress on our project. We reached out to schedule a meeting with our sponsor for next week so that we can put together requirements. Our goal for next week is to have our requirements document finished.

3) Week 5 Update: This week we met with our TA to introduce ourselves and touch base. We also met with our client Dave Lytle and talked about the requirements for the project. This meeting gave us a more complete idea of what we were going to create and we then wrote up our requirements document.

4) Week 6 Update: This week we began revising our requirements document before receiving feedback from our TA in hopes to get a head-start on what sections we think he'll want us to supplement or rewrite. We also had a team meeting for divvying up the portions on the technical review document. We discussed the technologies we are reviewing for each solution so we could work on our sections on our own time. Next week we will have it finished in time for turn in and start looking at the poster assignment.

5) Week 7 Update: This week we did our rewrite on the requirement document, which mostly involved making our sharepoint gantt chart fit in a document. We also wrote our Technical Review. At this point we have a pretty good idea of what technologies we are going to use for each piece of our project. Next week we will work on our poster and our elevator pitch.

6) Week 8 Update: This week, we killed it. After we finished our technical review last week, we created the first draft of our poster. It is very sparse at this point, since we haven't done any implementation. However, it will be useful to get feedback on the poster now so that we can improve it for next term. We also got an early start on our elevator pitch, which is going to be great.

7) Week 9 Update: After perfecting our elevator pitch to deliver in class we switched our sights to the design document project. While we are still in the planning and structure phase of it, we will be begin writing the rough draft Monday next week. With the holiday this week we won't be able to accomplish a whole lot, because we're each traveling a bit to visit family and won't be around. Week 10 will be a major progress week for us.

8) Week 10 Update: Last week we began assigning portions and planning our design document. We had already figured out our elevator speech and delivered that on Tuesday. We had a good meeting with our TA Jon Dodge and made sure we were all on the same page for ending the term. Our design document is now complete and ready to turn in (via hard-copy and email) and we are starting to plan / divvy up the progress report to turn in during finals week.

It's been a good term. Our teamwork is strong, the project's goals are clear, and we're ready to dive head-first into some development.

B. Winter

1) Week 1 Update: We neglected our project over break, so we spent this week doing a bit of catching up. We worked on installing and configuring everything we need to get up and running.

We will need a web server, so we contacted our client about hosting one. In the meantime, we will each host the website on our local machines. The code will be kept in sync on GitHub.

We also worked on learning how to set up and use Django. We have a Django project and app set up, and this is where we will do our backend development.

We had a bit of diffculty finding a good time for all of us to meet. We want to spend a solid block of several hours working together each week. This was tricky, but we have settled on Monday evenings at 6:30. We worked together well, and spent our time productively.

For next week, we aim to have our web layout finished so there is something to look at. We will also start working on manipulating the data into a relational database.

2) Week 2 Update: This week, we created our basic web layout. We have a website, and it has a header and sections for each of the three main sections. It uses bootstrap nicely, so that browser window size doesn't make our layout ugly. The whole thing looks pretty slick. Actually, it looks ugly as sin and has no content, but it's still impressive for a first start.

In the underlying code, each section (Map View, Statistics View, Filter View) is a separate view in a separate file. We've assigned each team member to bring one view to alpha level functionality. This will mean each view pulling in static data from the database. Beta level will be allowing the filter view to actually control what is displayed in the map and graph views.

We worked well together this week. We are meeting for a few hours a week to go over group stuff, and now have separate tasks to work on in the meantime. We should be meeting with our TA later this week. We still need to hear back from our client to get the cleaned up data, a hosting site, and an ArcGIS account though.

Next week, we will make a script to import the raw data into our database. This will be a big part of our alpha release. Once we have data, we can actually display it and make it look pretty. Hopefully, we can also start including that data in our views by next week as well.

3) Week 3 Update - Jasper: My task for this week was to finish alpha level functionality for the Filter View. I finished this on time, but had to meet with Ty to figure out how to merge it into master. I am yet a Git padawan. Our meeting was more training on Degobah than productive work. Stil, we got it merged, and now the filter view looks pretty.

