sharkPulseApps

Mia Hagood, Anhtuan Vuong Tran, Patrick Warner

CS 4624: Multimedia, Hypertext, and Information Access

Instructor: Dr. Edward A. Fox

Clients: Assistant Professor Franceso Ferretti, Graduate Research Assistant Jeremy Jenrette

Virginia Tech, Blacksburg VA 24061

May 9, 2024

Table of Contents

Table of Figures 3
Table of Tables 4
1. Abstract 5
2. Introduction 6
2.1 Problem 6
2.2 Motivation 6
2.3 General Approach 7
3. Requirements 9
3.1 Initial Setup and Codebase Access 9
3.2 Challenges 11
4. Implementation 13
5. Testing, Evaluation, and Assessment 15
6. User's Manual 20
7. Developer's Manual 25
7.1 Dynamic Scripting 25
7.2 Server Environment 26
7.3 Scripting 27
7.4 Species Coverage Script 29
7.5 Conservation Status Script 30
7.6 Top 5 Species Script 31
7.7 Global Affiliate Map Script 33
7.8 Taxonomic Accuracy Table Script 35
7.9 Database Connection Script 36
8. Lessons Learned 38
8.1 Timeline/Schedule 38
8.2 Problems 40
8.3 Solutions 40
8.4 Future Work 40
9. Acknowledgements 42
10. References 43

Table of Figures

Figure 1: Trello Workspace for Project Management on May 1, 2024 8
Figure 2: Example of Initially Static Visualizations on sharkPulse Home Page. 9
Figure 3: “sharkPulse” Database Table 10
Figure 4: “taxonomy3” Database Table 11
Figure 5: Example Intermediate Script 13
Figure 6: Example of iframe HTML in Elementor 14
Figure 7: Elementor Edit Page 14
Figure 8: Top 5 Species Visualization Layout Options Proposed to Clients 15
Figure 9: Website Feedback Form Preview 16
Figure 10: Pilot Survey Question 1 Responses 17
Figure 11: Dynamic sharkPulse Numbers Section in Home Page 20
Figure 12: Dynamic Species Conservation Status Bars 20
Figure 13: Picture Submission Page 21
Figure 14: Pulse Monitor Page 21
Figure 15:Validation Monitor Page 22
Figure 16: Validation Leaderboard Page 22
Figure 17: Identification Guide Page 23
Figure 18: Shark Detector AI Login Page 23
Figure 19: sharkPulse Team/Network Page 24
Figure 20: Publications Page 24
Figure 21: Main Directory File System 27
Figure 22: sharkPulse Database Structure 29
Figure 23: Species Coverage Query 29
Figure 24: Species Coverage Visual 30
Figure 25: Conservation Status Records Query 31
Figure 26: Conservation Status Visual 31
Figure 27: Top 5 Species Query File 32
Figure 28: Final Top 5 Species Visual 33
Figure 29: Global Affiliate Map App 34
Figure 30: Global Affiliate Map Visual 34
Figure 31: Taxonomic Accuracy Table Accuracy Extraction 35
Figure 32: Taxonomic Accuracy Table Visual 36
Figure 33: Database Connection Script 37

Table of Tables

Table 1: Summary of Pilot Survey Responses 19

Table 2: Scripts and Descriptions 25

Table 3: Languages Used 28

Table 4: Timeline of Work Completed 39

Table 5: Acknowledgements 42

1. Abstract

This project is an extension of work that has been done in previous years on the sharkPulse website. sharkPulse was created due to the escalating exploitation of shark species and the difficulty of classifying shark sightings. Due to sharks' low population dynamics, exploitation has only exacerbated the issue and made sharks the most endangered group of marine animals.

sharkPulse retrieves sightings from several sources such as Flickr, Instagram, and user submissions to generate shark population data. The website utilizes WordPress , HTML, and CSS for the front end and R-Shiny, PostgreSQL, and PHP to connect the website to the back end database. The team was tasked with improving the general usability of the site by integrating dynamic data-informed visualizations. The major clients of the project are Assistant Professor Franceso Ferreti from the Virginia Tech Department of Fish and Wildlife Conservation and Graduate Research Assistant Jeremy Jenrette.

The team established regular contact through Slack, scheduled weekly meetings online with both clients, and acquired access to all major code repositories and relevant databases. The team was tasked with creating dynamic and data-informed visualizations, general UI/UX improvements, and stretch goals for improving miscellaneous pages throughout the site. The team developed PHP scripts to model a variety of statistics by dynamically querying the database. These scripts were then sourced directly through the site via the Elementor WordPress module.

All original requirements from the clients have been met as well as some stretch goals established later in the semester. The team created a Leaflet global network map of affiliate links which dynamically sourced the sharkPulse social network groups from an Excel spreadsheet and generated country border markers and links to each country's social network sites as well as a Taxonomic Accuracy Table for the Shark Detector AI. The team created and distributed a survey form to collect user feedback on the general usability of the site which was compiled and sent to the client for future work.

2. Introduction

2.1 Problem

Sharks are one of the most data deficient animals on the planet and because of this it is incredibly difficult to accurately determine the conservation status of threatened species. Shark populations are decreasing, and nearly a quarter of shark species are threatened with extinction. The exploitation of sharks has escalated, making sharks the most threatened group of marine animals. Furthermore, the identification process for shark species is relatively complicated as there is a rather large pool of species, and several key features must be identified to accurately classify a shark [1].

2.2 Motivation

The sharkPulse [1] web application was created to address the issue with deficient shark population data and contribute to identifying the conservation status of shark species. There are various features within the site meant to improve the number of records on shark sightings, validity of classifications, and knowledge of how to correctly identify sharks.

sharkPulse [1] utilizes data mining on popular social media platforms to aggregate images of shark sightings into a global population data set. Users can also submit their own pictures through the website, to be added to the database. These pictures are automatically classified using computer vision and machine learning techniques. Shark sightings are stored in the Validation Monitor where citizen scientists can log into the website to keep track of their progress, classify the sightings, and validate spatial and temporal data. The Validation Monitor appears as a global map sorted by image with icons that users can click on to validate sightings in that icon's location. The sharkPulse team implemented gamification in the form of a Validation Leaderboard to help track users' validation progress and motivate more people to help validate the many records in the database.

Separate from the Validation Monitor, users can explore the Pulse Monitor, which is a global map of all shark sightings in the database. They can also use the Identification Guide, which instructs and informs people on how to classify sharks based on various anatomical

features. On the home page, users can find summaries of the website's features and total records, and altogether, these features contribute to researching and helping shark species worldwide. As the research continues, the sharkPulse team is committed to continually improving the website's functionality, visualizations, dynamic features, and more.

2.3 General Approach

The project goals and approach were primarily formed based off the client's expectations and guidance. The team's priority was to improve the accuracy and dynamics of the data visualizations displayed on the home page, especially in the sharkPulse Numbers section, as well as analyze how the page could be visually improved. All members have been tasked with familiarizing themselves with the codebase and relevant technologies. Tuan was the team leader and main point of contact with the clients and professor, Mia was the lead on front end design and presentations, and Patrick was the lead on back end design and key phases of the project.

Due to flexible requirements, the general approach across the project was quite agile and involved regular phases of client feedback, brainstorming and proposing ideas, and implementing ideas to receive feedback. The different tasks were tracked using a Trello workspace as shown in Figure 1. Tasks were represented by cards and organized into categories for Canvas deliverables, work in progress, future work, and completed work. Each card could be assigned to a team member who would lead the work on that task, and the date feature was utilized when tasks needed to be completed by specific deadlines.

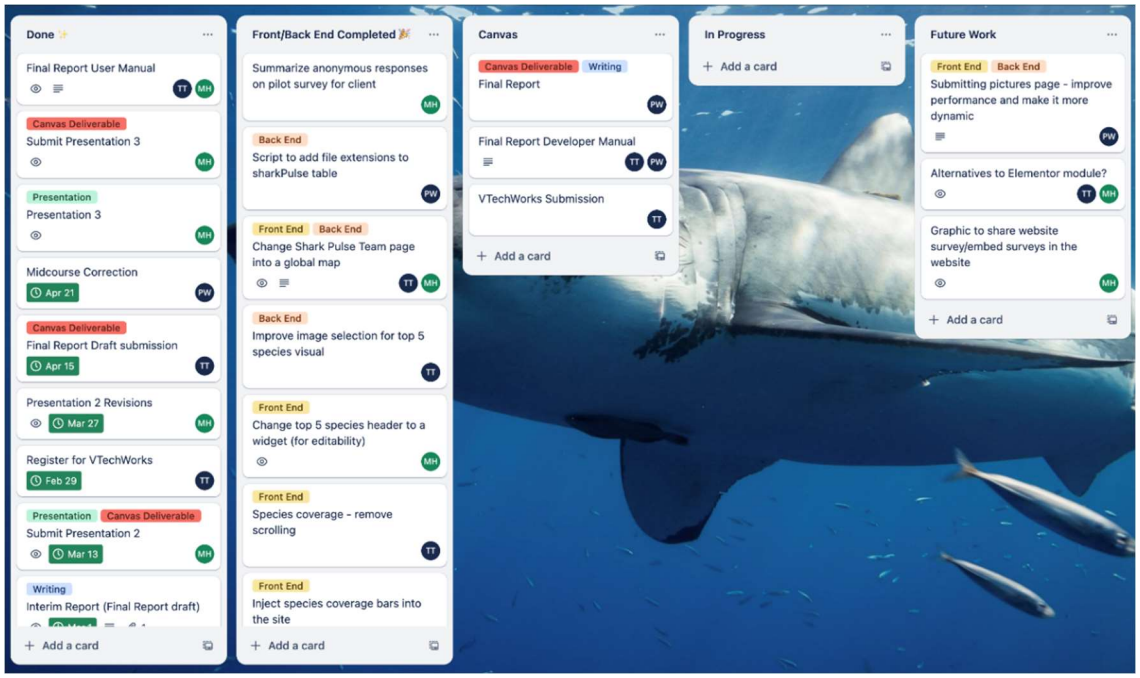


Figure 1: Trello Workspace for Project Management on May 1, 2024

3. Requirements

Immediate project requirements involved integrating dynamic data visualizations into the site and making website editability faster and more flexible. Figure 2 is a screenshot of the initial sharkPulse [1] home page at the start of this project. It is an example of a static visualization where our client requested the team adjust to dynamically source from the database. To make these improvements, each team member created an account for the sharkPulse [1] website and was granted editing access in WordPress [2] and the Elementor module [3]. Authentication credentials were shared to access the sharkPulse database, and relevant scripts were shared through GitHub repositories. Over the course of the project, the team documented the process and maintained the Trello workspace for task management.

