Vertebrate Map Visualization Project

CS 4624 Multimedia, Hypertext, and Information Access
Interim Report

Virginia Tech, Blacksburg VA 24061
12 May, 2019

Instructor: Dr. Edward Fox
Client: Dr. Meryl Mims
Team: Courtney Duncan, Christian Garcia-Neal, Wasay Mehdi, Andre Urcia

Table of Contents

Table of Figures

Table of Tables

1. Executive Summary

Our client, Dr. Mims, and a team of researchers collected trait data on lesser-known vertebrate species in the northwestern United States. The goal of this research is to find links from trait to climate change vulnerability. She then published her data in a report that was made available through VertNet. Since the research comes from publicly available museum records it is only fitting to create a publicly accessible website to not only access the research but to engage the public on this important issue.

The goal of our project is to make a multiple page website with relative quick links, resources, and research all attached to their respective vertebrate/species. We also made sortable lists of the species based off of their trait data. Also a developers manual is included with our website will be a manual on how to extend or maintain the website for future use and extensibility when we are no longer working on the website.

Another focus of the website is an informative visualization/infographic map that allows users to investigate the data of the species and their populations in different regions. Different parts of the map should be linked from each species individual page for easy association of information. Advancement on the infographic/visualization map that allows for input to clarify or maintain interest in the relevant data. Easy to understand controls that allow for detailing or generalizing parts of the map to meet criteria for different areas of interest or research.

2. Introduction

VertNet is a National Science Foundation (NSF) funded collaborative project that makes biodiversity data free and available on the web. While some species on the west coast of the United States have been heavily researched, such as salmon, many species are left undocumented and under-researched. This is because there are little game or conservation interests for those species. To help document these species, Dr. Mims and her team of researchers collected data in the Oregon-Washington area of the United States on the lesser known species of amphibians, reptiles, and fishes.

2.1 Objective

The goal of this project is to create a interactive, easy to use website which will display Dr. Mim's research findings clearly to the public. She wanted an application that would cater to a non-technical audience. In addition, this website is intended to raise awareness for a host of under-researched species. Many funds that may go towards these species, usually end up going towards a flagship species that already have a large amount of research done on them, such as the salmon in the North-West Pacific region. One of the hopes of the website, and presenting the research on many of these lesser-known species to the public in a convenient manner, is to then introduce the public to these species that have small amounts of research done on them, and hopefully garner interest so that more research may be done on them as well.

2.2 Client

Dr Meryl Mims is an assistant professor in the Department of Biological Sciences at Virginia Tech, and the head of the Virginia Tech organization Mimslab. She is also an affiliated faculty member with VT's Global Change Center. Dr Mims completed her Master's (2010) and PhD (2015) at the University of Washington in the School of Aquatic and Fishery Sciences. She then served as a Mendenhall Postdoctoral Research Fellow with the U.S. Geological Survey in Corvallis, Oregon, before joining the faculty at Virginia Tech.

Dr. Mims's research investigates how species' traits and environmental attributes interact to influence community and population structure of aquatic organisms. Her research integrates the fields of population, community, and landscape ecology, and she uses a suite of approaches including population and landscape genetics, spatially explicit individual-based models, traits-based inference, species distribution models, and multivariate statistical tools. The overarching goal of her research is to uncover,

understand, and predict differential response of aquatic species to a changing landscape and climate. Dr. Mims works to bridge fundamental work in freshwater population and community ecology with applied conservation and management needs.

For more information on Dr Mims and her research, please visit her website or follow her on Twitter, @merylmims.

2.4 Project Roles

All roles were loosely integrated based on an area of ability at the beginning of the project. No roles mentioned are rigid, and all of the following roles crossover and require a good amount of communication between persons on the team. Roles may shift and vary over time to accommodate mis-estimations of time and difficulty of required work or the compatibility of certain members to that work.

Name	Roles
Courtney Duncan	**Team Leader** Appoints other members various roles for the assignment. Initializes all group activities and documents. Delegates work on presentations and reports based on the work done and familiarized with each member. Administers group meetings. **Scheduling and Correspondence Manager** Sets deadlines, meetings, and reminders for each of the other group members based on their varying schedules. Communicates with client and instructor to set up meetings.
Christian Garcia-Neal	**Website Designer** Main developer on wireframe models. Implementing home page and textualized research page. **Data Visualization** Implementing table sets and rcs index table. Creating sortability by various aspects of each category.
Wasay Medhi	**Technical Lead** Most familiar and experienced with web development. Main source of technical help for other members, also helps coordinate each section of technical development. Implemented most of the non-geographic data parsing.

	Website Designer Created website skeleton using angular framework. Set up page routing and functional aspects of the website. Implementing main filter system.
Andre Urcia	**Interactive Map Designer** Creating interactive map using resources from ESRI's ARCGIS system. Implementing widgets and various capabilities to work with full interactivity. **Geographic Data Visualization** Rendering various mapping and geographic datas usable for the map. Implementing a system to more easily add future information easily as layers to the map, that will automatically update the widgets to work with any new geographic information provided by the client or future project participants.

Table 1: Project Roles for each group member

2.5 Scope

The Vertebrate Map Project for the Pacific Northwest Region is a semester-long project. However, the project as a whole will be extensible beyond the Pacific Northwest Region and can continue to update as more research and data is gathered for each species, beyond the current semester. The main focus of the project is the visualization and interactivity of the data on the web. With extensibility in mind, ease of update also plays a big role in the project. Each aspect of the project must be easy to upkeep in the future, with the functionality from a user standpoint ready by the end of the semester

3. Requirements

3.1 Website

The objective of this project is to create a website that can educate the public about lesser-known vertebrates in the pacific northwest. A public website is essential in spreading information to a large number of people. The website we are going to host must be publicly available so that others can access this anywhere. We will be hosting the website on Dr. Mims' server.

3.2 Data

Dr. Mim's research findings should be displayed in full detail in some place on the site. We decided that we would have a list that would go through all the species studied and their research information.

There were three distinct types of vertebrates that were studied: amphibians, reptiles, and fishes. Each type of vertebrate will be in their own list, respectively. According to Dr. Mims, comparisons and analysis between amphibians, reptiles, and fishes are not necessary because of how different they all are to each other. There is hardly anything to compare in data across fields because most of the categories in each list are different from the other. Filtering will be available for any list so different common groups can be seen all at once for better analysis.

3.3 Map

The map is the most important element of this project. We are responsible for taking all of the written sightings for the different vertebrates and converting them into represented areas. Areas, where certain vertebrates are sighted, should be highlighted and marked in some sort of way so that it is clear to the user where that vertebrate resides.

In addition, the map must be interactive and not just display still images of where the vertebrates are. This means it is important to use a certain map API such as Google Maps or ArcGIS. The objective of having the map be interactive is so that the experience can be as dynamic as possible and the user will be able to consume the content however they want. The more interesting the map is, the more users will pay attention to each of these species, and it will heavily encourage people to use more of the website or even read more into the research being done by Dr Mims. A picture is worth a thousand words.

3.4 Maintenance

The servers, APIs, and other resources used must be open-source or at the very least not have any special licenses to use or require funding. The website will be for public use, so it is important that we avoid legal or monetary issues. Future groups working on the project should not have to worry too much about server issues or API updates, and the website should be able to continue functioning without heavy attention. A backend for maintenance will not be necessary, however, some sort of verification that allows developer access to modify the website must be included. The scope of the project for our group alone is only a semester long, and it is important that future groups be able to easily fix any unforeseen mistakes we make.

3.5 Extensibility

The website itself should be extensible for the future so that once we have completed this project, Dr. Mims can still use the website with modified data sets. It should be possible for Dr. Mims to update the lists of vertebrates and their locations on the map as sightings and information change in the future. The website should be able to end up accommodating not just the Pacific Northwest Region, but the entirety of the world.

4. Design

The purpose of our website is relatively straightforward and that will be reflected in the design of our website. Our website will feature a basic toolbar at the top to navigate between different website pages. The pages are split into 5 main sections: Home, Lists, Map, Research, About.

The Home page is the landing page for the website. It is friendly and aesthetically appealing to get users to want to explore the website. It also includes a brief description of the purpose of the website and the background of the research. The home page also include a few resources to other related websites.

The data section contains three lists with one for fishes, amphibians, and reptiles. Each list features different traits, as each different kind of vertebrate has many different qualities that are of interest but not all traits are applicable across the different kinds of vertebrates. Sorting and filtering is accessible on these pages for optimal analysis.

The map page displays a map of the pacific northwest to start with but can be manipulated with the users controls. The map is interactive so the user can have the best personal experience possible. The map has sorting and filtering capabilities, based on the species trait data, to display multiple species at once.

The research page will give a brief description of the research done by Dr. Mims. This gives reason to the other pages of vertebrates and shows why this information is important.

