

Virginia Tech

Multimedia, Hypertext, and Information Access (CS4624)

Department of Computer Science, Blacksburg, VA 24061

Instructor: Dr. Edward A. Fox

Shark Validator Game

Authors

A. Kothari

F. Patel

R. Raya

T. Shroff

A. Tiwari

December 08, 2021

Table of Contents

Executive Summary	5
1 Introduction	6
1.1 Requirements and Objectives	7
1.2 Client	8
1.3 Challenges	8
2 Design	9
2.1 Methodology	9
2.1.1 User Types	9
2.1.2 Tasks Flowchart	9
2.1.3 Specific Services	13
2.2 UI/UX Redesign	14
2.2.1 Validation Monitor	14
2.2.2 Timeline	18
2.3 Structural Design	18
2.3.1 RShiny and JavaScript	18
2.3.2 Leaflet Map	19
3 User Manual	21
3.1 User Login	21
3.2 Validation Monitor	21
3.2.1 Gamification	22
3.3 Pulse Monitor	25
3.4 Identification Guide	27
4 Developer's Manual	30
4.1 Backend Redesign	30
4.2 Database	33
4.3 RShiny Redesign	35
5 Implementation	37
5.1 Testing	37
5.2 Evaluation and Assessment	38
6 Future Work	39
Acknowledgments	41
References	42

List of Figures

Figure 1.1: Team Member Roles	7
Figure 2.1: New User Workflow	10
Figure 2.2: Veteran User Workflow	10
Figure 2.3: Web App Developer Workflow	11
Figure 2.4: Validation Form (Previous)	15
Figure 2.5: Validation Form (Updated)	16
Figure 2.6 Validation Form (Submitted Response)	16
Figure 2.7: Validation Form (Newest Version)	17
Figure 2.8: Google Maps Link	17
Figure 2.9: Timeline and Milestones	18
Figure 2.10: Leaflet Map Render	19
Figure 2.11: Modal Box Machine Learning Classification	19
Figure 2.12: Modal Box HTML	20
Figure 3.1: The User Login Interface	21
Figure 3.2: Validation Monitor Web Page	23
Figure 3.3: Shark Validation Form (Shark In Image)	24
Figure 3.4: Shark Validation Form (No Shark In Image)	25
Figure 3.5: All-Time Leaderboard Sorted By ValPoints	25
Figure 3.6: Pulse Monitor Map	26
Figure 3.7: Pulse Monitor Pop Up Data	26
Figure 3.8: Pulse Monitor Data Table	27
Figure 3.9: Identification Guide (General Shark Species Information)	28
Figure 3.10: Identification Guide Tool	29
Figure 3.11: Identification Guide Results	29
Figure 4.1: Backend connection flow redesign	30
Figure 4.2: Custom plugin embedded into the controlling theme's functions.php	31
Figure 4.3: Excerpt from network.php	31
Figure 4.4: Excerpt from api.php	32
Figure 4.5: Redesigned Pelagic database and the relational model.	34
Figure 4.6: The problem with reconnections to the database in RShiny	36

List of Tables

Table 2.1: Validation Monitor Services	13
Table 4.1: Database table description and Backend action/connectivity.	35

Executive Summary

SharkPulse is an initiative to involve citizen scientists in monitoring global shark populations. It is inspired by Stanford's Shark Baseline Project, and it aims to collect image-based sightings of sharks from around the world to support research on ecology and conservation and increase public awareness of their conservation status. This project is currently headed by Dr. Francesco Ferretti. He is an Assistant Professor for the Department of Fish and Wildlife Conservation.

The team was provided with a platform to update and implement new design changes to the original website. The original website was built using WordPress for frontend CSS and HTML, with RShiny providing the backend to implement the Validation Monitor. We were tasked with converting the static framework of the website to a more dynamic and responsive framework, improving on the current gamification scheme with a better rewards system and incentives, sourcing data from streamlined pipes, and developing a convenient user authentication system across multiple social media platforms to allow users to log in and store their points.

The team identified three major user bases that would be interested in using the SharkPulse platform. First, we have the "New User", who will be interested in using the shark validation monitor to play the game. Then, we have the "Veteran User", who is used to the nuances of the website and is experienced in the process of validating images. Their goal will be to make it to the global leaderboard. Finally, we have "Web App Developers", who will be tasked with mining new images and maintaining the databases.

The User Interface is redesigned to increase user appeal to the overall layout from a crowdsourcing validation tool, into a fun and competitive validation game. We have added two new features to the validation modal box. The first feature is a sorted list of the top three shark predictions based on the image from a customized SharkPulse machine learning algorithm. The second feature added to the modal box is a more detailed geolocation Google Maps image. The team encountered multiple challenges in working with WordPress and RShiny. We have proposed a structural redesign using JavaScript to increase the accessibility of the software for future teams that will be working on the project. We have redesigned the backend to use AJAX scripts that use an API to request information from the database.

The team has proposed improvements on the previous gamification as implemented in the previous iteration of the project. We have written a test script in Native JavaScript that allows for a new "Time-Based" mode, where users are tasked with identifying and validating shark images in a fixed time window.

1 Introduction

Archiving and storing information for taxonomic purposes presents itself as a daunting task to researchers. The data to be collected includes but is not limited to visual characteristics, observed habitats, behavior patterns, etc. Images can assist researchers in documenting such specific features and more [8]. The goal of SharkPulse is to assist in the archiving of subclass Selachii media for better defining population characteristics for conservation purposes.