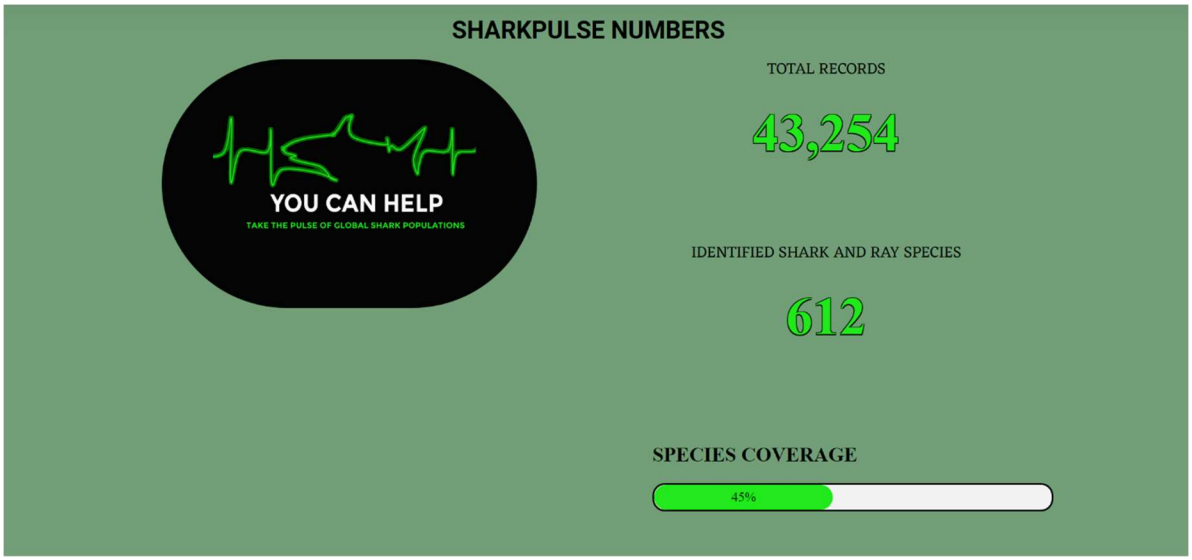


Figure 2: Example of Initially Static Visualizations on sharkPulse Home Page.

3.1 Initial Setup and Codebase Access

Before beginning work on the scripts and explicitly editing the site, the team had to become accustomed to the codebase, database, and relevant server-side logic. Several repositories were shared with the team that contained code for the various monitoring maps on the site as well as the scripts that were used to create a variety of static visualizations. These files were stored on a remote server the team had to access via remote SSH. Establishing a connection

to this server also required connection to the Virginia Tech eduroam network, so the team needed to set up VPN access using the Ivanti Secure Access Client [4]. The team initially spent a set amount of time getting accustomed to the codebase and establishing access to the server. Upon further discussion with the clients, the team investigated the primary database tables involved with the currently implemented visualization for the site. Figures 3 and 4 display the primarily utilized tables and their columns.

sharkpulse
date date
time text
users_email text
species_name text
latitude double precision
longitude double precision
img_name text
notes text
device_type text
given_name text
common_name text
location text
given_date text

Figure 3: “sharkPulse” Database Table

taxonomy3
row.names text
species_name text
order_name.x text
taxonid integer
kingdom_name text
phylum_name text
class_name text
order_name.y text
family_name text
genus_name text

Figure 4: “taxonomy3” Database Table

3.2 Challenges

Most of the team’s initial challenges were due to unfamiliarity with the technology stack involved with the sharkPulse [1] site. The team had no prior experience with R-Shiny [5] or PHP [6] and limited background with R-Shiny [5] and WordPress [2]. Initially, it was challenging to establish contact with both Assistant Professor Ferreti and Jeremy Jenrette. There was also a lag setting up database access for the R-Shiny [5] app in the code repository because the team was initially unaware of the need for a VPN setup. After the first presentation, the team and all clients were able to meet and review the project plans. In this meeting, the project requirements were clarified, and new ones were introduced, so the team members had to become familiar with a new section of the codebase and recreate the timeline milestones to fit the new priorities. Following this major shift, the full team of students and sharkPulse clients scheduled weekly meetings online and established communication through Slack, which made it easier to ask questions and give the clients project updates. In the latter half of the project, scheduling challenges resurfaced and interfered with the online meetings, but the sharkPulse team was still able to communicate remotely through Slack.

There were no major setbacks in terms of development but there were several issues when it came time to integrating the visuals into the site itself. The team ran into difficulty getting the PHP [6] script to cooperate with the Elementor module [3] and inject the data visualization properly. The team struggled with several server-side issues and not having sudo privileges. The server logs for the R-Shiny [5] app were unable to be viewed by the team so debugging it required constant communication with the client. There were also minor formatting issues along the way to adjust according to the clients' feedback.

4. Implementation

The site directly sources PHP [6] scripts which require a web server to run so the team decided to create intermediary scripts to test functionality locally. The intermediate dynamic visualization scripts utilize Node.js [7] to connect to and query the PostgreSQL [8] database and were written in JavaScript [9]. These scripts output raw HTML [10] using the JavaScript [9] File System library and were evaluated by the team for any necessary changes. The code snippet in Figure 5 displays some of the SQL queries used to generate the data-informed visualizations.

```
async function queryConservation(cons_statuses) {
    const total_sql = "SELECT COUNT(*) FROM taxonomy3"
    const total_res = await client.query(total_sql)
    const total_count = Number(total_res.rows[0].count)

    const cons_percentages = []
    percentage_sum = 0
    for(const status of cons_statuses) {
        const conservation_sql = "SELECT SUM(CASE WHEN category LIKE " + "'" + status + "' THEN 1 ELSE 0 END) FROM taxonomy3"
        const conservation_res = await client.query(conservation_sql)
        const conservation_count = Number(conservation_res.rows[0].sum)
        cons_percentages.push(Number((conservation_count/total_count) * 100).toFixed(1))
    }
    return cons_percentages
}

async function queryRecords() {
    const sql = 'SELECT count(distinct sp_id) FROM sharkpulse;';
    const res = await client.query(sql);
    return Number(res.rows[0].count).toLocaleString();
}

async function querySpecies() {
    const sql = 'SELECT count(distinct species_name) FROM sharkpulse;';
    const res = await client.query(sql);
    return Number(res.rows[0].count).toLocaleString();
}
```

Figure 5: Example Intermediate Script

The output of the visualization scripts was compared with the static visuals already present on the website to ensure a reasonable degree of accuracy. After going through an internal review process within the team, the scripts were translated into PHP [6] and copied into the server-side environment using the terminal. All new or edited scripts would also be pushed to the appropriate GitHub repository with a descriptive commit message to help document the script versions and project progress. To connect the server-side back end to the front end of the website, each Elementor widget [3] on the site containing a visualization uses an iframe that sources its respective visualization script as shown in Figure 6.

```
<iframe style="border: 0; width
    :100%; height: 700px;
    overflow: auto;" src="https
    ://sp2.cs.vt.edu/dynamic
    /topfive.php"></iframe>
```

Figure 6: Example of iframe HTML in Elementor

The team then injected the visualizations and rearranged the site within an edit draft using the Elementor module [3]. After meeting with the client to discuss and receive feedback for the changes, the team would publish the edits to the public website. The editor allowed for mistakes in the publishing process by tracking a version history and having a feature to revert to a previous version if necessary. Figure 7 is an example of injecting the PHP [6] script into an Elementor HTML [3] code widget.

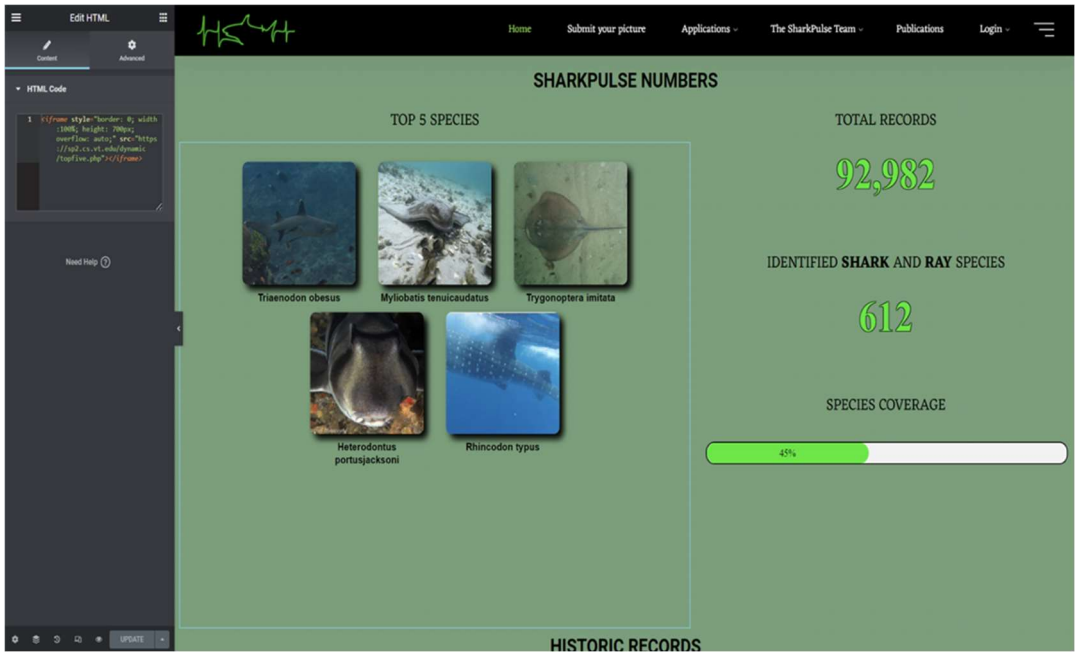


Figure 7: Elementor Edit Page

5. Testing, Evaluation, and Assessment

As the team developed the SQL queries and PHP [6] scripts, local testing and evaluation of accuracy was conducted by comparing the visualization outputs with the current website's visualizations. The aesthetics of visualizations were deliberated on and evaluated between the team and clients by sharing intermediary screenshots or saving drafts in the Elementor module [3] in WordPress [2] before publishing any new final visualization to the sharkPulse website [1]. For example, when proposing potential changes, the team would present visual comparisons such as the one in Figure 8 for the clients to evaluate and provide feedback on. This approach made it easier for clients to assess the pros and cons of the implementations we proposed.