4.1 Wire Frames

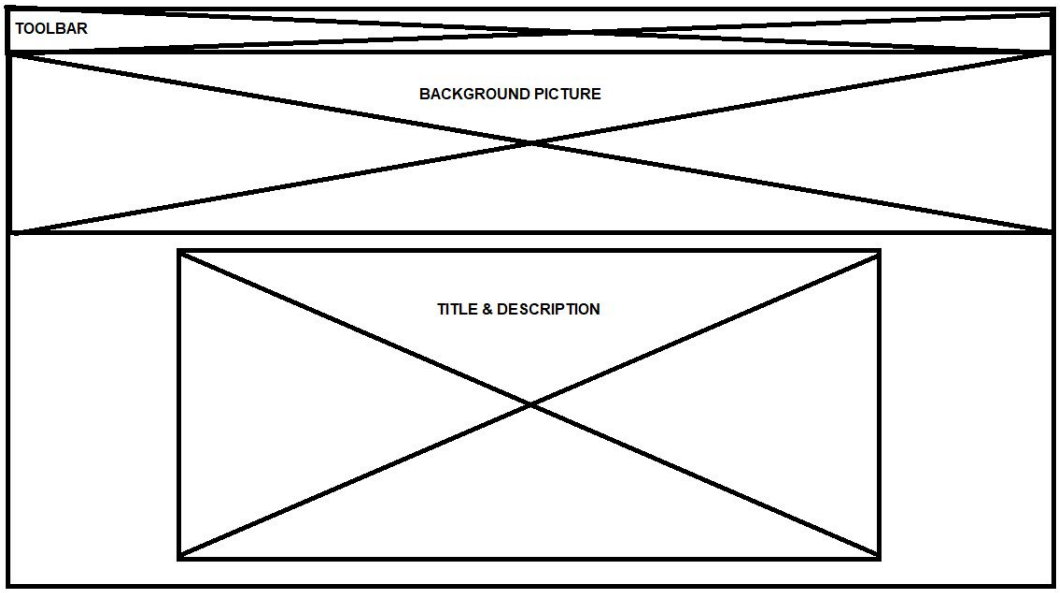

Figure 1: Wireframe for our home page

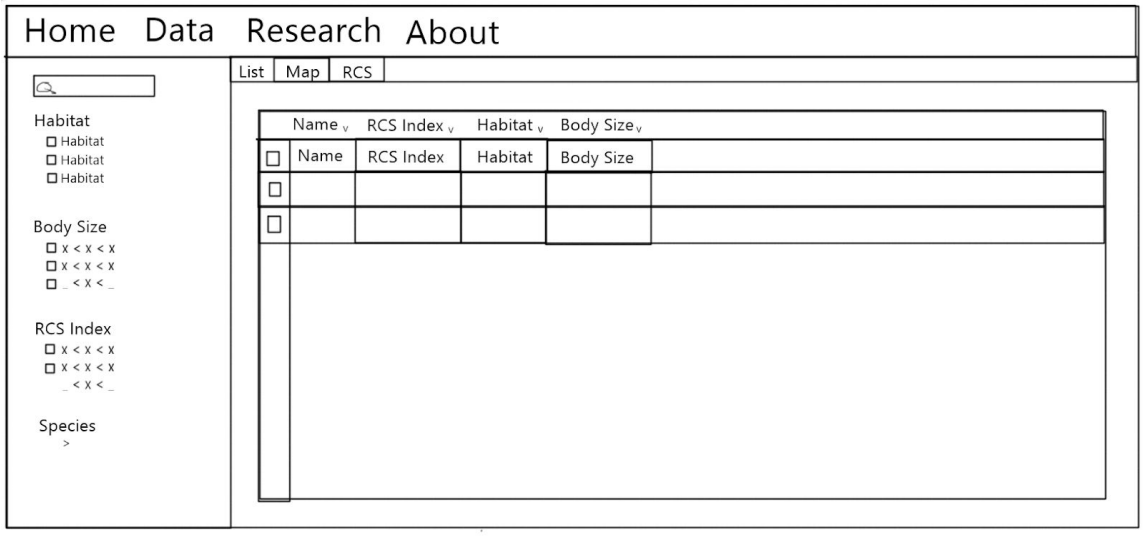

Figure 2: Wireframe for the Data page of our application

5. Implementation

5.1 Data

Dr. Mim's provided us with some files which compiled all of her research findings. The main file used XML format and included all the vertebrate species that were studied and sorted in a hierarchical manner by their taxonomic rank information. For the purposes of our lists and maps, we realized that we wanted all of these species to be seperated with all their information in individual JSON objects, as seen in Figure 3.

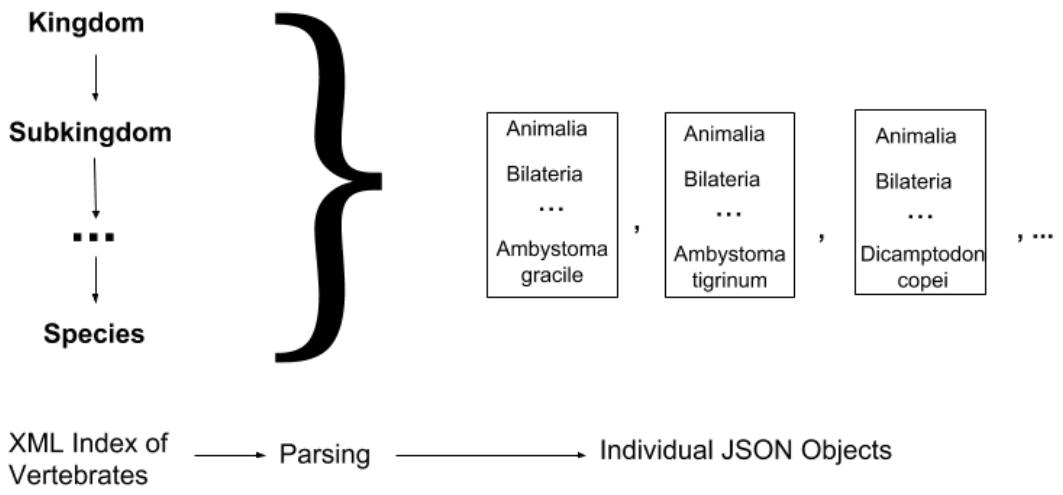

Figure 3: Conversion of taxonomic information into individual JSON objects

To accomplish this, we first converted the XML file into JSON format, so that we could use it with our program more efficiently. To parse the hierarchical structure into flat objects, we used a modified depth-first-search algorithm. We store the data into separate lists based on if they are an amphibian, fish, or reptile. We also store it into a hashmap, which maps the species name to the vertebrate object; this will help us when we are parsing the trait value files.

The files containing her research data were: Amphibian/Reptile/Fish_TraitValues. These files contain her research information in a comma-separated (CSV) format - and linking the species name to her research findings. We can use the map generated in the previous step to grab the vertebrate object from the species name, and then populate

the data properties in that object to match the information we parsed from the trait values file.

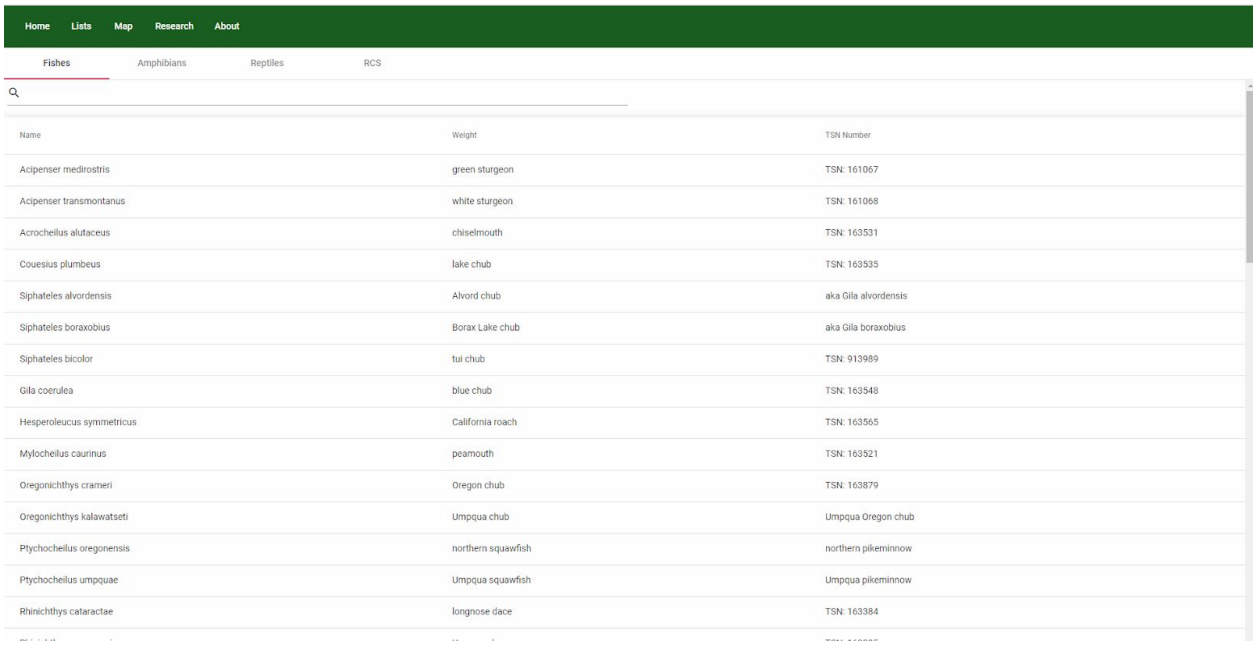

Figure 4: Vertebrate list design on the website

She also provided us with a data file that included all sightings of each species from 1930 to 1999, providing the latitude and longitude of each sighting. We also parse that file and add all of this location information into our flat vertebrate objects. These will help when mapping the vertebrate sightings and also allow us to filter sightings by year, species, and location.