The goal for this coming week is to start a script for importing the database from a CSV file. We will work on this as a team, and we will need to meet with our sponsor first to clarify how the Database is to be set up.

4) Week 4 Update - Jasper: This week was a little slow. I meant to make enhancements to the Filter View this week, but did not get around to it. I did meet with the team to make our plan of attack for the midterm report. We created a skeleton of the report. We also started the database import process. Our goal for next week is to have the website ready for alpha release. This way, we can spend the following week focusing on writing it up and doing the video.

5) Week 3 Update - Ty: This week the plan was to work with Jasper in settling merge conflicts from his branch containing the relevant "filter-view" code, developing the script to import the bug data to MySQL, and creating the models to map to the database. Last week I was able to set up the webroot to be organized the way we want w/ all of our dependencies working and I developed the graph view with side scrolling capability and filler data. Jasper and I have since settled the merge conflicts and merged his branch into master. All that remains is Alec's code for the map view and we're near alpha-level release quality. We decided to reach out to David about the data he supplied before importing it further, because we noticed some weird fields that may or may not be relevant and wanted to meet with him to figure out exactly what we need. Next week we will email him and/or meet w/ him if we can and clean up the main view whether or not we have the map view.

6) Week 4 Update - Ty: This week we were trying to get a better idea of how to handle the data and get all of our components into the repository. I worked with Alec to get his database configured and his map view set up and he was also able to start thinking about ways to deploy our project. We also met with David and his lab and went over more ideas and details on how to continue with the data. We figured out some graphs that we might try to implement soon. Basic ones for alpha release, but more complex ones for beta and after. Next week I want to work on making some improvements to the main interface and start working on the write up and video, since almost all the base components are in for the alpha. We need to get the data in the database and render that data, but that shouldn't be too bad.

7) Week 5 Update - Jasper: This was a productive week. Our team finished the database import script, so we are now ready to start supplying real data to the website. I made the necessary additions and enhancements to the Filter View to get it ready for the alpha release. These turned out to be relatively straightforward. I also drafted my parts of the progress report. I may need to add more content later. Next week, our team will meet to finish the progress report and make the video. I'm pleased with our progress.

8) Week 5 Update - Ty: This week was very good for us. All three of us met several times and created issues on github for tracking and setting goals for alpha and beta. We established who needs to do what over the coming weekend to polish our alpha and to get crackin' on our report. Alec and I met on Wednesday and managed to write an import script which parsed our clients excel spreadsheets and placed the data into our database correctly. Now that all of us have working databases, providing info to our views shouldn't be a problem at all. This next week we are all about working on the report, creating the video, and wrapping up our alpha. We've divvied up the progress report so that everyone knows what they need to type up and each of us have one development task for alpha release. Mine will be getting some better placeholder graphs, maybe even with some data in them.

9) Week 6 Update - Jasper: This week, I did my parts for the alpha release. We got together as a team on Monday and recorded the video. I gave an introduction to the project and talked about my contribution to the Filter View. I also completed my writing sections for the report. I wrote about the progress and work needed for the Filter View, as well as an intro and an overview of the graphs we will include for our beta release. This was all relatively straightforward, since I had finished the necessary code last week. Next week, we are working on getting data from the database to actually display on the database. My goal is to populate the Filter View with the actual data.

10) Week 6 Update - Ty: This week we were able to push our project into alpha and wrapped up those minor tasks. We spent our meetings this week focused on writing our portions of the design document and filming our video. We were able to shoot our video in one go after we each wrote an outline for ourselves on what we would say. Our document was already split into portions from a previous meeting, so all we had to do was go back and fill out our sections. Alec was kind enough to turn in the final product (document and video). Next week we plan on working on getting our data dynamically into our views for all of our templates. once the data has been initially rendered, we'll be able to finally start doing queries between views and hopefully stroll into beta.

11) Week 7 Update - Ty: This week we are working on getting data into our templates dynamically from the database. Up until now it's been statically supplied. We have all three views embedded and it's coming along nicely. Next week we will focus on wrapping up all of our branches and getting them merged into master. Then after that we will be tweaking the design in preparation for showing it to our customer next week.