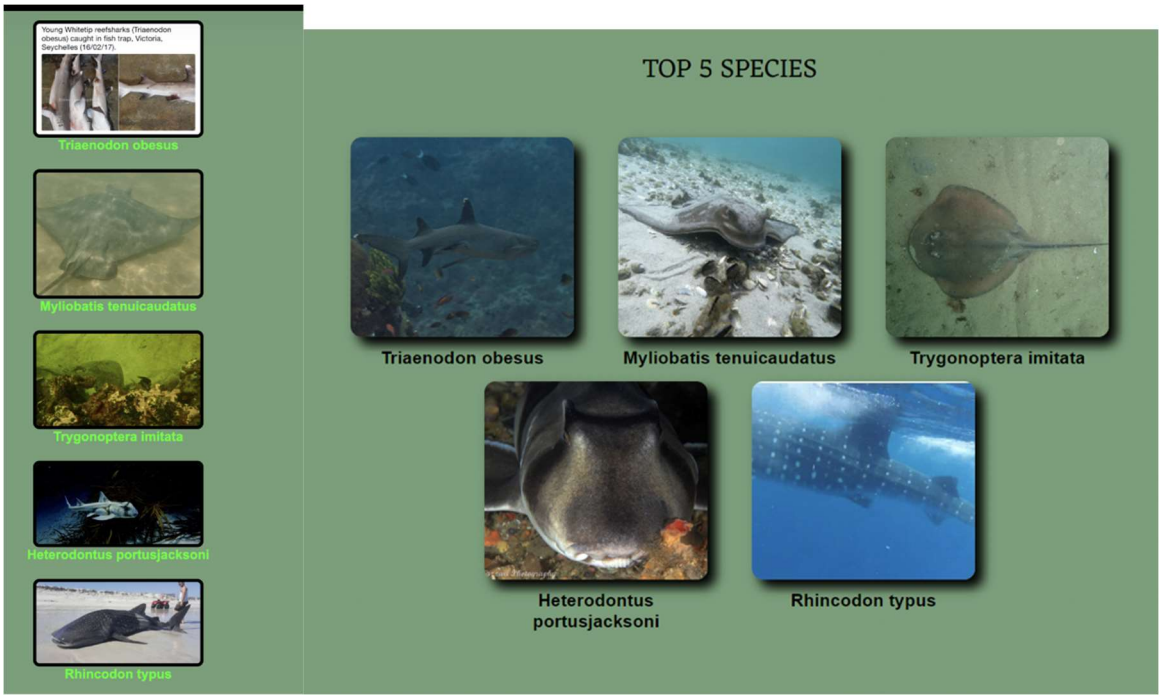
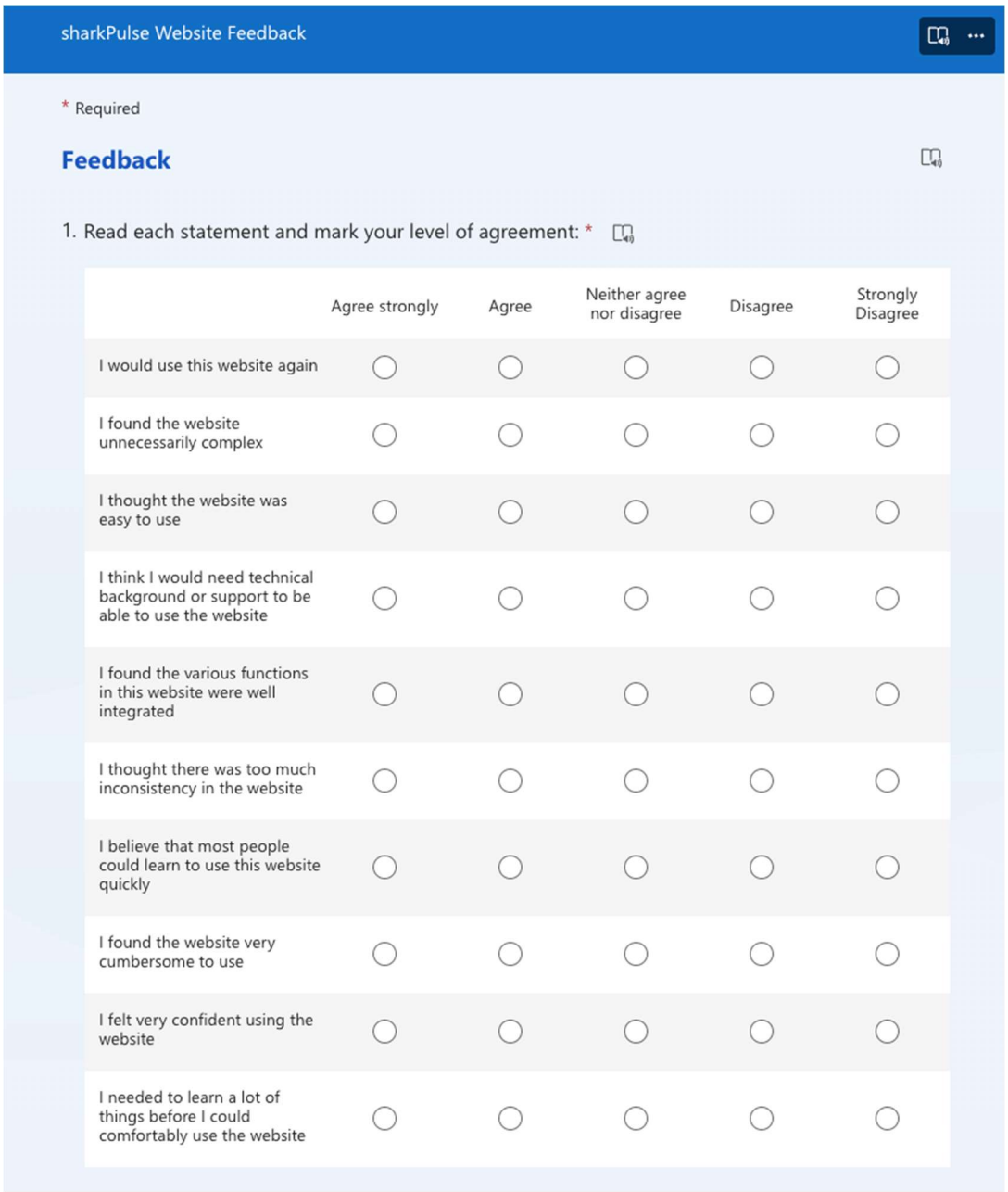


Figure 8: Top 5 Species Visualization Layout Options Proposed to Clients

The team developed an anonymous survey that the clients can share to collect feedback on the sharkPulse website [1] even after the completion of this semester. The survey is a Microsoft Form since this is the platform that Virginia Tech officially uses and was designed to take less than ten minutes to complete. The survey includes an explanation of sharkPulse's background. The questions were strategically developed based on best practices for user

experience surveys such as keeping the questionnaire concise and focused, asking open-ended questions, and mixing up question types. The first question shown in Figure 9 is modeled off the System Usability Scale (SUS), which is a standardized 10-item questionnaire presented in a Likert scale format where participants can indicate how strongly they agree with statements, and questions are phrased to alternate between positive and negative statements [11]



sharkPulse Website Feedback

* Required

Feedback

1. Read each statement and mark your level of agreement: *

	Agree strongly	Agree	Neither agree nor disagree	Disagree	Strongly Disagree
I would use this website again	○	○	○	○	○
I found the website unnecessarily complex	○	○	○	○	○
I thought the website was easy to use	○	○	○	○	○
I think I would need technical background or support to be able to use the website	○	○	○	○	○
I found the various functions in this website were well integrated	○	○	○	○	○
I thought there was too much inconsistency in the website	○	○	○	○	○
I believe that most people could learn to use this website quickly	○	○	○	○	○
I found the website very cumbersome to use	○	○	○	○	○
I felt very confident using the website	○	○	○	○	○
I needed to learn a lot of things before I could comfortably use the website	○	○	○	○	○

Figure 9: Website Feedback Form Preview

The clients reviewed the survey questions, and the team sent the survey out to peers and individuals of various ages and backgrounds. The purpose was to gather initial pilot responses from general users that could be summarized for the clients to gain insight into potential areas of improvement for the website's design. The results in Figure 10 are users' feedback on the usability and visual experience, which was primarily positive. Users did not indicate feeling that the website was overly technical or complex, which satisfies the clients' goals to create content that is accessible to users without technical background in this subject area.

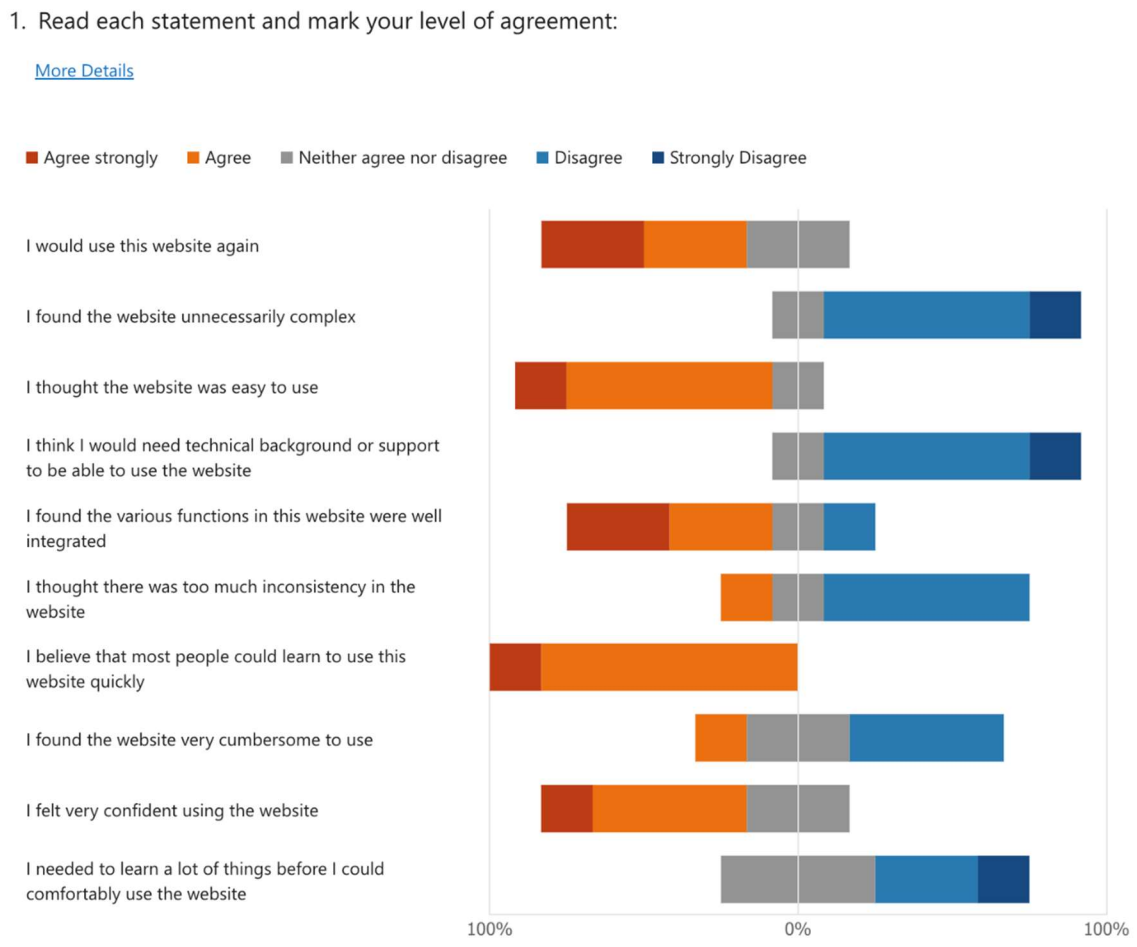


Figure 10: Pilot Survey Question 1 Responses

In the additional survey fields, users could add questions, concerns, or suggestions on navigation, visuals, and areas for improvement. These results are summarized in Table 1 and will be used to help the clients continue improving the website.

Question	**Responses**
Navigation	• Pulse monitor load time is very slow and seemed like a blank page, so someone needs to know to wait for it to load. • Navigating through the different pages felt easy. • Awkward on mobile device.
Visuals	• Clashing colors affected readability. • Liked the pictures, graphs, and figures. • Inconsistent home page styling with some sections having no margins on the side and having a different colored background from the rest of the webpages.
General Improvement	• Improve contrast and functionality. • Reduce the number of words on some of the home page slider for readability purposes. • Bolder breaks between sections on homepage. • Add some intro for Submit your Picture page. • Match menu icon list to navigation bar on top.

	• It seems like the links in the top-right menu are sorted in alphabetical order; Consider sorting them by category, and making the font much smaller so it fits on one screen without scrolling. • Add margins to parts of the home page and put a small caption below all images so users can easily see what they are about before reading the longer explanation. • Consider making the big images at the top of the home page shorter in height so that they fit on one screen without having to scroll, as this makes it easier to read them and quickly move to the next image

Table 1: Summary of Pilot Survey Responses

Additionally, the clients can continue to adjust questions based on the information they wish to collect and share the survey with their networks, around Virginia Tech's campus, and through other outlets to gather feedback.

6. User's Manual

All the visualization deliverables and styling updates produced by the team during the duration of this project can be seen on the sharkPulse website's home page at https://sp2.cs.vt.edu/ [1]. Figures 11 and 12 depict examples of the current home page updates implemented. Most of the team's work was directed at the general user visiting the website, which is why the bulk of updates were located on the home page.