All of these files will also be included in our project code inside of src/assets. To read more about the object formats used in this project, please read section 8.3 of the Developer's Guide.

5.2 Website

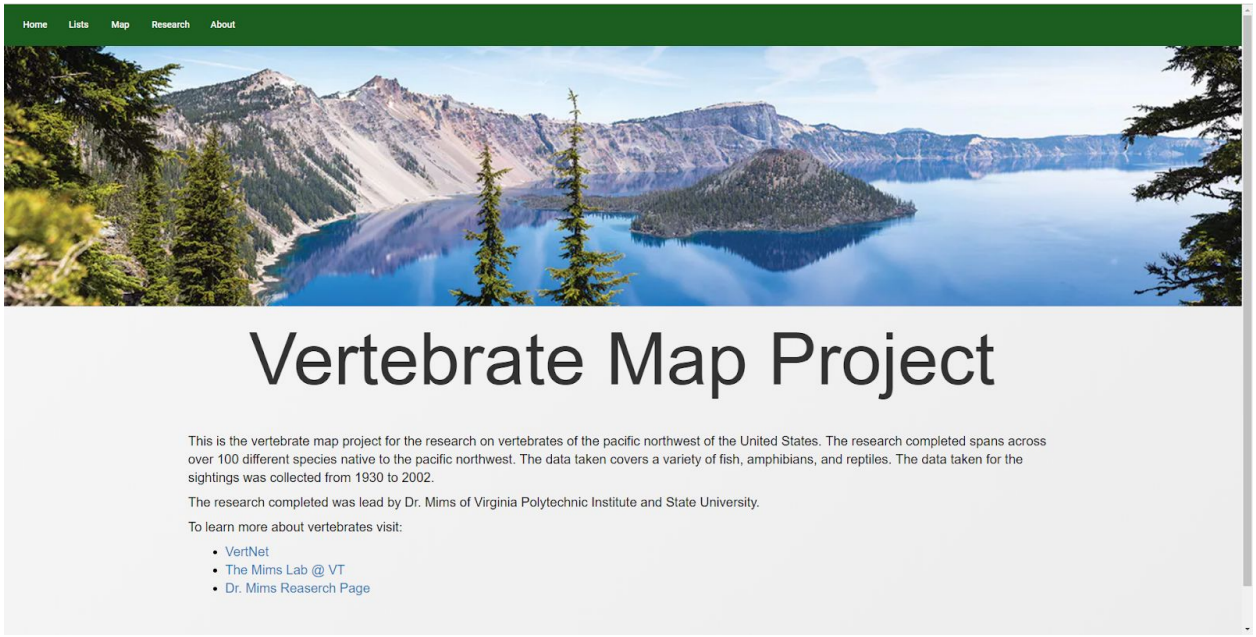

Figure 5: The homepage design on the website

For the longevity of this project, it seemed appropriate that we use the latest technologies. That is why we decided on using Angular 6 - the latest version of Angular from Google. Another benefit of using Angular 6 is that it separates the responsibilities over multiple modules - making it easier for us to work on the website simultaneously. Angular 6 uses a command line interface which allows you to generate new components, services, and guards through simple command line functions. For more information on how to set up Angular 6 on your machine, please reference section 8.1 (Developer's Guide).

Angular also uses a superset of JavaScript known as TypeScript. However, since most browsers cannot read TypeScript, Angular uses a transpiler to convert the TypeScript into JavaScript before the code is put on the web page. The benefit of using TypeScript is that it allows for strong typing, which leads to a much more robust code. Most errors that would have to be detected at run time with JavaScript can now be detected at compile time - which helps speed up the development process. For more information on TypeScript syntax, please reference section 8.2 (Developer's Guide).

Dr. Mims wanted a modern website design which would be attractive to people visiting it. For the look and feel of the website, we decided to use Google's Angular Material Design. Since google develops both Material Design and Angular - this seemed like the

obvious choice. It has seamless integration with the Angular framework and also gives us two important benefits: modern interfaces with responsive animations, and the ability to transfer onto mobile. Since more and more people access sites through their mobile phones, it is important for the site to be functional on mobile. Material Design automatically adjusts the layout to be easily usable on mobile without taking up development time writing custom css for mobile-specific cases.

5.3 Map

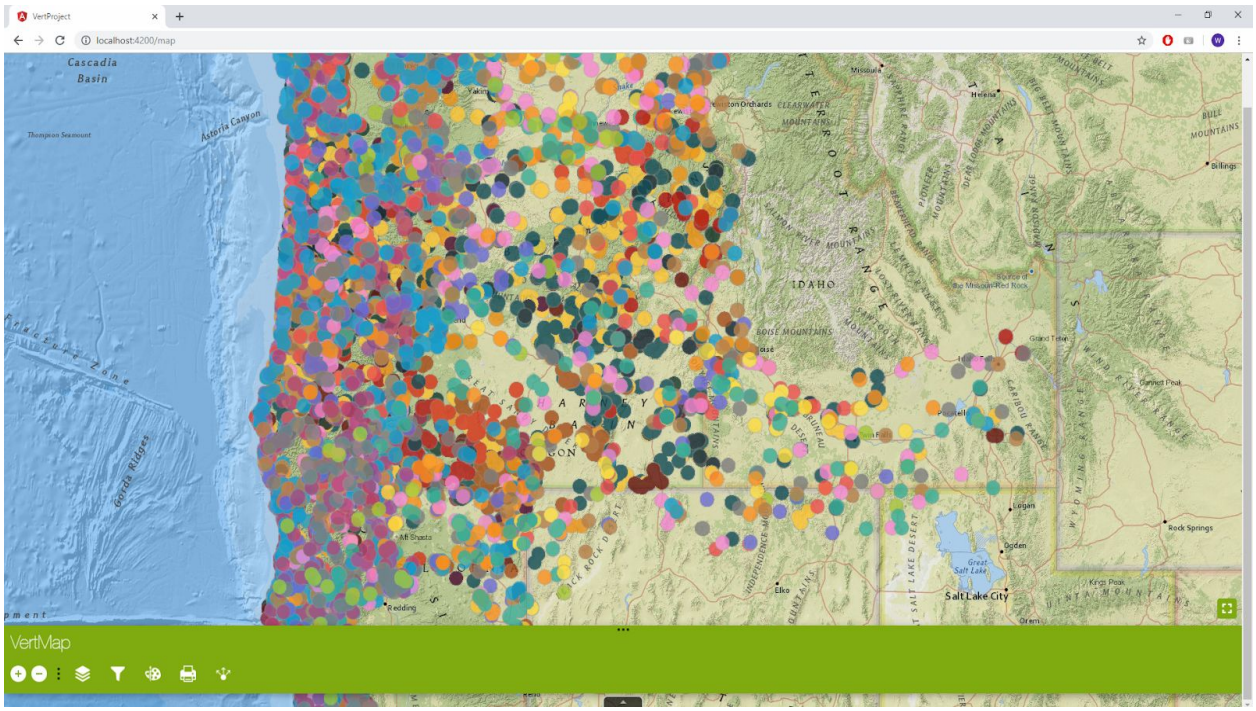

Figure 6: The map design on the website

For the interactive map, we decided on using the JavaScript ArcGIS API. Since TypeScript is a superset of JavaScript, it allows us to use all JavaScript libraries with a little bit of effort. To work correctly with Angular and TypeScript, the JavaScript API must be translated a bit. Coding the interactive map requires more patience than simply making the map and then embedding it onto the website. However, coding the map is necessary to have it work with our filter system.

Some aspects of the map are trivialized through the use of widgets. The API enables you to pull certain tools for a map directly through the web, and because ESRI was the creator of both the API and the widgets, they work relatively seamlessly with the map. The widgets include full-screen capabilities, legends, etc.

Not everything can be solved with widgets. Taking the actual geographic information and turning them into layers for mapping is a big obstacle. Once a shapefile has been turned into a layer, we will simply add that layer to a list that we iterate through and put on the map. The widgets provided by the ESRI will, for the most part, do the rest. However, we will then need to tie the geographic data of individual species back to the filter.

6. Testing

We initially started with doing developer-side testing. Since the purpose of this project is to be usable to the general public, there is no specific audience that we need to test the project on. We are able to test most functionality through the other members of the group, just to see if it works properly. In terms of website design, we consulted Dr. Mims on the look and feel of the wireframes and the website.

We completed UX benchmarks with our website with average students on campus. The UX benchmarks test whether the layout of our website is intuitive to the the average user. Since our website can be accessed by any one, it is important the the website makes sense to both the undergraduate art major and the computer science graduate student. The benchmark was completed by using a latestage version of our website.

In terms of assessing the usability of the site, we lay out a set of tasks for users to complete on the site. They will try and complete the task list on our website without any help, while we note any possible confusion or pain points when using the site. Then we will ask if they have any input on how they feel about the design and usability of the site.

For testing we decided to use a UX target table to get measurable results of how long it takes for users to complete certain tasks.Our projects target user is someone who is interested in biology and our research but they may not use this website again. Because of this we want everything to be intuitive to use on the first use. Or tasks reflect this by having short uncomplicated tasks that are expected to be completed quickly.

For the actual testing we had a pool of 15 people. Some students from other majors and also people of various ages.