12) Week 8 Update - Ty: This week we showed our application to our customer and his team and it was very well received. They liked it a lot in it's current state and it was really cool to get that kind of feedback. Being able to see the product was really helpful for all of us to talk about goals moving forward and visualizing requests. Afterwards we went to the library and made some changes we talked about in the meeting. we added all of our beta features to the github issue tracker in order to better keep ourselves in order. We're gonna try to work on the site simultaneously with our report/video rework.

13) Week 9 Update - Ty: This week I was able to get the filtered data to create a fancy new chart based on what was selected in that view. Thanks to Jasper, who got the filter object created after a successful post request, I was able to then piggyback on that process and set my graph labels and x-axis values to whatever is necessary to zoom in on that dataset. I was really worried that this'd be the hardest thing to do, but now that we have one done we are going to totally be able to crush this and have beta looking good. For the rest of this week I am going to be focusing on learning how to use LaTex and start getting some content into that paper. I think next week our group is going to continue getting work done on the project along with cleaning up our poster and creating the video.

C. Spring

1) Week 2 Update - Jasper: This week, our team met to assign jobs for the 1.0 release. Most of our tasks involved fixing bugs. In fact, we powered through a bunch of bugs at our meeting. I fixed the NULL and repeat entries that were showing up in the filter view, which makes it look much cleaner. In fact, our website overall is looking much cleaner. Hopefully, we'll be able to fix our last bugs soon and proceed to our stretch goals. Exciting!

2) Week 2 Update - Ty: This week we got together and ironed out the rest of what was to be done for the 1.0 release. We found about 8 minor issues with our beta release that could be smoothed over that covered small code refactors, more specific database queries, and some data massaging. We found an interesting issue with how to go about handling NULL entries in the database, because while they're un-labeled, they were still a reading at some taxanomic level and we're not sure whether to count it once total, once each, or not at all. This is definitely something to bring up to David or his team very soon to get ironed out.

3) Week 3 Update - Ty: This week we were able to meet with Richard, an associate of our primary client, and go over some of the data decisions covered in our posts from last week. Without going into too much detail, he said that we were probably safe to disregard any NULL datapoints on the levels at which they occur. This means tracking the taxa as long as it's identified in the graphs, but as soon as the level we're looking into has any undefined sub-levels we will ignore them from the plots. Outside of the primary data issue, we demo'd our project for him and he was very happy with it. During the demo he showed us how they'd probably spend their time using it, so we found a couple more minor bugs that we'll address soon. Other than that we will be wrapping up our poster and focusing on our documentation soon!

4) Week 4 Update - Ty: This week we made great progress towards being finished with our project. Firstly, we were able to close all but 1 or 2 minor issues on github. The remaining issues are further testing/code clean-up along with a bit more visual functionality that Alec is gonna take care of regarding data points on the map. Next

we were able to get the cosine labs to open our server to IPs off-campus. Finally, we got our poster ironed out with a couple iterations of review from Jon. Next week we are going to be wrapping up our midterm report and video on Wednesday. We reached out for a meeting with Dave Lytle and he said he could meet in a week or so, since he's been out travelling. We're excited to show him how far we've come!

5) Week 5 Update - Jasper: This week, we got everything ready for the upcoming deadlines. I am finished with all of my assigned issues on GitHub. I finished my sections of the report, which had only minor edits to make. I also made the necessary changes to my sections of the video, which were also minor.

I forgot my blog posts during the last several weeks, during which time I was mainly working on my issues on GitHub. We also met with our sponsor's grad student during that time to show him the current state of our product. He was really pleased with it.

Next week, I'll probably be focused on my thesis, but if I have some spare time, I might get a start on the final documentation, if the assignment for that is up by then.

6) Week 5 Update - Ty: This week was primarily focused around our documentation and presentation goals. We revisted our final report from last term and cut out sections that need updating and those that we can re-use. Our video is finished at the moment, leaving only our document. I still have one issue on Github for labeling the graphs, but I should be able to get that done soon. Next week we will be meeting with our client and reconnecting after a long time due to his business trip.