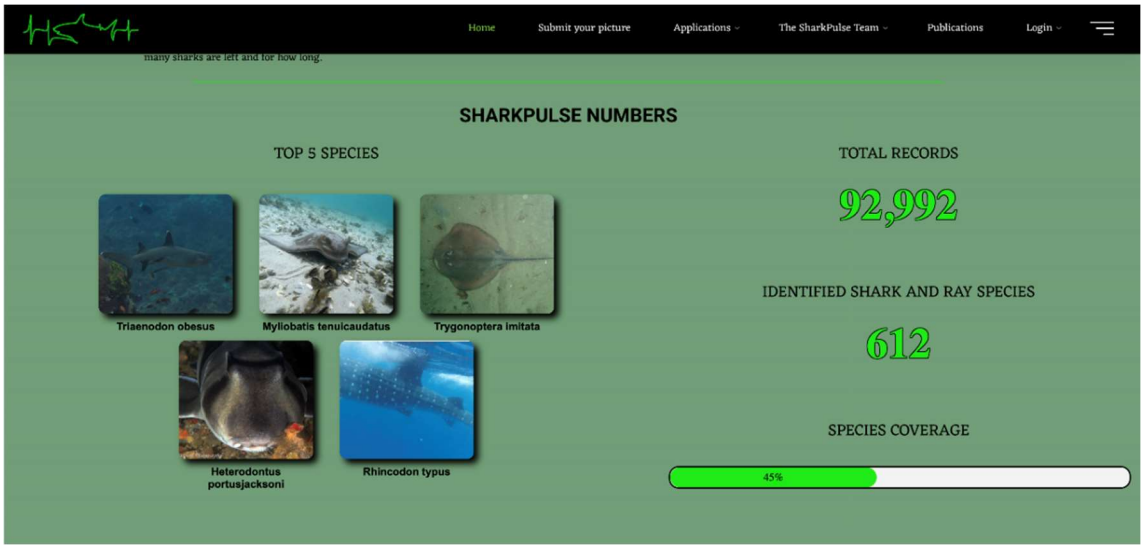


Figure 11: Dynamic sharkPulse Numbers Section in Home Page

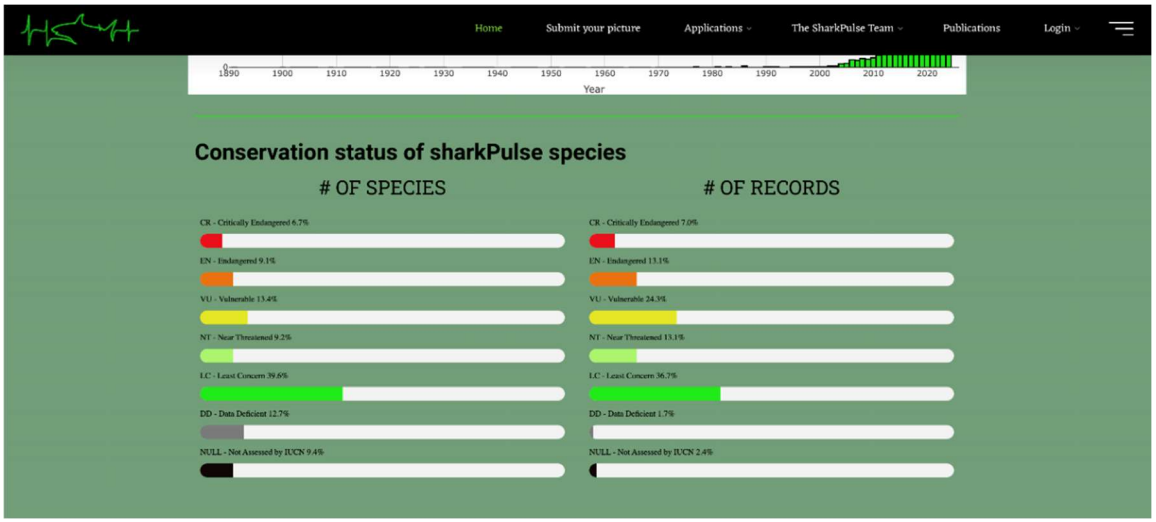


Figure 12: Dynamic Species Conservation Status Bars

Citizen scientists who may wish to submit pictures of shark sightings to the sharkPulse database can go to https://sp2.cs.vt.edu/the-app/ as seen in Figure 13.

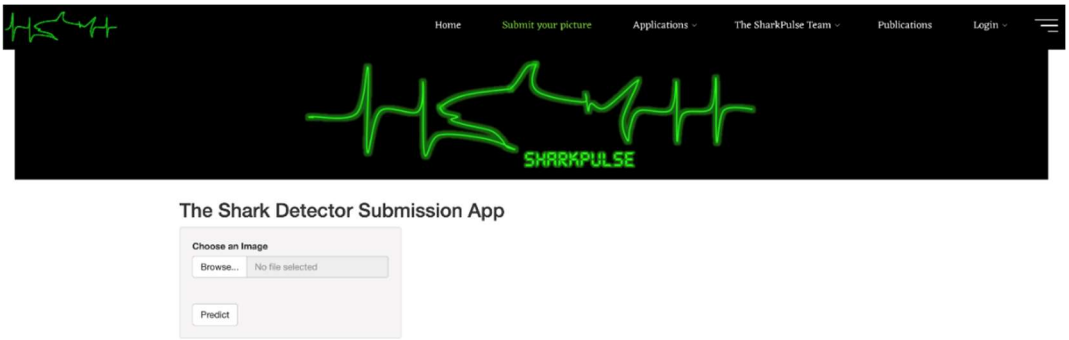


Figure 13: Picture Submission Page

They can explore the Pulse Monitor in Figure 14 at https://sp2.cs.vt.edu/pulse-monitor/ and the Validation Monitor to validate sightings in Figure 15 at https://sp2.cs.vt.edu/validation-monitor/.

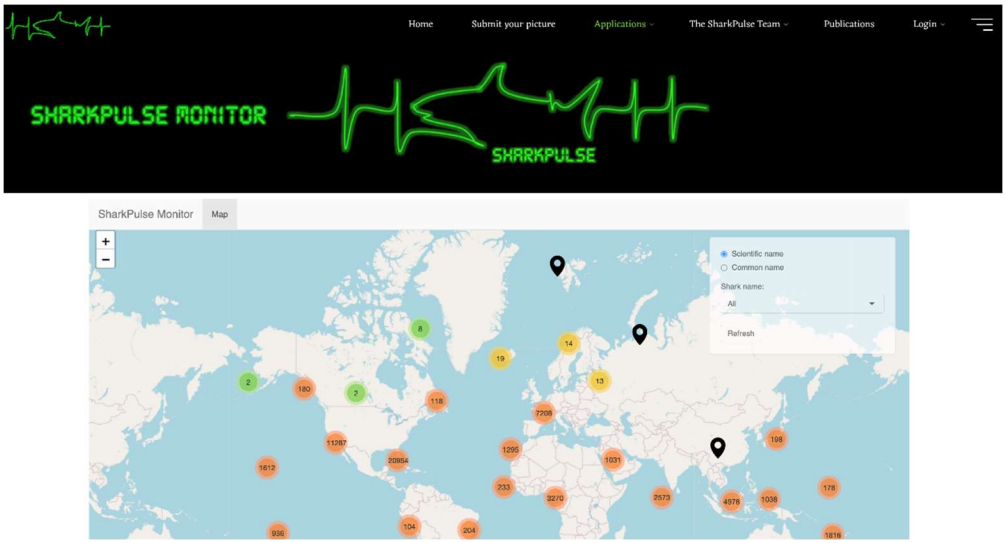


Figure 14: Pulse Monitor Page

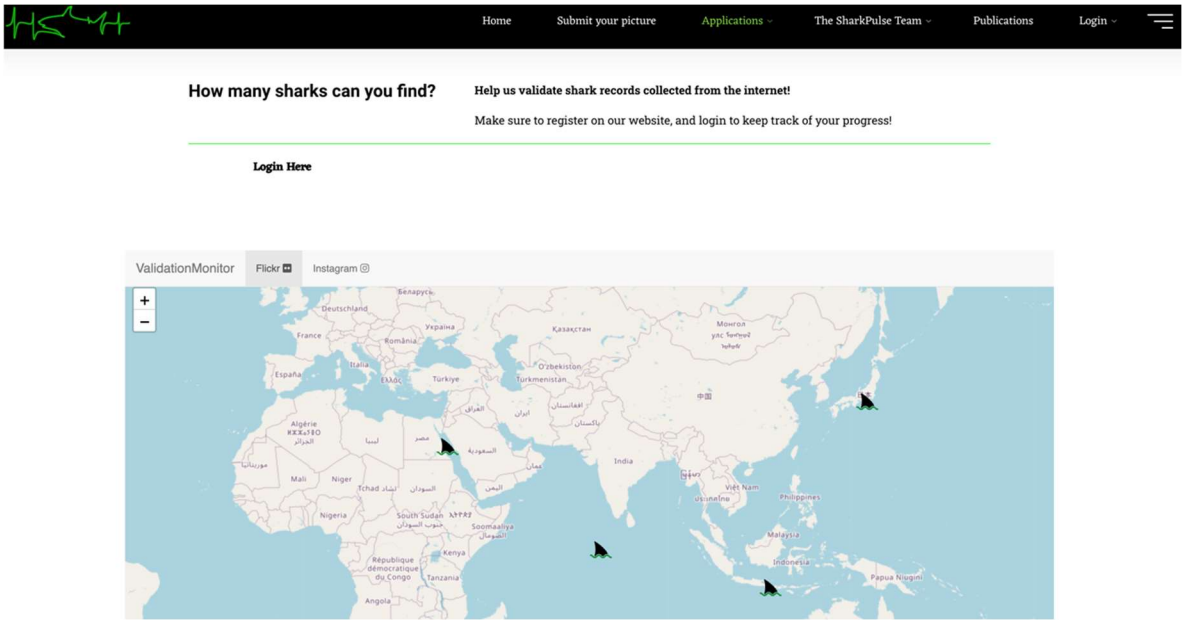
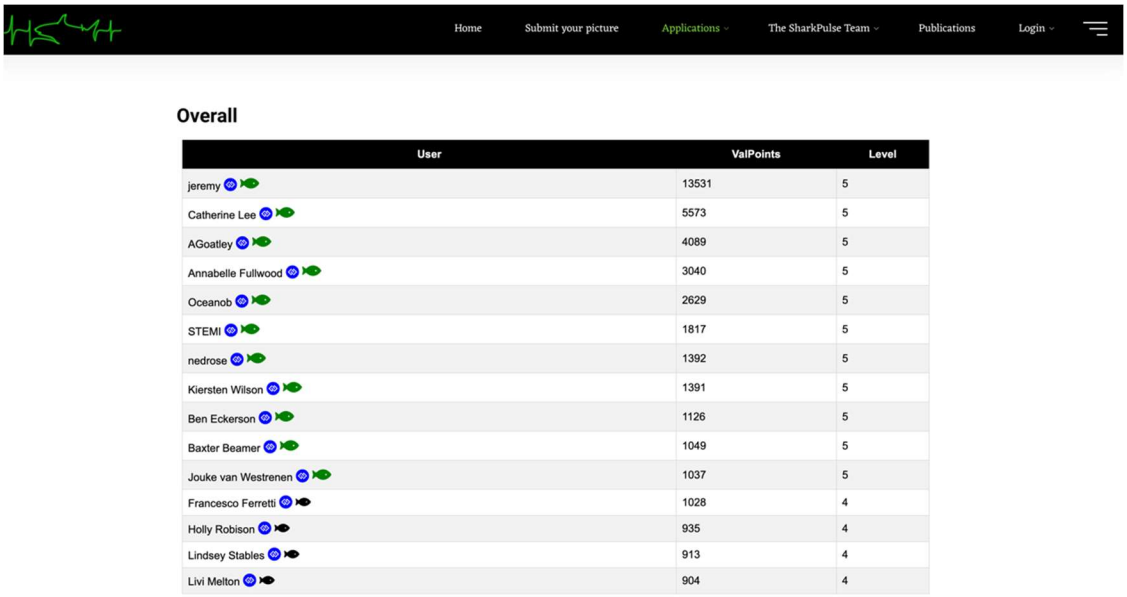


Figure 15:Validation Monitor Page

Any user who is interested in checking the progress of validating the sharkPulse records can explore the Validation Leaderboard in Figure 16 at https://sp2.cs.vt.edu/leaderboard/.