Work Role: User Class	UX Goal	UX Measure	Measuring Instrument	UX Metric	Baseline Level	Target Level	Avg. Observed Results (sec)
Someone interested in biology: Using Design for the first time	Ease of Use	Initial User Performance	Access the map page of the website	Average tIme on Task	10 seconds	5 seconds	3
Someone interested in biology: Using Design for the first time	Ease of Use	Initial User Performance	Set the layer to be a certain species	Average tIme on Task	45 seconds	30 seconds	5
Someone interested in biology: Using Design for the first time	Ease of Use	Initial User Performance	Access the list Page	Average tIme on Task	10 seconds	5 seconds	2
Someone interested in biology: Using Design for the first time	Ease of Use	Initial User Performance	Find detailed information on a select species	Average tIme on Task	10 seconds	5 seconds	12
Someone interested in biology: Using Design for the first time	Ease of Use	Initial User Performance	Access the Reptiles List	Average time on task	5 seconds	3 seconds	3

Table 2: UX Target Table

The UX target table above features some of the most common tasks required to use our website. The goal of our UX target table is to judge the ease of use on our on these tasks and to judge how intuitive our design is. On all of our tasks we met the target level except for the task to “Find detailed information on a select species”. Through our observations during our testing we found that some users were not aware that the entries in the list section were clickable to bring up the modals with specific details. This lets us know that in the future we should refine our UX through design patterns. In the future we will add a defined outline and shadow to let the users know that the entries are clickable. Beyond the one abnormality everyone said that completing the other tasks felt intuitive which was apparent by no one needing thinking to to complete tasks, and immediately accomplishing them.

7. Users Manual

7.1 Home page

The landing page is the home page. The home page itself has a brief description of the website and Dr. Mims research with external links to more information. From here and on every page after, the main pages will be able to be accessed through the green bar at the top of the page:

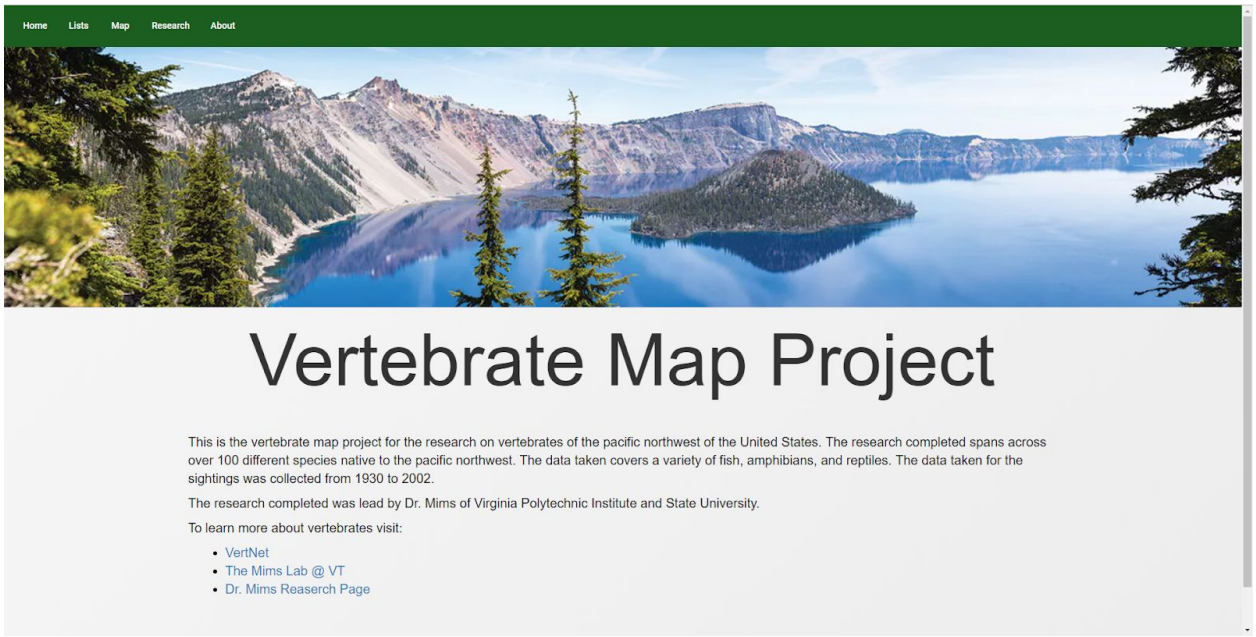

Figure 7: Home screen

7.2 List Data

The map and the listed data are both part of the navigation bar. Redirecting to the lists brings up this page:

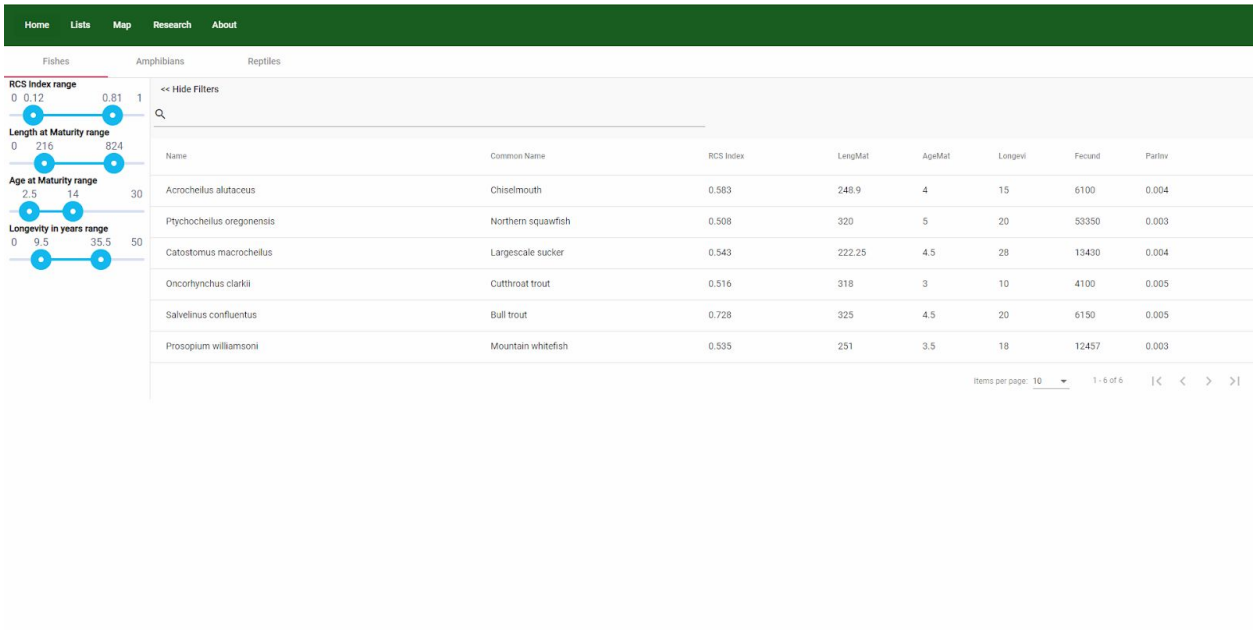

Figure 8: List Data page

The lists page contains a sub-navigation bar for the separate categories. You'll be able to sort things by their categories by clicking on the headers of each column. You can also search for specific species by entering keywords such as their name in the top left corner next to the magnifying glass icon.

For all the relevant data displayed in the list, you can also filter them by ranges using the sidebar. The bar can be shown or hidden using the upper left button if you want just a better view of the list. This also helps make the site look better on mobile phones.

The research navigation link redirects to a page with text and links directly to Dr Mims' and other relevant research, while the about tab also contains mainly text. For both circumstances, the only interaction needed to be understood is the header navigation bar.

7.3 Map

The map navigation link will pull up the following page:

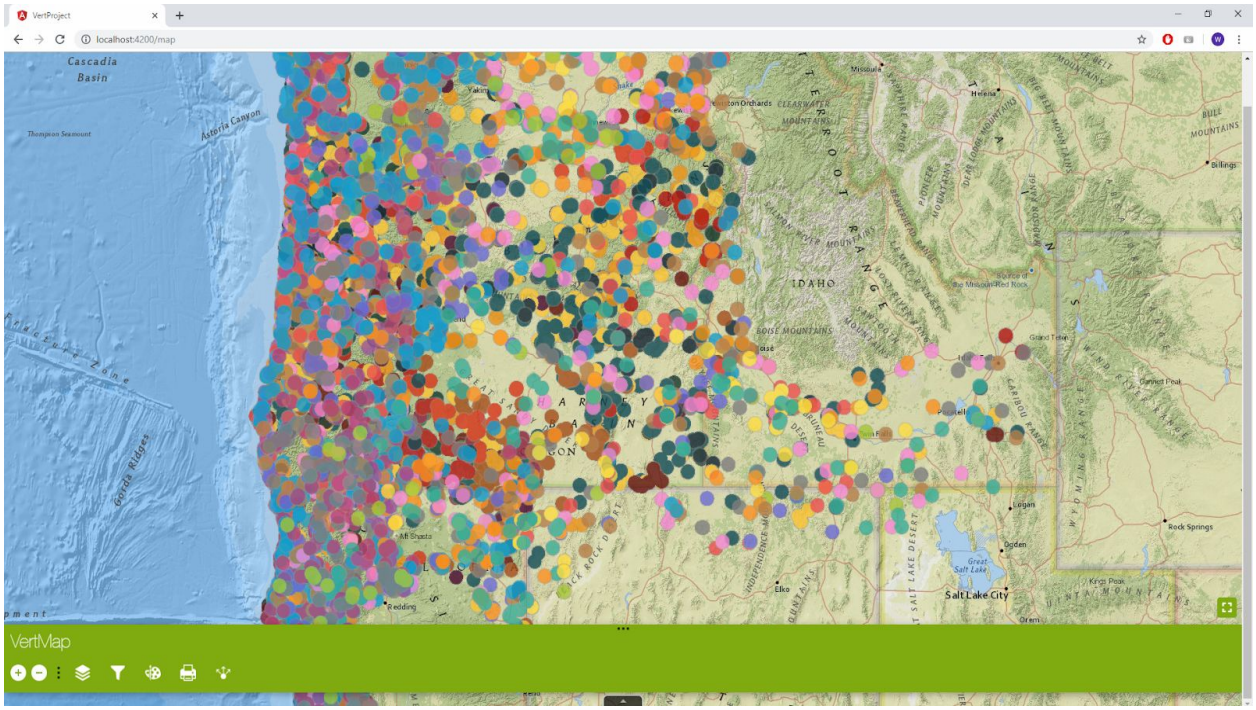

Figure 9: Map landing page

This page purely contains the map and all entities associated directly with it, inclusive of a key and an in map filter system. You may need to scroll to view the full map, or you can use the button in the top right corner to enter full screen mode with the map. The key will show the layers on the map in relevant detail, as you use you scroll wheel, or the buttons in the top left to zoom in and out, the key will update itself automatically to only show relevant layers. On the bottom left is a view filter for the layers on the key, by clicking the eye, you can hide and show layers as is convenient. All of these tools are available in fullscreen mode.

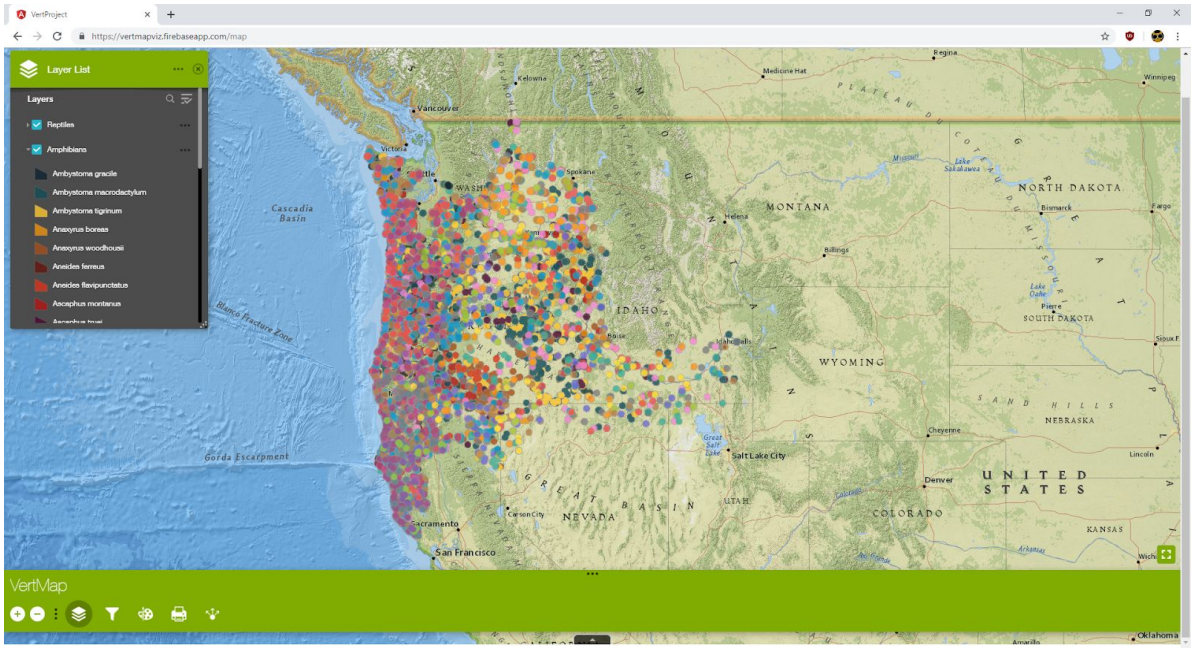

Figure 10: Map layer page

To access the map layers click the layers button highlighted above. This lets users show different layers of species. To see all sightings of an individual species click a species dot. This pop up also contains information about the species such as body length, fecundity, etc.

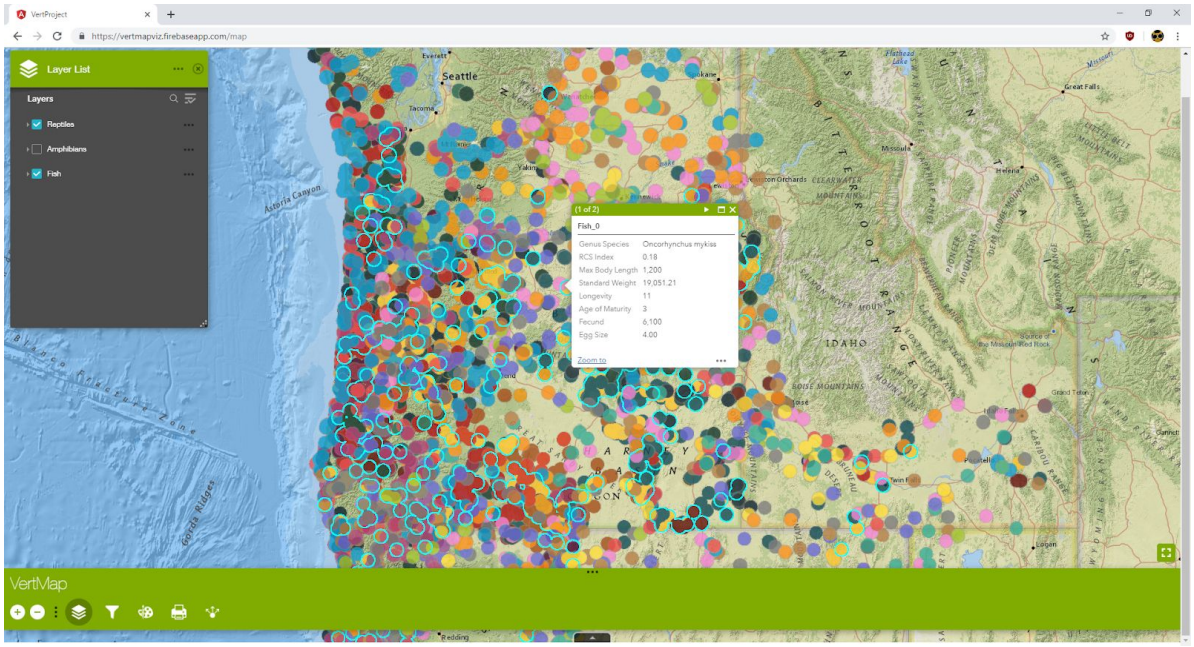

Figure 11: Individual Species Selection

7.4 Research

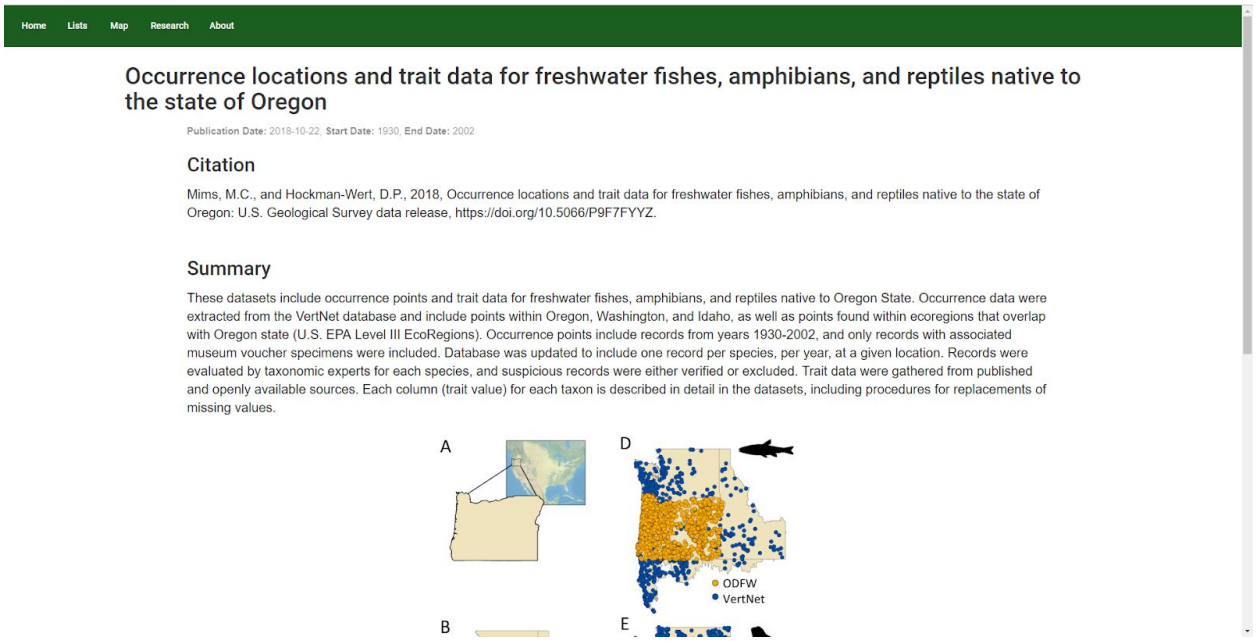

Figure 12: Research page

The research page contains a brief summary of Dr. Mims' research and the purpose of her research. At the bottom of the page there are external links to the full sources of Dr. Mims' research.