7) Week 6 Update - Ty: This week we are fully finished with our project. I was able to close out the last issue and Alec was able to set a meeting with our client. We spent a majority of our time wrapping up the production report (mid-spring term report) and video. That's all submitted and now we're just preparing for expo and documenting the code. This week we plan to meet with our client to touch base and show him the full product. We've sent him all of our correspondence and a link to the live site, but it'll still be good to have a face to face meeting. We're now in full expo preparation mode.

8) Week 7 Update - Ty: This week we finally closed all of our issues on github and met with our client for a pre-expo meeting. Now that we're fully polished and feature complete, showing off the product to David was really fun and exciting. He and his team are ecstatic about our product and want us to potentially want to scale it out later this year. This next week is expo and we're spending this time preparing for that. This means making sure we have our poster, we have nice clothing, etc.

9) Week 8 Update - Ty: Expo Expo Expo. It was a long day of delivering the same shpeal to many people and standing in uncomfortable shoes, but it went really well. We got a lot of good feedback on our project and were able to strike up potential continued development interest with many different groups. I'm glad that it's done and that we're complete on the project. This coming week we have lecture on Wednesday, which will be a big recap and Kevin's unveiling of our final write up for the term and project. It will be a bitter-sweet process to have to write another, more comprehensive, paper and have to say goodbye to our project.

10) Week 9 Update - Ty: This week our group met to divvy out responsibilities on the final paper/presentation. We each have equal sections to write and will be filming our presentation this Wednesday. Other than that there's no more development being done and we're focusing on ending the term well.

11) Week 10 Update - Ty: This was our last week for capstone as a whole. We met once on Wednesday to film the remaining portions of our video and discuss the document. Over the coming week we'll merge our individual branches for our paper and be completely finished! It was a fun class and I'm glad I had the Jasper and Alex as teammates, Jon as a TA, and David's lab as a sponsor :)

12) Week 10 Update - Jasper: It's the end of Senior Capstone! Last week, we got our final report set up and assigned tasks. This week, we finished our final video presentation. Now all that's left to do is finish up the final report and write group evaluations. It's been a great year, and I'm really pleased with how Capstone has gone.

VII. Final Poster

VIII. Project Documentation

A. How it works

Our system consists of three main views. The Filter View allows the user to select information of interest. The Map View displays selected information on a map. The Graph View shows graphs of the selected information. When the user selects information of interest in the Filter View, the website posts these parameters to the server, as seen in Figure 9. A Django script on the server uses these parameters to form a database query. The MySQL database returns the requested data to the server, which formats the data into the Map and Graph Views.

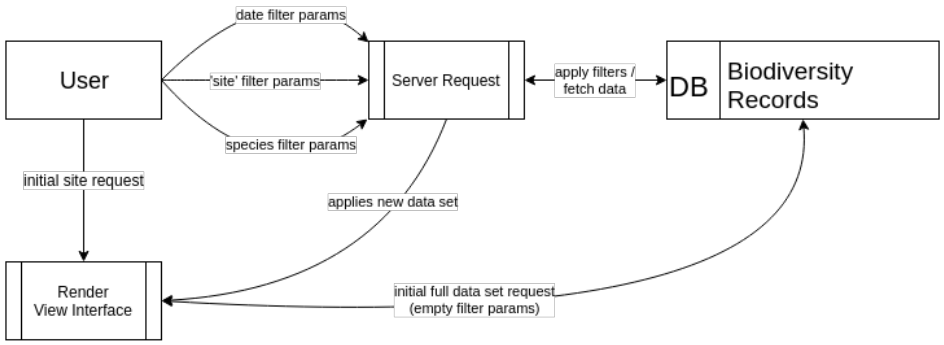

Fig. 11: Dataflow diagram. The server forms filter parameters into a database request, the results of which are passed to the frontend views.

B. How to install it

Installing our web app is actually pretty simple:

- Install the `mod_wsgi` package for Apache, which is a module that implements a compliant interface for hosting Python based web apps.
- Install the Python package manager, Pip.
- Using Pip, install the framework Django: `pip install Django`
- Clone the code repository and install MySQL.
- Finally, at the command-line run `./manage.py make migrations; ./manage.py migrate` to generate database tables based on our ORM.