User	ValPoints	Level
jeremy	13531	5
Catherine Lee	5573	5
AGoatley	4089	5
Annabelle Fullwood	3040	5
Oceanob	2629	5
STEMI	1817	5
nedrose	1392	5
Kiersten Wilson	1391	5
Ben Eckerson	1126	5
Baxter Beamer	1049	5
Jouke van Westrenen	1037	5
Francesco Ferretti	1028	4
Holly Robison	935	4
Lindsey Stables	913	4
Livi Melton	904	4

Figure 16: Validation Leaderboard Page

Citizen scientists or general users who want to learn more or find instructions for classifying shark species can use the Identification Guide in Figure 17 at https://sp2.cs.vt.edu/identification-guide/.

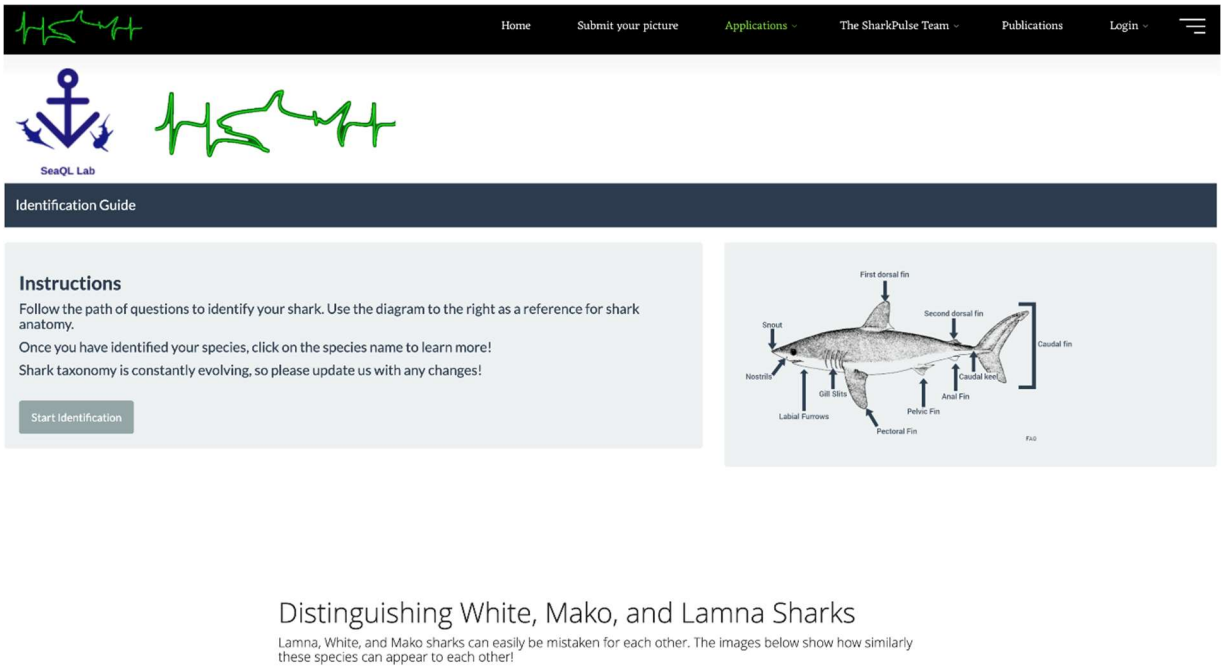


Figure 17: Identification Guide Page

On the Shark Detector AI page as shown in Figure 18, authenticated scientists can enter login information.

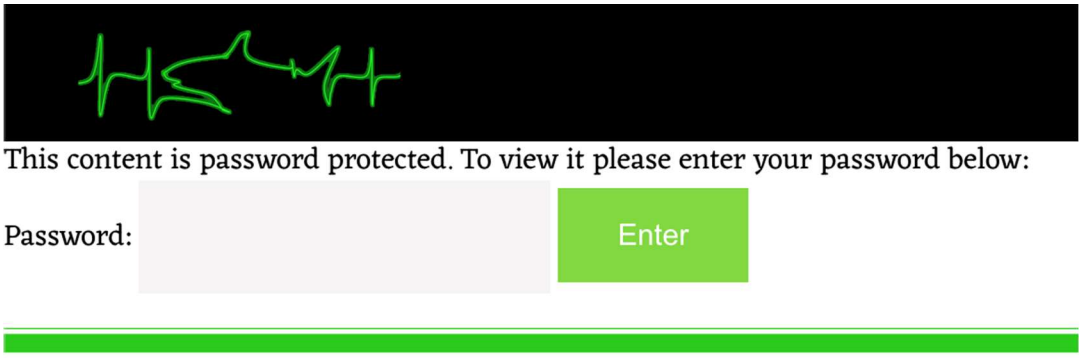


Figure 18: Shark Detector AI Login Page

Finally, for information about the sharkPulse network and research, any user can visit the sharkPulse Team page in Figure 19 at https://sp2.cs.vt.edu/network-2/ or Publications page in Figure 20 at https://sp2.cs.vt.edu/publications-2/ [1].

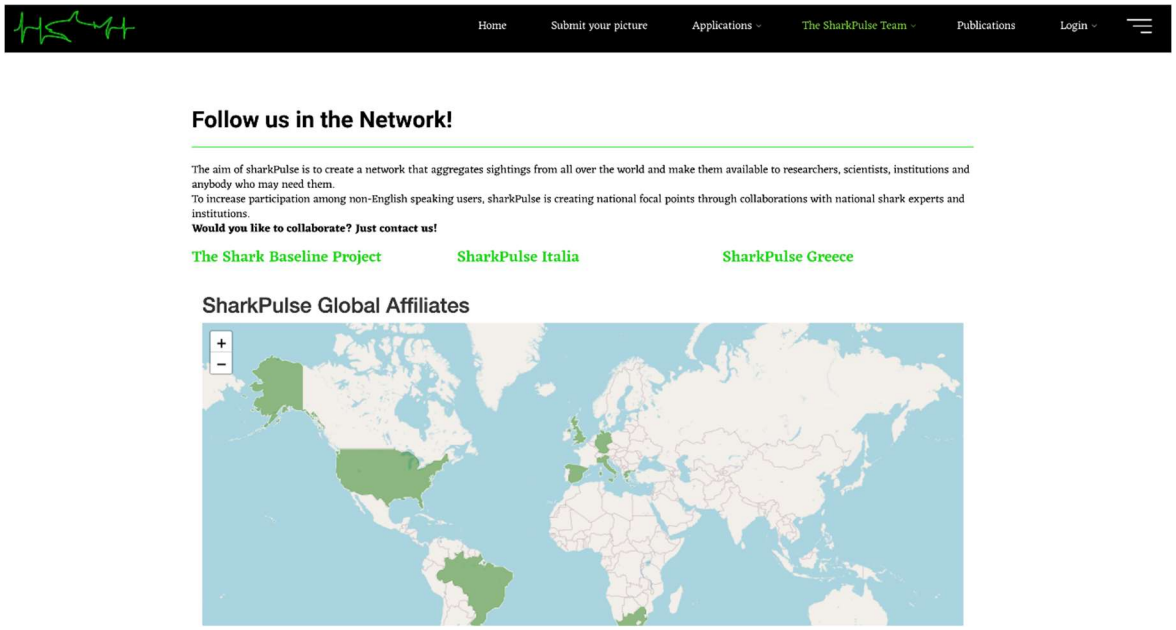


Figure 19: sharkPulse Team/Network Page

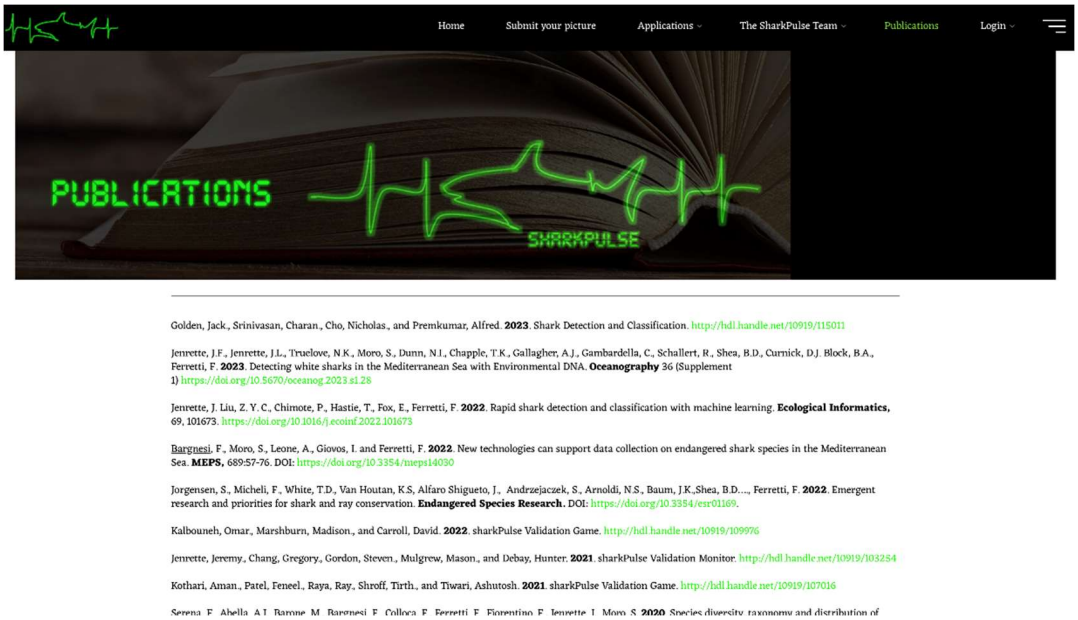


Figure 20: Publications Page

7. Developer's Manual

7.1 Dynamic Scripting

File	Description
DynamicVisuals.js	JavaScript that contains all the intermediate visualizations
queryConservation.php	Generates a percentage breakdown of each conservation classification in the database and its respective frequency on a per species basis
queryConservationRecords.php	Generates a percentage breakdown of each conservation classification in the database and its respective frequency on a per record basis
topfive.php	Generates a list of the top five species in the database by record and sources a representative image from the file system for each species
species_coverage.php	Generates a progress bar indicating the percentage of shark species that are represented within the sharkPulse database
app.R	Contains the UI and server side code for the global affiliate map. Dynamically sources country/social network data and generates country borders by pulling GeoJSON information from the rnaturalearth library.
queryTaxonomyAcc.php	Generates a table of bolded genus entries and italicized species sub-entries. Entries that have a valid F1-score (predictive accuracy mdoel) in the back end file system are displayed along with image counts. Clicking on the bolded genuses creates a drop down of the corresponding species.