7.5 About

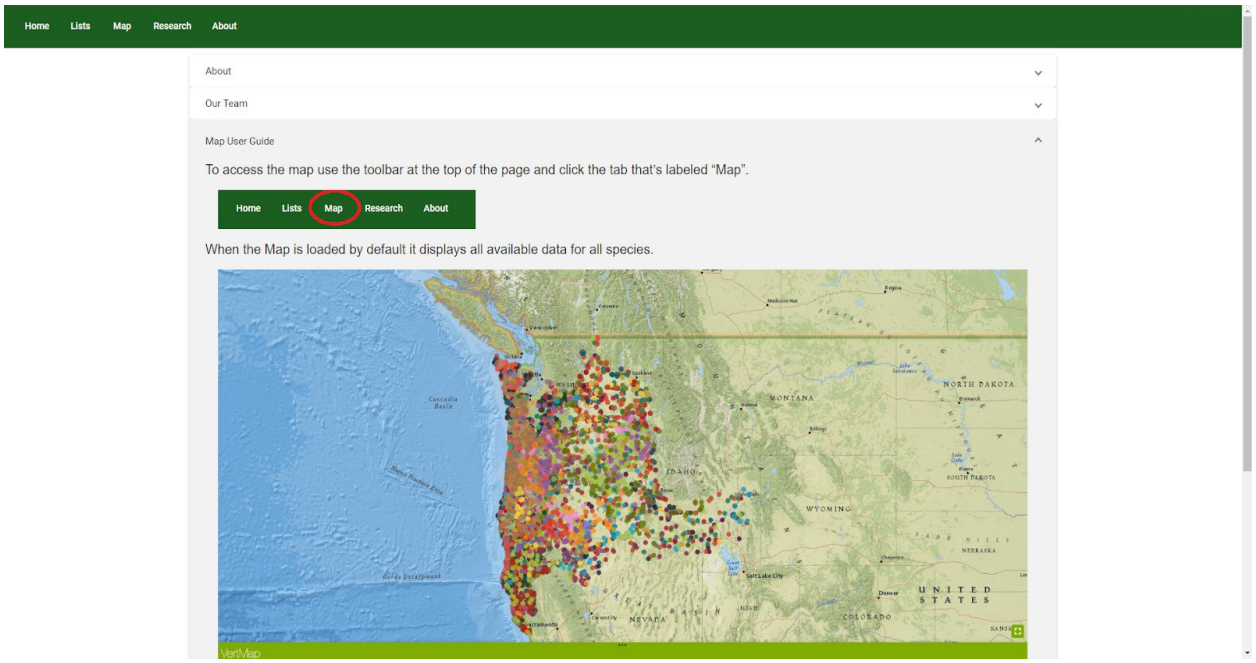

Figure 13: About page

This about page contains all the extra information needed concerning the website such as what the website is about, our team information and contact, and a map user guide. The team information has our roles and contacts, so people can get in touch with us if needed. The map user guide will probably be the most helpful portion of this section, since some of the interactions with the map might not be intuitive to everyone.

8. Developers Manual

8.1 Setting Up The Project

8.1.1 Setting up NodeJS and NPM

To set up the repository, you will first need to install NodeJS onto your machine. This will give you access to the “npm” utility that allows you to install javascript packages through the command line.

For windows: the installation process requires you to download the packages from https://nodejs.org/en/ and just follow the steps after running the .exe file.

For other systems: follow the guide provided by the nodejs team at https://nodejs.org/en/download/package-manager/

8.1.2 Installing the Angular CLI and ARCgis

To install the Angular CLI, in the command line, run:

```
npm install -g @angular/cli
```

Then, to add the needed packages to your repository. Open command line inside of the project folder, where package.json is located:

```
npm install --save esri-loader
npm install --save @types/arcgis-js-api
npm install
ng add @angular/material
```

To run the server, just run `ng serve` in the same folder. The site will be available at http://localhost:4200.

8.1.3 Hosting the server

While there is no one way to host the server, to actually get the files that you need to host the server, run the terminal command:

```
ng build --prod --aot
```

This will add all the relevant files into your ./dist/ folder. Angular transpiles into plain JavaScript and CSS. This is because most browsers can comfortably read JS and CSS, whereas TypeScript serves as a layer on top to help us develop a more clean project. By running the above command, it also obfuscates and uses ahead of time compilation to produce efficient JavaScript code. You can read more about it at: https://angular.io/guide/aot-compiler.

8.1.4 Installing IntelliJ IDEA*

*This is an optional step. For our project we used IntelliJ IDEA to develop our project. However, you need IntelliJ Ultimate Edition to be able to code an Angular project in it. If you don't have access to IntelliJ Ultimate Edition, we recommend using VSCode.

To install IntelliJ onto your machine, please follow the download here: https://www.jetbrains.com/idea/download/

To set up the project using IntelliJ, go to *File > New > Project From Existing Sources,* as seen in Figure 11. Pick the correct folder, and just click next using the default options in the loading screen. Since we will be using the command line interface to run our project, we only need to use IntelliJ as a text editor.

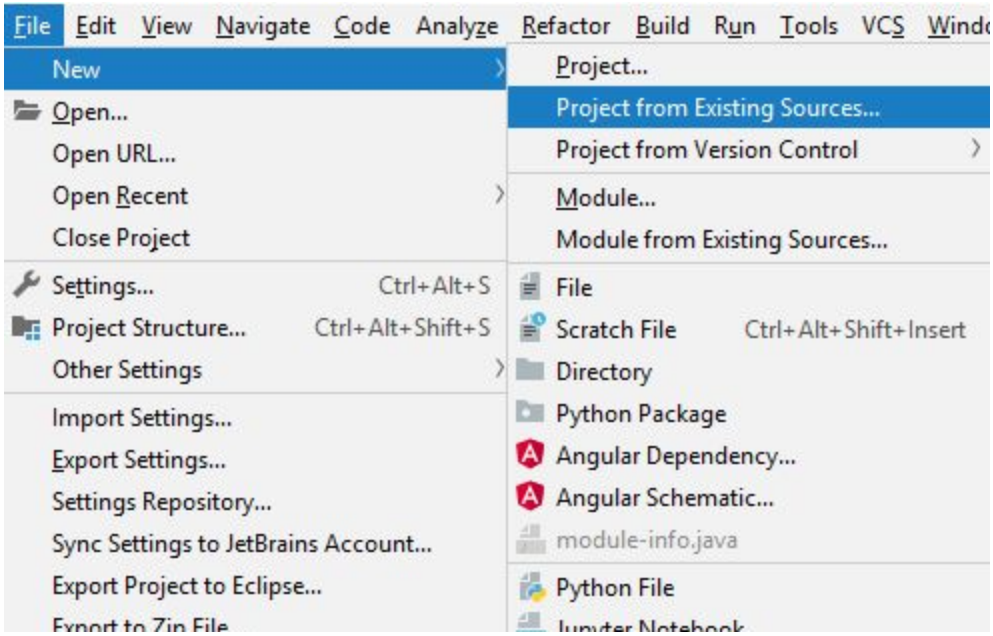

Figure 14: Setting up the repository in IntelliJ

8.2 Developing in Angular 6

Angular 6 uses a combination of TypeScript (similar to JavaScript), HTML, and CSS. For the purposes of this report, this section will give a very quick overview of how Angular 6 works, to help your understanding for the next section.

Angular 6 creates the website as a combination of components, that are saved as modules in their individual folders. Each page of the website is a combination of components put together. To create a new component, you can use ng generate component <component-name> in the command line to make a new component. It will generate four files in a new folder:

component-name
> component-name.ts
> component-name.spec.ts
> component-name.css
> component-name.html

The first *.ts* file contains all the pertinent TypeScript/JavaScript code that is pertinent to that component. The HTML file contains the layout information for the component, and uses Angular's variable binding to allow TypeScript variables to be visible on screen.

The CSS file contains the styling information for the HTML file. The *.spec.ts* file is a test file.

A service acts as a utility class that can be injected into any other class through their constructor. To create a service, run:

ng generate service <service-name>

A service is a single TypeScript file inside of the *services* folder. A service is used when you want to store data or functions that need to be accessible by multiple components inside of the application. Such as making an HTTP request or a request key that needs to be stored.

```
//Injects the global DataParserService instance into the constructor
constructor(private parser: DataParserService) { }
```

To learn more in depth about how to program using the Angular CLI please visit the official documentation at: https://angular.io/cli

8.3 Overview of the Project Code

8.3.1 Overview of the components

Components

- Data
    - This component represents the [List] tab page, it contains the three sections for the three separate types of data lists that will be displayed on the site. It also handles the parsing of the data before it hands it over to the data-list component to display. It also sends the information that will be used to build the filters in each specific list.
- Data-list
    - This component displays a list of data. The type of data for each species varies, and is passed in from the data component. This component is re-used in three sections of the Data component, just with seperate lists being passed into it. It also handles sorting and searching of the data. For each of the attributes, you can filter them by a range
- Esri-map

    - This component is inside of the [Map] tab of our website. It uses ArcGIS's map API to display a map with all of the vertebrate data. It simply contains a link to our map
- Home
    - This is the landing page of our application, under the [Home] tab. It has a simple and inviting design and links users to related sites. It doesn't do anything
- Fish-modal
    - This represents the pop up that will show up if you click on any of the items in a list. It is called fish-modal but serves as the modal for all different types of vertebrates in their respective lists.
- Research
    - This tab goes over Dr. Mim's research paper in more detail and also gives credits to everyone that was involved in the research paper.
- Bios
    - This contains the information that displays the map user guide and information about us (the developers).
- Toolbar
    - The very top toolbar, contains router links to all the tabs and components of the project.