C. How to run it

Running the server at the command line is as easy as typing `./manage.py runserver`. To deploy it via apache, simply configure apache to point to the project folder, the `mod_wsgi` installation, and the Python binary. This will host it statically and will run the necessary scripts on every load of the page.

D. System requirements

As hinted at in the previous sections, there are certain technologies which are required to run this web app:

- MySQL server
- Django v1.6.11
- Python 2 or higher
- Apache
- mod_wsgi

IX. New Technology

X. What We Learned

A. Jasper

I am a firm believer in the educational benefits of long-term projects. When I am a teacher, I will be the asshole teacher who assigns lots of projects. My students will hate me. But at the end of the year, they will be able to look back at all of the skills they could not have learned any other way. I know from experience with my thesis and this project.

In the course of this project, I learned technical information on the fly. When I needed to know how to use Django, I read up and the documentation and figured it out. When I needed to know how to use Javascript libraries, I looked up examples and figured it out. Besides now having these skills, I have honed my ability to pick up new technical skills. I have also acquired an appreciation for the patience of a long-term project. This project was a marathon, not a sprint. It seems daunting at first, but from the other side, it is a lot more relaxed than a sprint. Having many small deadlines along the way keeps things from building up, and you do not find yourself so out of breath at the end.

Of course, I did not do this on my own. When I needed help, I had it from my teammates, who were excellent. If this project taught me one thing, it is the importance of a good team. I have rarely been so pleased with a group for group work. I will definitely be making future career decisions based on how well I could work with my prospective colleagues. The most important thing we did to foster good teamwork was to stay in contact. Thanks to weekly meetings, it was hard to ever get more than a week behind.

My only regret is that this project became more implementation than research. Our primary objective was to make a product that our client was happy with, which we did. But we did not necessarily contribute generalizable knowledge to the field of visualization. I think this comes of having a sponsor whose area of research was outside of Computer Science. As a result, our solution contributed more to biology than Computer Science. In other words, being a bit of an academic, I am a bit disappointed that my work is actually of practical use.

B. Ty

Contrary to what I expected going in, I ended up learning a lot through my capstone project. Initially I assumed it'd be just another web development project on large set of data, which is something that I've done before many times. Looking back I can say that I'm pleasantly surprised by how much I gained in this process and the experiences I had.

Honestly, this project taught me the importance of having a solid team. A solid team isn't just people you enjoy to work with, which I did, but it means people that you can count on. It's safe to say every student, computer science or otherwise, has a go-to horror story for some kind of group work project that they couldn't wait to be done with. This year was a testament to the fact that good groups do exist and reinstilled my faith in group projects. No matter the work, deadline, or curveball- I knew I could count on my team to get things done.

Outside of that I learned some cool new technology this term. My primary gain in this area was definitely in the use of Python as a language for a standalone web app. I've done plenty of web development in my time, heck it's been my primary job throughout college, but I still love to learn new things and new technologies. The use of Python was a compromise between all members of the group, since some of us were familiar with web development and some of us were more familiar with Python as a language. Since working on this project I've been able to talk about the experience and my work in Python as a web language and it's been very beneficial for me.

Lastly, the most stand-out thing that I learned this year I'd have to say is the technical writing skills and a proficiency in laTex. We've done a ton of writing in this class and almost all of it was done through the use of laTex. I've gotten much better at putting words to paper, so to speak, while being appropriately technical in my explanations of concepts. laTex, however, has grown on me significantly. I first learned it this year due to the requirement with this class and I have come to love it very much and use it for almost all writing projects.

Looking back on this project, the only thing I can say I wish we did differently is put more emphasis on our stretch goals. This is hard to say, because it was a very tough year for all of us and we put the most we could into the project. With hindsight being 20/20 and all it's easy to say we should have spent less time polishing lesser aspects, but in the moment it felt right and you can't fault us for that. The stretch goals were to simply make cooler graphs and statistics, but the app came first and we wanted it to be our best.

C. Alex