Table 2: Scripts and Descriptions

7.2 Server Environment

The team connected to the sharkPulse server using the Remote SSH VSCode extension with the credentials provided by the clients and Virginia Tech's SSL VPN, which allows access to Virginia Tech remotely across the globe. The installation process for the VPN can be found for both Windows [12] and Mac [13] on the VT4Help site. Figure 20 breaks down the file system structure of the sharkPulse server environment. The server's file system was large and complex with directories to supplement different components of the site. Most of the work done by the team was within the /var/www/html/sharkPulse/dynamic directory which is the directory pointed to by the site to host the PHP [6] scripts that are sourced to generate the visualizations. Further work involving the image retrieval for the scripts and the network map was done in the /home/spr and /srv/shiny-server directories, respectively. The images were stored within two sub-directories, namely /home/spr/instaval_images/shark and /home/spr/sp_images/www. The filenames in the database correlate directly to the image filenames in these directories without the file extensions. This resulted in the team creating a script to append the proper file extensions into the database. The /srv/shiny-server directory hosts all of the site's R-Shiny [5] apps and contains the UI and server side code for the global affiliate map. The break down for the file system structure is indicated in Figure 21 and the list of scripts and their respective functionality is detailed in Table 2.

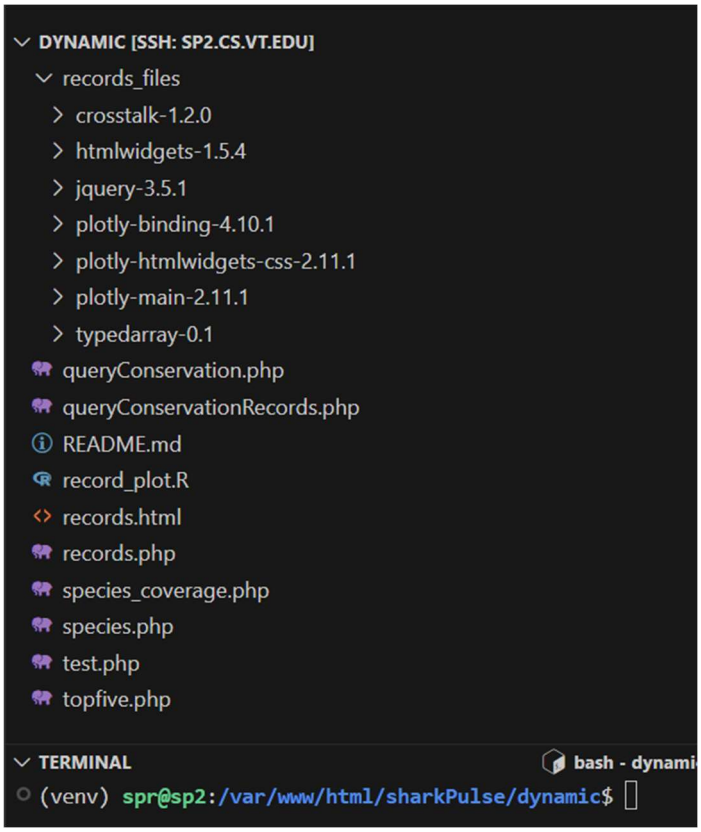


Figure 21: Main Directory File System

7.3 Scripting

Every listed PHP [6] script has a correlated intermediate JavaScript [9] script that is stored in DynamicVisuals.js. This condensed script asynchronously queried the database for all the necessary information and generated multiple HTML [10] files that are then stored in an image directory. These files can be opened directly through the team's file explorer or browser to display the contents. After getting accustomed to PHP [6], the translation from JavaScript [9] was relatively simple and the concepts carried over throughout the scripts. The installation of a local web server was unnecessary since intermediate changes could be tested locally and final edits to the site could be made using the website's web server by sourcing the final versions of the PHP [6] scripts. The pros and cons of each scripting language meant that the most optimal development approach involved using both. The details of each language are listed in Table 3.

	JavaScript	PHP
Database	NPM pg library creates client object that connects to database	Built in PDO framework [14]; executes SQL queries directly (automatic database support)
HTML	Writes to HTML file using JavaScript File System library	Returns physical HTML object
Web Server	Not Required	Required
Site Compatibility	Requires cron jobs to run scripts and source outputted visuals	Elementor module sources script directly for visuals

Table 3: Languages Used

7.3 Database Management

The underlying database for the website is a PostgreSQL [8] database with nearly 70 tables. The portion of the site that our team worked directly on utilized mainly two of these tables: sharkPulse and taxonomy3. The former contained information on all user-submitted and data-mined shark sighting records from the site, while the latter contained information on each unique taxonomical shark/ray species as detailed by the IUCN (International Union for Conservation of Nature) [15]. The scripts queried all relevant information from one of these two tables, or a SQL join of the two. Due to the size and complexity of the database, the team utilized pgAdmin4 [16], an open-source database management platform (https://www.pgadmin.org/download/pgadmin-4-windows). Figure 22 details the structure of the sharkPulse database that our team primarily worked within.

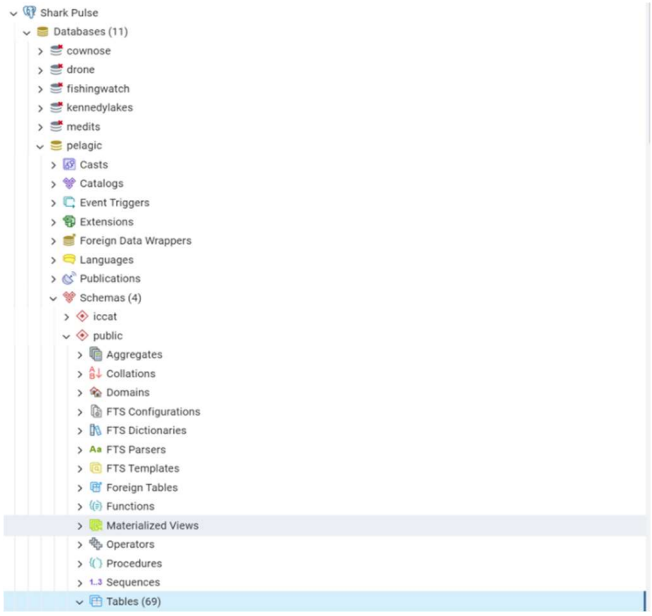


Figure 22: sharkPulse Database Structure

7.4 Species Coverage Script

The species coverage script queries the number of unique species name within the shark pulse table against the number of total species within the taxonomy3 table to calculate a representative percentage of shark species that are recorded within the site. The HTML [10] template for the visual is a simple animated green progress bar. This style of progress bar is used throughout some of the other scripts as well. Figure 23 is an example query for the Species Coverage visual which calculates the percentage of distinct shark species that the sharkPulse database has coverage of. Figure 24 is the dynamic visualization generated from this script.

```
async function queryCoverage() {
    const sql = "SELECT ROUND(CASE WHEN (SELECT COUNT(DISTINCT species_name) \
    FROM sharkpulse) > (SELECT COUNT(DISTINCT species_name) FROM taxonomy3) \
    THEN (CAST((SELECT COUNT(DISTINCT species_name) FROM taxonomy3) AS DECIMAL) \
    / (SELECT COUNT(DISTINCT species_name) FROM sharkpulse)) * 100 ELSE \
    (CAST((SELECT COUNT(DISTINCT species_name) FROM sharkpulse) AS DECIMAL) / \
    (SELECT COUNT(DISTINCT species_name) FROM taxonomy3)) * 100 END) AS rounded_percentage;"
    const res = await client.query(sql);
    return Number(res.rows[0].rounded_percentage).toLocaleString();
}
```

Figure 23: Species Coverage Query

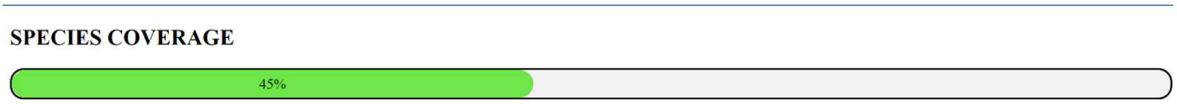


Figure 24: Species Coverage Visual

7.5 Conservation Status Script

The conservation status script generates a breakdown of the percentage of each conservation classification value. Each species in the taxonomy3 table has one of six possible classification values associated with it. There are two scripts: one that calculates the percentages on a per-species basis, and one that calculates the percentages on a per-record basis. The per-species script queries the database for the counts of each classification status and divides it by the total number of species to generate a percentage for each value. The per-record script is a bit more complicated as the records in the sharkPulse table have no classification column. In this script, the two tables are joined in such a way that each record in the sharkPulse table is mapped to a record in the taxonomy3 table. This joint table is then queried similarly to calculate the percentage values. The HTML [10] template for these scripts utilizes the progress bars from the species coverage script but tweaks them slightly for formatting purposes. Figure 25 is a code snippet of the script that queries the conservation status percentages and Figure 26 is the generated visualization.

```php
$cons_statuses = ['CR', 'EN', 'VU', 'NT', 'LC', 'DD', 'NULL'];
$cons_def = ['Critically Endangered', 'Endangered', 'Vulnerable', 'Near Threatened', 'Least Concern', 'Data Deficient', 'Not Assessed by IUCN'];
$cons_colors = ['#E91010', '#E97310', '#E6E722', '#ABF66E', '#21E91B', '#7A7A7A', '#100707'];

function queryConservation($cons_statuses, $dbconn) {
    $total_sql = "SELECT COUNT(*) FROM sharkpulse";
    $total_res = pg_query($dbconn, $total_sql);
    $total_count = (int) pg_fetch_result($total_res, 0);

    $cons_percentages = [];
    foreach ($cons_statuses as $status) {
        if ($status == 'NULL'){
            $conservation_sql = "SELECT COUNT(*) AS null_conservation_count
            FROM taxonomy3
            INNER JOIN sharkpulse ON sharkpulse.species_name = taxonomy3.species_name
            WHERE taxonomy3.category IS NULL;";
        } else {
            $conservation_sql = "SELECT COUNT(*) AS count
            FROM sharkpulse
            INNER JOIN taxonomy3 ON sharkpulse.species_name = taxonomy3.species_name
            WHERE category = '$status'";
        }

        $conservation_res = pg_query($dbconn, $conservation_sql);
        $conservation_count = (int) pg_fetch_result($conservation_res, 0);
        $cons_percentages[] = number_format(($conservation_count / $total_count) * 100, 1);
    }

    return $cons_percentages;
}
```

Figure 25: Conservation Status Records Query

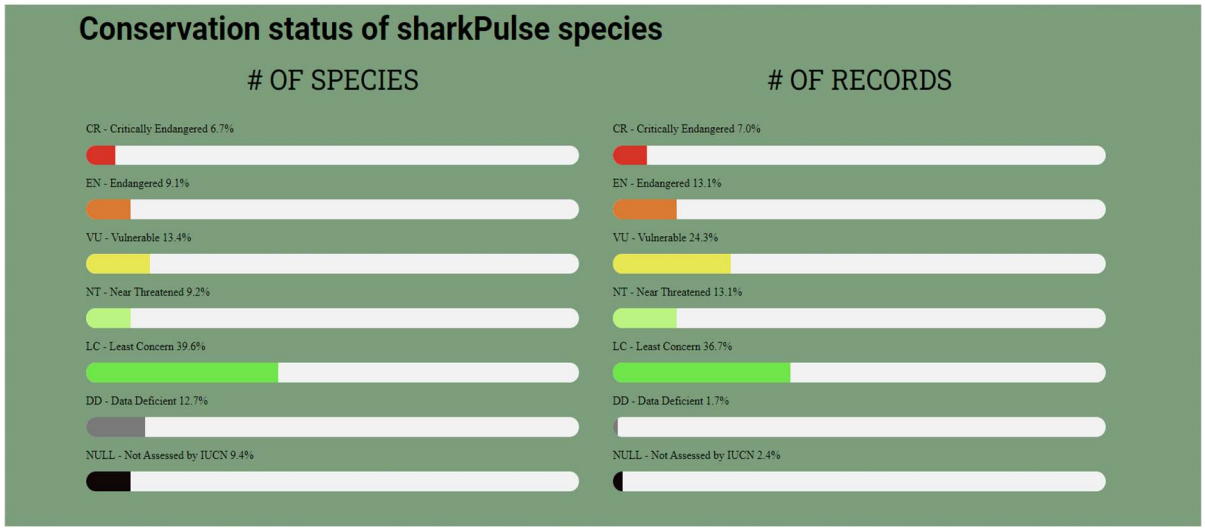


Figure 26: Conservation Status Visual

7.6 Top 5 Species Script

The top five species script was the largest and most complex requirement for the project. The query involved retrieving only the top five shark species and accompanying visuals. Each

record in the database contains a filename for the corresponding image for the record. Initially, the query covered all records in the database, but after receiving client feedback, the script was adjusted to ignore ray species records.