Services

- DataParserService
    - Loads the JSON files from the assets folder, and also has some helper functions to parse the data.

8.3.2 Overview of the object models

The main class we use is the Vertebrate class. Most of the values are loaded from the original XML file and mapped into a flat Vertebrate object for each species. This also means that there are multiple species that share class/genus names.

Vertebrate

- `superclass: string;`
- `common_superclass_name: any;`
- `clazz: string;`
- `common_class_name: any;`

- genus: string;
- common_genus_name: any;
- species: string;
- common_species_name: string;
- tsn_number: string;
- companion : InfoCompanion = {};
- rcs_index: number;

InfoCompanion acts as a superclass to the possible companion types. Since Fish, Reptiles, and Amphibians differ so greatly in the trait values, they have their own unique information companions. The variable names are drawn directly from the CSV files, as shown in the next section.

The RCS index, as described earlier in this

8.4 Overview of the Data

8.4.1 The Data Files

The main data file contains the taxonomic information for all the species that were studied. The format of the relevant part of this file is:

```
<taxonrn>Class</taxonrn>
<taxonrv>Amphibia</taxonrv>
<common>Amphibians</common>
<taxoncl>
    <taxonrn>Order</taxonrn>
    <taxonrv>Caudata</taxonrv>
    <common>Salamanders</common>
  - <taxoncl>
        <taxonrn>Family</taxonrn>
        <taxonrv>Ambystomatidae</taxonrv>
        <common>Mole Salamanders</common>
      - <taxoncl>
            <taxonrn>Genus</taxonrn>
            <taxonrv>Ambystoma</taxonrv>
            <common>Mole Salamanders</common>
          - <taxoncl>
                <taxonrn>Species</taxonrn>
                <taxonrv>Ambystoma gracile</taxonrv>
                <common>Northwestern Salamander</common>
                <common>TSN: 173597</common>
            </taxoncl>
```


Figure 15: Sample XML raw data

…

For the trait value specific data, it is stored in comma-separated files. Each value populates a new

Amphibians:
Species,AdultmaxSVLmm_F,ClutchSize,ClutchpY,Fecundity,EggDiam,ParInv,ParInvStd,ParInvCat,EggIncub,PaedPhase,FemMatY,MaxLongY,LarvDevd,SeasBreeding,SeasBreedDur

Reptiles:
Species,AdultMassg,MaxLongY,FemMatd,FemMaty,ClutchSize,ClutchpY,Fecundity,ReprodMode,ReprodBin,EggMassg,Egg/AdultMass,ParInvStd,ParInv,ParInvCat,IncGestd,BirthWeightg

Fish:
Species,MaxBodyL,StdWt_a,SdtWt_b,StdWeight,LengMat,Longevi,AgeMat,Fecund,EggSize,ParInv,ParInvStd

This lists the species name first, as Genus species, and then the trait values for that respective species afterwards. It is also why it is so important as to store a direct reference from the species name to the vertebrate object.

The last file we received was one with all the times the species' location was recorded since 1930.

TaxaLocation:
Taxon,Species,Year,Latitude,Longitude.

Since there is the possibility of multiple locations being recorded, this information is stored as an array inside of Vertebrate. Most vertebrates have up to 200 sets of location information.

8.4.2 Parsing the data

First, all the data was converted into JSON using online utilities that convert from XML to JSON or from CSV to JSON for free. The JSON format made it possible to work with them in our TypeScript application.

First, we need to parse the data for all the Vertebrate species. This is because all the other information files link a species name to their trait values, so we need all the species names already populated. Since at each level, taxoncl could either be a single value or array. This adds a bit of complexity to our algorithm. At each level, we want to do essentially the same thing: grab the pertinent data, have it saved for the next level, and proceed to the next level. To accomplish this, we use a recursive algorithm as follows:

```
/**
* Recursive algorithm that will parse the whole JSON file into seperate objects
* to represent the fish species
* @param {Vertebrate} v
* @param w - the JSON file object
* @returns {Vertebrate} - the final vertebrate object, can be ignored.
*/
arrayLevel(v : Vertebrate, w) : Vertebrate {
 if(!w) {
   return v;
 } else if(Array.isArray(w)) {
   for(let p of w) {
     v = this.populateInformation(v, w);
     this.arrayLevel(v, p);
   }
 } else {
   v = this.parseArray(v, w);
   return this.arrayLevel(v, w.taxoncl);
```

```
 }
}
```

The “populateInformation” method just populate’s the correct information data inside of Vertebrate, and returns a clone of it. This is because we don’t want to override the information being used in another level with this new information being added.

```
populateInformation(v: Vertebrate, w) {
 v = v.clone();
 switch ((w.taxonrn as string)) {
   case 'Superclass':
     v.superclass = w.taxonrv;
     v.common_superclass_name = w.common;
   case 'Class':
     v.clazz = w.taxonrv
     v.common_class_name = w.common;
     break;
   case 'Genus':
     v.genus = w.taxonrv
     v.common_genus_name = w.common;
     break;
   case 'Species':
     v.species = w.taxonrv;
     v.common_species_name = Array.isArray(w.common) ? w.common[0] : w.common;
     v.tsn_number = Array.isArray(w.common) ? w.common[1] : w.common
     //this is where it adds to the list - we are done with this object.
     this.speciesToVertebrateMap.set(v.species.toLowerCase(), v);
     this.parser.vertebrate_list.push(v);

     break;
 }

 return v;
}
```

Running this algorithm gives us 126 unique vertebrate objects in a list, and a map - that links species names to said object.

Parsing all the comma-seperated files is much easier. On each line, we use the given species name to grab the relevant Vertebrate object, using Map#get functionality provided in TypeScript. Then we just set the Vertebrate’s *information* variable with the new data.

We actually had to manually add the RCS index values from Dr. Mim's research into our code because they weren't provided in a nice format for us to parse. Even though we parsed through 126 unique vertebrates, only 114 of them were relevant to Dr. Mim's research. Any species that didn't have an RCS index didn't have to be displayed on the list - even though the information is shown on the map.

8.4.3 Filtering the data

For the simple word search, we use Angular Material Table's filter functionality. This lets use filter using string matching very efficiently. More information can be found in their material guide at https://material.angular.io/components/table/overview

To create the range slider for filtering, we used a library called ng5-slider. It is a powerful utility that lets us bind our typescript values to the minimum and maximum values of the slider. This helps us filter the list whenever a value in the slider is changed. You can learn more about the specifics here: https://www.npmjs.com/package/ng5-slider

The filtering itself is done by updating our filter constraints, and then applying all filters to our unfiltered list at the same time. Adding new filters based on any value inside of the "component" object of a vertebrate can be done inside of data.component.html inside of /app/data/. Modify the filters array by adding or removing another object.

```
[obj]="[{val:'rcs_index', name:'RCS Index', options:{floor: 0,ceil: 1,step: .01},...
```

- val - What the variable is called inside of it's respective InfoCompanion
- name - What name is going to be displayed on the filters list
- options: The options object that is being passed into the slider (more information can be found inside of the documentation for ng5-slider).

This system allows for adding and removing filters to be relatively simple, while the brunt of the logic is handled by the data-list component once the information is passed in.

9. Lessons Learned

9.1 Timeline

2/1/2019: Approval Deadline

2/2/2019: Team Meeting; Milestone interactive paper prototype

2/9/2019: Team Meeting;

2/16/2019: Team Meeting; Milestone Map Prototype

2/19/2019 : Pres 1; Milestone wire frames

2/23/2019: Team Meeting; Milestone Begin implementing table set

3/2/2019: Team Meeting

3/16/2019: Milestone have website up and running with mock data set

3/19/19 : Pres 2;

3/23/2019: Team Meeting

3/30/2019: Team Meeting Milestone most functionality of website present, begin User Testing

4/6/2019: Team Meeting

4/9/2019: Pres 3; sorting on , website functional, maybe not pretty yet list

4/13/2019: Team Meeting

4/16/2019: Milestone Have an interactive map to see the locations of each species

4/20/2019: Team Meeting Milestone Have the map be able display multiple species

4/27/2019: Team Meeting Milestone Improve the design and website aesthetics,

4/25/2019: FInal Presentation

5/4/2019: Team Meeting

5/8/2019: Project Due

9.2 Problems

One of the problems we faced was due to using the Angular 6 framework. Since it is one of the newer technologies - the ArcGIS API and tutorials are all written in JavaScript and not directly compatible with it. In addition, modifying the code to work with non-ARCGIS parts of the website, such as the filter is a tricky subject as well.

Another problem we faced was the corruption of the publically available data from VertNet that we were trying to gain a hold of. Much of the data we did manage to get was in an illegible format, and was incredibly difficult to make sense of.