There was much difficulty involved in obtaining high quality representative images for these species. The filenames did not contain file extensions and the images were separated into two sub-directories in the server environment (the sub-directory a file was in could not be deduced from the database). Furthermore, the resolutions of the images were vastly different, which necessitated a check for a specific resolution range within the query as well as a hard limit on the image size specified within the HTML [10] template code. In the future, the sharkPulse team can address this issue by creating a separate database table with standardized representative images on a per-species basis that has been discussed with the clients. This task has become a long-term goal for improving the website. Figure 27 is the SQL [8] query that retrieves the top five species by record count and their respective images filenames and Figure 28 is the generated visualization.

```
// Query the top five species data
try {
    $sql = "
    SELECT species_name, img_name
    FROM (
        SELECT species_name, img_name,
               ROW_NUMBER() OVER(PARTITION BY species_name ORDER BY img_name) AS row_num
        FROM sharkpulse
        WHERE img_name != 'no image found!'
    ) AS subquery
    WHERE row_num = 26
    AND species_name IN (
        SELECT species_name
        FROM (
            SELECT species_name, COUNT(*) AS species_count
            FROM sharkpulse
            WHERE img_name != 'no image found!'
            GROUP BY species_name
            ORDER BY species_count DESC
            LIMIT 5
        ) AS top_species
    )
    ORDER BY (SELECT COUNT(*) FROM sharkpulse WHERE species_name = subquery.species_name) DESC;";
    $stmt = $pdo->query($sql);
    $speciesData = $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    die("Error querying top five species: " . $e->getMessage());
}
```

Figure 27: Top 5 Species Query File

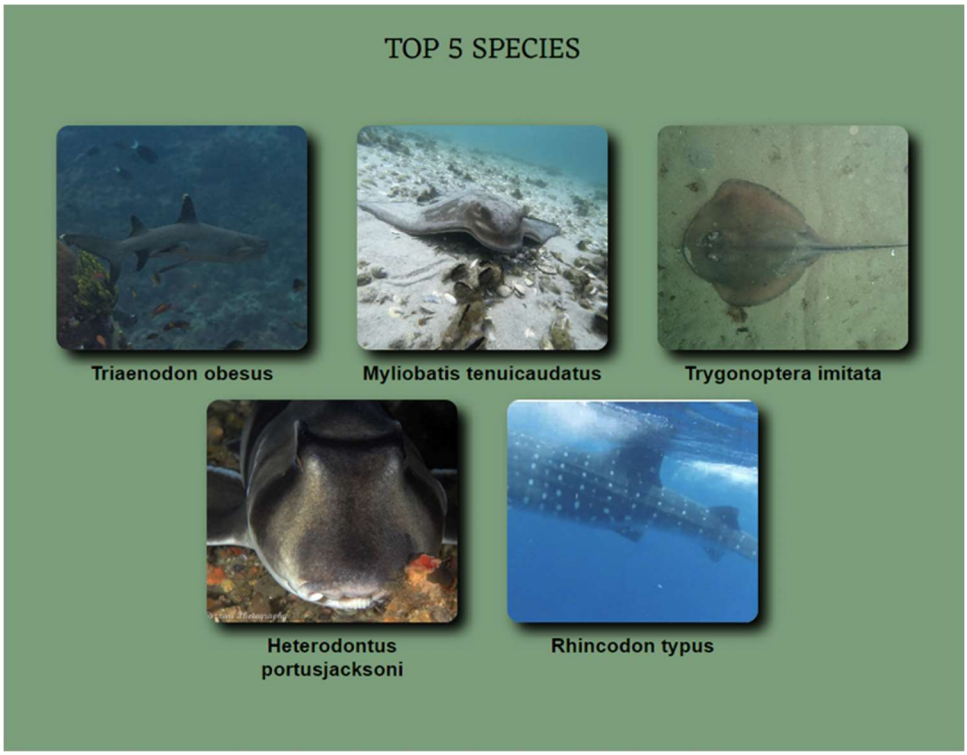


Figure 28: Final Top 5 Species Visual

7.7 Global Affiliate Map Script

The Global Affiliate Map is a Leaflet [17] map that sources sharkPulse social network data from an Excel spreadsheet in the back end file system. The spreadsheet contains information on the different countries that are affiliated with the sharkPulse team and their corresponding social network sites. The app dynamically retrieves GeoJSON data using the rnaturalearth [18] library to shade in the country borders and vector applies a popup function across the spatial coordinates for each country. The popups contain hyperlinks to relevant social media sites for each respective country. Figure 29 displays the source code for the Global Affiliate Map and Figure 30 shows the generated visualization.

```r
library(shiny)
library(leaflet)
library(readxl)
library(rnaturalearth)

# Define UI
ui <- fluidPage(
  titlePanel("SharkPulse Global Affiliates"),

  # Leaflet map output
  leafletOutput("map", width = "100%", height = "500px")
)

# Define server logic
server <- function(input, output, session) {

  # Read Excel file and extract relevant data
  social_media_data <- read_excel(".xlsx", col_types = c("text", "text", "text"))

  # Create links based on social network and handle
  social_media_data$link <- paste0("https://www.", tolower(social_media_data$SN), ".com/", social_media_data$handle)

  # Create map of countries to links
  country_links <- tapply(social_media_data$link, social_media_data$FocalPoint, function(x) paste(x, collapse = "<br>"))

  # Load country polygons using rnaturalearth
  world <- ne_countries(scale = "medium", returnclass = "sf", datum = "WGS84")

# Check if the warning persists


  # Create leaflet map
  output$map <- renderLeaflet({
    leaflet(world) %>%
      addTiles() %>%
      setView(lng = 0, lat = 30, zoom = 2) %>%
      addPolygons(
        fillColor = "green",
        fillOpacity = 0.5,
        color = "white",
        stroke = TRUE,
        weight = 1,
        popup = lapply(names(country_links), function(country) {
          paste("<b>", country, "</b><br>", country_links[[country]])
        })
      )
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

Figure 29: Global Affiliate Map App

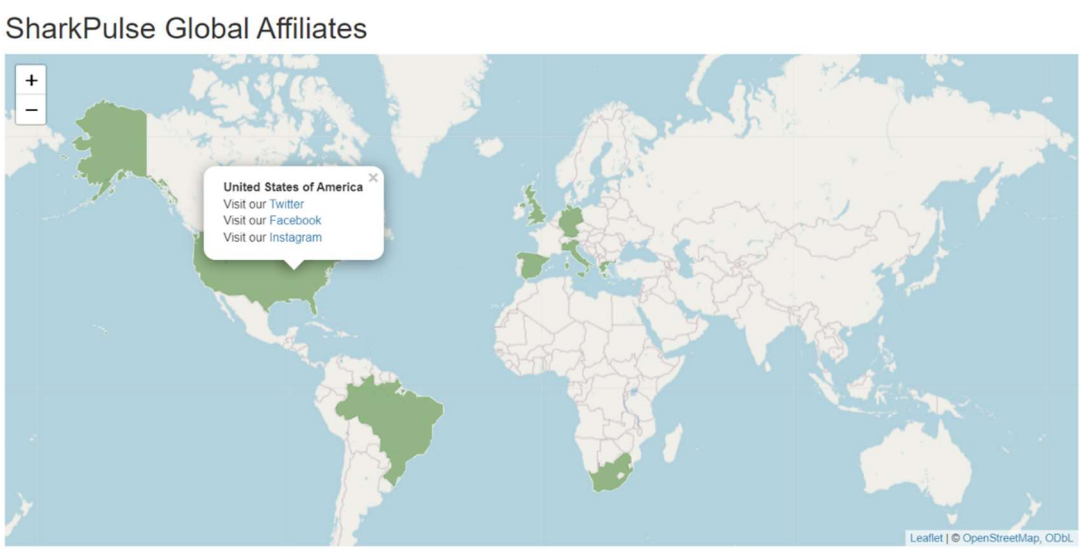


Figure 30: Global Affiliate Map Visual

7.8 Taxonomic Accuracy Table Script

The Taxonomic Accuracy Table script retrieves pre-existing predictive model accuracy data from the back end file system and displays the values in a table on a per genus and species basis. The script queries every record in the training copy table which contains all the submitted records for the Shark Detector AI. The script then iterates through the file system directories that contain text files with breakdowns of the predictive model runs and extracts the F-1 score for all species with an existing classification report. This score is indicative of the machine learning models predictive accuracy for that specific species/genus. These values are then matched up with the species/genus names and record counts and displayed within an HTML [10] table. Figure 31 displays a code snippet for the species accuracy score extraction and Figure 32 is the generated visualization.

```
function parseSpeciesAccuracies($filePath) {
    $report = file_get_contents($filePath);
    $lines = explode("\n", $report);
    $accuracies = [];
    foreach ($lines as $line) {
        $line = trim($line);
        if (strpos($line, '_') !== false) { // Check if line contains a species name
            $parts = preg_split('/\s{2,}/', $line);
            $name = $parts[0];
            $f1Score = $parts[3];
            $speciesName = substr($name, strpos($name, '_') + 1); // Extract species name (remove genus)
            $accuracies[$speciesName] = (float) $f1Score; // Parse accuracy value
        }
    }
    return $accuracies;
}
```

Figure 31: Taxonomic Accuracy Table Accuracy Extraction

Species	Images(Acc)
Acroteriobatus	213 (0)
Aculeola	15 (0)
Aetobatus	1343 (0)
Aetomylaeus	73 (0)
Alopias	945 (0.94)
Amblyraja	61 (0)
Apristurus	101 (0)
Aptychotrema	150 (0)
Asymbolus	77 (0)
Atelomycterus	108 (0)
Atlantoraja	4 (0)
Aulohalaelurus	39 (0)
Bathyraja	29 (0)
Bathytoshia	792 (0)
Beringraja	235 (0)
Brachaelurus	960 (0)
Brevitrygon	27 (0)
Bythaelurus	11 (0)
Callorhinchus	3 (0)
Carcharhinus	16852 (0.91)
Carcharias	4099 (0.79)
Carcharodan	1 (0)
Carcharodon	3345 (0.72)
Centrophorus	66 (0)
Centroscyllium	36 (0)
Centroscymnus	25 (0)
Centroselachus	4 (0)
Cephaloscyllium	1500 (0)
Cetorhinus	892 (0)
Chaenogaleus	7 (0)
Chiloscyllium	767 (0)
Chimaera	4 (0)
Chlamydoselachus	22 (0)
Cirrhigaleus	12 (0)
Dalatias	84 (0)

Species	Images(Acc)
Gymnura	381 (0)
Halaelurus	9 (0)
Haploblepharus	1781 (0)
Harriotta	1 (0)
Hemigaleus	17 (0)
Hemipristis	58 (0)
Hemiscyllium	560 (0)
Hemitriakis	10 (0)
Hemitrygon	84 (0)
Heptranchias	40 (0)
Heterodontus	5758 (0.98)
H. francisci	*1215 (0.79)*
H. galeatus	*884 (0.82)*
H. japonicus	*314 (0.97)*
H. mexicanus	*19 (0)*
H. portusjacksoni	*3258 (0.88)*
H. quoyi	*58 (0)*
H. zebra	*10 (0)*
Hexanchus	985 (0)
Hexatrygon	1 (0)
Himantura	140 (0)
Hydrolagus	391 (0)
Hypanus	1636 (0)
Hypnos	244 (0)
Hypogaleus	5 (0)
Iago	47 (0)
Isistius	101 (0)
Isurus	2965 (0.88)
Lamiopsis	4 (0)
Lamna	487 (0)
Leptocharias	1 (0)
Leucoraja	125 (0)
Loxodon	45 (0)
Maculabatis	27 (0)
Malacoraja	14 (0)
Megachasma	354 (0)
Megatrygon	5 (0)

Species	Images(Acc)
Plesiotrygon	3 (0)
Pliotrema	1 (0)
Poroderma	941 (0)
Potamotrygon	70 (0)
Prionace	4170 (0.7)
Pristiophorus	21 (0)
Pristis	50 (0)
Proscyllium	1 (0)
Psammobatis	5 (0)
Pseudobatos	281 (0)
Pseudocarcharias	10 (0)
Pseudoginglymostoma	15 (0)
Pseudotriakis	2 (0)
Pteroplatytrygon	36 (0)
Raja	684 (0)
Rajella	4 (0)
Rhincodon	5165 (0.93)
Rhinobatos	34 (0)
Rhinoptera	440 (0)
Rhizoprionodon	1566 (0)
Rhynchobatus	203 (0)
Rioraja	3 (0)
Rostroraja	337 (0)
Schroederichthys	160 (0)
Scoliodon	88 (0)
Scyliorhinus	3617 (0.99)
Scymnodon	9 (0)
Somniosus	64 (0)
Sphyrna	5036 (0.89)
Spiniraja	16 (0)
Squaliolus	5 (0)
Squalus	1237 (0)
Squatina	1427 (0)
Stegostoma	964 (0)
Styracura	175 (0)

Figure 32: Taxonomic Accuracy Table Visual

7.9 Database Connection Script

This script creates a connection to a PostgreSQL [8] database and executes SQL [8] queries to the database. Specifically, it is trying to access the database that contains all the images from the Instagram data scraping. The code then proceeds to copy each image from the source directory to a destination directory. If the source files exist, the image gets copied to the destination directory and ensures that the database connection is closed. Figure 33 is the Database Connection Script.

```python

# Set up database connection
try:
    connection = psycopg2.connect(
        database="pelagic",
        user="spr",
        password="spr_pass",
        host="localhost",
        port="5432"
    )
    cursor = connection.cursor()

    # Execute query to get the names of images to be copied
    query = """
    SELECT img_name FROM instagram
    WHERE date > '2023-12-23'
    AND validated = 't'
    AND shark = 'shark'
    AND repost IS NOT NULL;
    """
    cursor.execute(query)
    img_names = [row[0] for row in cursor.fetchall()]

    # Paths for source and destination directories
    source_dir = '/home/spr/instaval_images/shark'
    dest_dir = '/home/spr/sp_images/www'

    # Copy each image from the source to the destination directory
    for img_name in img_names:
        source_path = os.path.join(source_dir, full_img_name)
        dest_path = os.path.join(dest_dir, full_img_name)
        if os.path.exists(source_path):  # Check if the source file exists before copying
            shutil.copy(source_path, dest_path)
            print(f"Copied {full_img_name} to {dest_dir}")
        else:
            print(f"File {full_img_name} does not exist in {source_dir}")

except (Exception, psycopg2.DatabaseError) as error:
    print("Error while accessing database or copying files:", error)
finally:
    if connection is not None:
        cursor.close()
        connection.close()
        print("Database connection closed.")
```

Figure 33: Database Connection Script

8. Lessons Learned

8.1 Timeline/Schedule

Dates	Milestones
February 16-29	- Complete Interim Report (Complete) - Establish access and familiarity with existing files and GitHub (Complete) - Identify strategies to improve the website's user experience (Complete) - Identify where dynamic features and visualizations can be implemented and integrated into the website (Complete)
February 29-March 15	- Complete Presentation 2 (Complete) - Implement and troubleshoot initial proposed improvements to the website's dynamic visualizations (Complete)
March 15-March 31	- Test and evaluate implemented website changes (Complete) - Implement feedback and adjustments based on testing results (Complete) - Optimize efficiency of the website changes (Complete) - Consider additional ideas for data visualizations or dynamic improvements (Complete)

March 31-April 12	- Test and evaluate implemented website changes (Complete) - Implement feedback and adjustments based on testing results (Complete) - Develop survey for clients to collect website feedback (Complete)
April 12-April 26	- Conduct final tests and evaluations and gather pilot survey responses (Complete) - Implement final feedback and changes from testing and client (Complete) - Compile project documentation and assessments of the work completed (Complete)
April 26-May 8	- Finalize and review project documentation including User and Developer Manuals (Complete) - Complete Final Presentation (Complete) - Complete Final Report (Complete) - Complete VTechWorks submission (Complete)

Table 4: Timeline of Work Completed

Table 4 lays out the team's timeline in the first column and milestones with status in the second column. The team broke down the schedule into chunks lasting about 2 weeks that would end on Fridays to match the members' work schedules and school breaks. The milestones cover the team's deliverables and the class submissions due.

8.2 Problems

The team has worked together to minimize the problems faced while trying to complete the goals the stakeholders had provided. One of the first problems faced was trying to connect to the website to make changes to it. The sharkPulse website [1] was set up in a way where one had to connect to it through the school's network to login to make changes. This caused problems as all the group members live off campus, making it inconvenient to work on the project. Another problem the group faced was that at the beginning the client was hard to communicate with because they were traveling at the time. For a while email was the only method of communication, which resulted in slower response times for everyone involved.

8.3 Solutions

Our solution to not being able to connect to the website without the school's network was utilizing VT's SSL VPN [4] which allows us to connect to the network remotely. With this, the team could freely make edits to the website from any location, drastically improving our workflow. The team was able to smooth over all communication problems by creating a Slack channel and setting up a Trello board to keep track of progress. The team set up weekly meetings to go over current work and project updates with the client and receive constructive feedback.

8.4 Future Work

In the future, the clients can share and refine the feedback survey and use the responses to guide future website improvements. It may be helpful to design a visually appealing, informative graphic that includes a QR code and physically post this flyer around campus buildings to reach a wider audience. The QR code can also be hyperlinked to the survey, and then the flyer can be saved as a PDF; this would enable the clients to share the flyer online by email for example and give people the option to either scan or click the QR code to access the survey. There are already ideas for improving the aesthetics of the home page such as re-evaluating the background images of sections and adjusting the formatting and styling of content to be more consistent. The website could also be designed to be more compatible with mobile devices. They have also introduced additional problem areas that can be addressed including the website's editability, gamification, and the content of other pages like the sharkPulse Team and picture submission pages.

The clients are interested in exploring alternatives to the Elementor module [3] to increase the editability and flexibility of the website. In the picture submission page, the clients want to make the process more dynamic and responsive so that the form is more streamlined and load times are faster. It would also be useful to add a cropping feature that can be utilized before the picture is submitted to the database. Other discussions for future work involved general UI changes, performance improvements, and database additions.

9. Acknowledgements

Name, Position	Contact Info
Franceso Ferretti, Assistant Professor, Department of Fish and Wildlife Conservation	ferretti@vt.edu
Jeremy Jenrette, Graduate Research Assistant	jjeremy1@vt.edu
Dr. Edward Fox, Professor, Department of Computer Science	fox@vt.edu
Satvik Chekuri, Graduate Teaching Assistant	satvikchekuri@vt.edu

Table 5: Acknowledgements

10. References

[1] sharkPulse Team. 2024. "sharkPulse," *sharkPulse*. https://sp2.cs.vt.edu/ (accessed May 04, 2024).

[2] WordPress Team. 2024. WordPress, "WordPress.com: Create a Free Website or Blog," *WordPress.com*. https://wordpress.com/ (accessed May 04, 2024).

[3] Elementor Team. 2024. "Create a Website - Elementor Website Builder for WordPress," *Elementor*. https://elementor.com/?utm_source=google&utm_medium=cpc&utm_campaign=6487935276&utm_term=elementor&gad_source=1&gclid=Cj0KCQjw0MexBhD3ARIsAEI3WHIoF2VQxjz_K9hmaJEAcyTUbuON1t0KEyCHXP_kH_g6WvhjXTRQ0j4aAoFfEALw_wcB (accessed May 04, 2024).

[4] Ivanti Team. 2024. Ivanti, "Secure Access Client | Ivanti," *www.ivanti.com*. https://www.ivanti.com/products/secure-unified-client (accessed May 04, 2024).

[5] RStudio Inc. 2024. "Shiny," *www.rstudio.com*. https://www.rstudio.com/products/shiny/ (accessed May 04, 2024).

[6] The PHP Group. 2024. "PHP: Hypertext Preprocessor," *Php.net*, Apr. 04, 2019. https://www.php.net/ (accessed May 04, 2024).

[7] OpenJS Foundation. 2024. Node.js, "Node.js," *Node.js*. https://nodejs.org/en (accessed May 04, 2024).

[8] The PostgreSQL Global Development Group. 2024. "PostgreSQL: the World's Most Advanced Open Source Database," *Postgresql.org*, 2019. https://www.postgresql.org/ (accessed May 04, 2024).

[9] MDN Contributors. 2024. "JavaScript," *MDN Web Docs*, Sep. 25, 2023. https://developer.mozilla.org/en-US/docs/Web/javascript (accessed May 04, 2024).

[10] MDN Contributors. 2024 "HTML: HyperText Markup Language," *MDN Web Docs*, Dec. 10, 2018. https://developer.mozilla.org/en-US/docs/Web/HTML (accessed May 04, 2024).

[11] K. Betteridge. 2024. "What Every Client Should Know about SUS Scores | Bentley University," *www.bentley.edu*. https://www.bentley.edu/centers/user-experience-center/what-every-client-should-know-about-sus-scores (accessed May 04, 2024).

[12] "Knowledge Article View - Service Portal," *vt4help.service-now.com*, 2022. https://vt4help.service-now.com/sp?id=kb_article&sys_id=d5496fca0f8b4200d3254b9ce1050ee5 (accessed May 04, 2024).

[13] VT4Help. 2024. "4Help - Information Technology Support : Get central-based Information Technology support - Service Portal," *vt4help.service-now.com*, 2024. https://vt4help.service-now.com/sp?id=sc_cat_item&sys_id=8353be880ff30e00005de498b1050e74 (accessed May 04, 2024).

[14] The PHP Group. 2024. "PDO" php, https://www.php.net/manual/en/intro.pdo.php (accessed May 5, 2024).

[15] International Union for Conservation of Nature and Natural Resources. 2024. "About IUCN," IUCN, https://www.iucn.org/about-iucn (accessed May 5, 2024).

[16] The PostgreSQL Global Development Group. 2024. "PgAdmin," pgAdmin, https://www.pgadmin.org/ (accessed May 5, 2024).

[17] Volodymyr Agafonkin. 2024. "Leaflet — an open-source JavaScript library for interactive maps," *Leafletjs.com*, 2010. https://leafletjs.com/ (accessed May 04, 2024).

[18] RStudio Inc. 2024. "Introduction to rnaturalearth," *Cran.R-Project*. https://cran.r-project.org/web/packages/rnaturalearth/vignettes/rnaturalearth.html (accessed May 04, 2024).