9.3 Solutions

Solving the ARCGIS problem involved simply planning and transformation of the JavaScript API example code into TypeScript that works with Angular. Deeper familiarization with the API itself was crucial, as well as understanding how certain widgets work and what properties could be modified for them.

Our client was able to help us with our data problems by sending data that her team had previously extracted from the website for her research.

9.4 Future Work

We had hoped at the beginning of this project that we would be able to seamlessly combine the map and the list in order to have selected entries perit between the two. This combination would make comparing animals with certain traits ranges a lot easier and could help provide insight to the users of this website. Unfortunately we may not be able to complete this during this semester but this would be a good place to start for a later team.

Also since this idea is so extensible there could be future work in processing other species and their data into usable formats. And with these added vertebrates there could be future work in further organising them.

With the mapping it would be nice to separate the large species layers into layers belonging to each species. The main limiting factor to this is time.

Finally in the future a admin login system would be nice to add so that the content of the website could be updated easily.

10. Acknowledgements

We'd like to acknowledge Dr Mims for requesting this project, and for her research and data. Without her work and her team none of this would have happened in the first place. If you'd like to join her in her research, you can apply to participate in the lab here. If you just wish to get in contact with her, you can email her at mims@vt.edu.

Next, we'd also like to thanks the GIS department at Virginia Tech, and specifically one Molly Mcknight, for assisting in the use of the ARCGIS and mapping technologies and techniques.

Finally, we would also like to thank our class and Dr Fox for grading and giving feedback on our work and presentations.

11. References

1. Jollie, M. T. (2018, November 12). Vertebrate. Retrieved March 26, 2019, from https://www.britannica.com/animal/vertebrate
2. Amphibian. (n.d.). Retrieved March 26, 2019, from https://www.thefreedictionary.com/amphibian
3. Parenti, L. R., & Weitzman, S. H. (2018, December 17). Fish. Retrieved March 26, 2019, from https://www.britannica.com/animal/fish
4. Fish. (n.d.). Retrieved March 26, 2019, from https://www.merriam-webster.com/dictionary/fish
5. Ecoregions. (2018, March 27). Retrieved March 26, 2019, from https://www.epa.gov/eco-research/ecoregions
6. Esri. (n.d.). Retrieved March 28, 2019, from https://www.esri.com/en-us/home
7. ArcGIS API for JavaScript. (n.d.). Retrieved February 2, 2019, from https://developers.arcgis.com/javascript/
8. Mims, M. C., Olson, D. H., Pilliod, D. S., & Dunham, J. B. (2018, October 30). Functional and geographic components of risk for climate sensitive vertebrates in the Pacific Northwest, USA. Retrieved from https://www.sciencedirect.com/science/article/pii/S0006320718300922?via%3Dihub
9. Forest and Rangeland Ecosystem Science Center. (2018, October 22). Retrieved from https://www.sciencebase.gov/catalog/item/5bbd20f2e4b0fc368eaeb96a

Appendices

Please reference the attached Appendix folder, VertData, for full appendices.

A.1 Definitions

1. Vertebrates - Vertebrates are animals of the subphylum Vertabrata, which means that they all share the common trait of having a backbone. They are also characterized by the type of muscular system they have, and by their central nervous system partly being enclosed by the backbone.

2. Reptiles - Of the class Reptilia under the subphylum Vertabrata. Typically scally or plated, examples of reptiles include snakes, lizards, crocodiles, turtles, and government employees.

3. Amphibians - Any cold-blooded vertebrate of the class Amphibia, typically living on land but breeding in water. Their aquatic larvae undergo metamorphosis into the adult form. Common examples of this class include newts, salamanders, frogs and toads.

4. Fish - Fish, for the purposes of this project, are any cold-blooded, vertebrate animals with gills that dwell entirely underwater for the full-extent of their natural lives. The term fish is not a taxonomic group, and is not a class of animal like reptiles and amphibians, but instead is more broadly inclusive of several aquatic classes that are just as distinct from each other as terrestrial classes of animal, such as mammals and birds.

5. Northwest Pacific Region - For the purposes of this project, the Northwest Pacific Region is an area that exists outside the bounds of state borders. This region that we are focusing on for the purposes of this study is inclusive of the state of Oregon, but also various ecoregions extending into parts of Washington, Idaho, California, and Nevada. This is due to the fact that state borders were not determined based on the borders of various biomes and regions of inhabitation of various species, and it makes more sense to define this project in those terms.

6. Ecoregion - Ecoregions are areas where ecosystems (and the type, quality, and quantity of environmental resources) are generally similar. Ecoregions are

identified by analyzing the patterns and composition of biotic and abiotic phenomena that affect or reflect differences in ecosystem quality and integrity. These phenomena include geology, landforms, soils, vegetation, climate, land use, wildlife, and hydrology.

7. AdultmaxSVLmm_F - Maximum snout-to-vent length, in mm, for adult females

8. Snout-to-vent-length: a measurement of size taken from the tip of an animal's nose to the opening of the cloaca at the tail base

9. ClutchSize - Maximum number eggs per clutch

10. ClutchpY - Number clutches per year

11. Fecundity - Product of clutch size (ClutchSize) and clutches per year (ClutchpY)

12. eggDiam - Average egg diameter, in mm

13. parInv - Parental investment, calculated as average egg diameter (EggDiam) divided by maximum female snout-to-vent length (AdultmaxSVLmm_F)

14. parInvStd - Parental investment (ParInv), standardized from 0 to 1, where ParInvStd = (ParInv - min(ParInv))/(max(ParInv)-min(ParInv))

15. parInvCat - Categories of standardized parental investment (ParInvStd), for comparability with other taxa, following natural breaks classification

16. eggInCub - Minimum time required for eggs to hatch, in days

17. paedPhase - Paedomorphic phase present in salamander

18. femMatY - Minimum time required to reach female maturation, in years

19. maxLongY - Maximum longevity, in years

20. larvDevD - Minimum time required for larval development, in days

21. seasBreeding - Breeding season, range of months 1 - 12

22. seasBreedDur - Duration of breeding season, in months

Fish Traits

23. maxBodyL - maximum body length, mm

24. stdWt_a - Standard weight parameters from FishBase, where W=aL^b and StdWt_a = a

25. stdWt_ b - Standard weight parameters from FishBase, where W=aL^b and StdWt_b = b

26. stdWeight - Standard weight, calculated as W=aL^b where L=MaxBodyL, a=StdWt_a, and b=StdWt_b

27. lengMat - mean length at maturation, mm

28. lonevi - maximum reported longevity, in years

29. ageMat - female age at maturity, in years

30. fecundity - maximum reported fecundity: number of eggs or offspring perspawning season per female

31. eggSize - egg diameter: mean diameter of mature, fully yolked ovarian oocytes, mm

32. parInv - Parental investment, calculated as average egg diameter (EggSize) divided by maximum body length (MaxBodyL)

33. parInvStd - Standardized parental invesment (ParInv) from 0-1, where ParInvStd = (ParInv - min(ParInv))/(max(ParInv)-min(ParInv))

Reptile Traits

34. adultMassG - Body mass (g) using data from males, females, and/or unspecified adults. Attribute corresponds to "adult_body_mass_g" in Myhrvold et al. 2015

35. maxLongY - Maximum recorded longevity for species. Attribute corresponds to "maximum_longevity_y" in Myhrvold et al. 2015.</

36. femMatd - In cases where seasonal or annual estimates are reported in guidebooks (e.g., "maturation in 2nd year"), maturation age is calculated as minimum age at maturation minus the incubation or gestation period. Attribute corresponds to "female_maturity_d" in Myhrvold et al. 2015.

37. femMaty - Female maturity (FemMatd), converted to years

38. clutchSize - Number of eggs per clutch. Attribute corresponds to "litter_or_clutch_size_n" in Myhrvold et al. 2015.

39. clutchpY - Number of clutches produced per year. Attribute corresponds to "litters_or_clutches_per_y" in Myhrvold et al. 2015.</

40. fecundity - Product of clutch size and number of clutches per year.</

41. reprodMode - Reproductive mode, laying eggs or bearing live young

42. reprodBin - Binary code for Reproductive Mode, where V = 1, O = 0

43. essMassg: number;
44. egg_adultMass - Egg mass (EggMassg) divided by Adult Mass (AdultMassg)</

45. parInvStd - Parental investment, using the Egg/Adult Mass value, standardized from 0 to 1, where ParInvStd = (Egg/Adult Mass - min(Egg/Adult Mass))/(max(Egg/Adult Mass)-min(Egg/Adult Mass))</

46. parInv - Standardized parental investment (ParInvStd), with 1 added for all species with value 1 in ModeBin (viviparous species)

47. parInvCat - Categories of parental investment (ParInv), for comparability with other taxa, following natural breaks classification

48. incGestd - average of incubation and gestation time from original database. Attribute corresponds to "inc_or_gest_d" from Myhrvold et al. 2015.

49. birthWeightG - weight at hatching (g). Attribute corresponds to "birth_or_hatching_weight_g" from Myhrvold et al. 2015

50. ESRI - Esri is an international supplier of geographic information system software, web GIS and geodatabase management applications. Made the ARCGIS mapping API.

51. JSON - JavaScript Object Notation. A format of file, and also the format in which javascript objects are created. Using a JSON file format loads quickly and intuitively into any JavaScript based language. The format looks something like: {“trait”: “value”}.