Currently, SharkPulse houses 13,107 photo records of 544 identified species. The data indicates a 49% species coverage, of which 17.4% are labeled as endangered species.

SharkPulse consists of observance statistics, identification apps, image submissions, social media presence, and publications that have benefited from SharkPulse's efforts. SharkPulse employs two major sources of media validation - crowd-sourced data and machine learning algorithms.

To facilitate filtering out unrelated media, SharkPulse has conjointly incorporated machine learning and computer vision applications to crop and extract frames of sharks from videos in addition to classification schemes to distinguish shark media from non-shark media.

Categorization classification of sharks is constantly debated for several species as they are morphologically alike to each other and there's very little knowledge that facilitates species-specific identification. Web scripts extract images from online platforms (Flickr) tagged with the word "shark" or similar words across multiple languages. The website then provides a platform for users to identify and validate images incentivized via a point-based rewards system. Our team has been tasked with extending the current rewards system and improving the design and gamification aspects of the current website. We are required to conduct research as to what design choices would enable users to stay on the website, as well as provide incentives to actively participate, such as to correctly identify the sharks from the images shown. Citizen scientists are curious and motivated members of the public who are eager to help in shark conservation but aren't necessarily provided with interactive outlets. With an improved gamified monitor, citizen scientists will be able to seamlessly interact with the platform, validate shark images, and participate in expanding species-specific data repositories describing this mysterious group of marine animals. Citizen scientists can actively participate in the data mining and collection process by uploading images of their own. Apart from online repositories, users themselves can provide valuable data in capturing shark images.

To guide understanding of this project and therefore the efforts of SharkPulse, we have mentioned the five team members and 2 mentors performing on either/both the categorization modeling facet or the main focus of this project, the Validation Monitor (Table 1). Jeremy is leading each effort as a Master's student, working with the client, Dr. Francesco Ferretti. Both the client and Dr. Edward Fox have mentored the team throughout the duration of the project. The Validation Monitor team members have assisted with multiple features of the project, but each also has devoted significant time to a specific part of the project.

Team Member	Role
Feneel Patel	Team Lead / Frontend Developer
Ray Raya	Full-Stack Developer
Aman Kothari	Backend Developer
Tirth Shroff	Frontend Developer
Ashutosh Tiwari	Backend Developer
Dr. Francesco Ferretti	Client
Dr. Edward Fox	Mentor

Figure 1.1: Team Member Roles

1.1 Requirements and Objectives

We have discussed clear and attainable objectives with our client. These will improve upon the design of the current website, and increase the gamification aspect and incentives of the website to allow users to appreciate the task of identifying and classifying shark images. The main requirements for this project are:

1. Converting the static framework of the website to a more dynamic and responsive framework.
2. Improving on the current gamification scheme with a better rewards system and incentives.
3. Sourcing data from streamlined pipes.

4. Developing a convenient user authentication system across multiple social media platforms to allow users to log in and store their points.

1.2 Client

Our client is Dr. Francesco Ferretti. He is an Assistant Professor in the Department of Fish and Wildlife Conservation. He is interested in how human impact has altered the marine ecosystems in the ocean, and how to develop solutions for sustainable use of marine resources. His research focuses on using data science methods and big data to address ecological issues and develop ocean solutions.

1.3 Challenges

There were several challenges that we ran into as a team throughout the process of working on SharkPulse. We currently use R-Leaflet for the Validation Monitor map, but it is difficult to make changes since we do not have the same flexibility with R as we do with native JavaScript. Fortunately, the same Leaflet library is offered in JavaScript (JS). If we switch, we can code the map in JS to make it more flexible and easier for implementation. Another challenge was to connect WordPress to the backend to allow for dynamic content on the web app. Unfortunately, WordPress requires proprietary/specific connections to have dynamic data be hosted on Elementor, which has significantly slowed the development of new functionalities. Lastly, we also ran into obstacles when trying to come up with the perfect gamification system. There are always ways to improve, so we have been continuously working with the client on how to improve the points, leveling, and achievements system as a whole.

2 Design

2.1 Methodology

To ensure that our changes and modifications to the project have the intended impacts, we outlined three separate sets of user experiences and workflows. These workflows serve as guides to ensure that each step of a user process is thoroughly examined and accounted for. Since we must authenticate these workflows to account for all impacts, we will require detailed descriptions of New User, Veteran User, and Web App Developer workflows.

2.1.1 User Types

New User: This user will be interested in using the shark validation monitor to play the game. The game will display shark images that this user can attempt to validate by guessing the common name or species name. Our goal is to allow new users to quickly pick up on how this game functions and its purpose by having a fluid interface for the users to interact with. We also want to encourage the users to continue playing the game, so we want to implement a scoring and reward system that allows them to earn achievements, badges, and points as they progress.

Veteran User: This user is used to the nuances of the website and is experienced in the process of validating images. They will be interested in either submitting shark photos or validating as many photos as possible to make it to the global leaderboard. Our goal is to make sure that the process for either submitting or validating is optimized to ensure the experience is enjoyable for those repetitive actions.

Web App Developer: This user primarily works in maintenance of the website, scheduling times to move validated images from SharkPulse to the PulseMonitor, and mining new images to be validated. They work a lot in the back-end to source images and move between databases. Our goal is to make sure that any such user finds the right tables in the database and has enough documentation to jump into the development of the website.

2.1.2 Tasks Flowchart

Figures 2.1, 2.2, and 2.3 illustrate the task flows for the 3 types of users, and are explained in the following subsections.

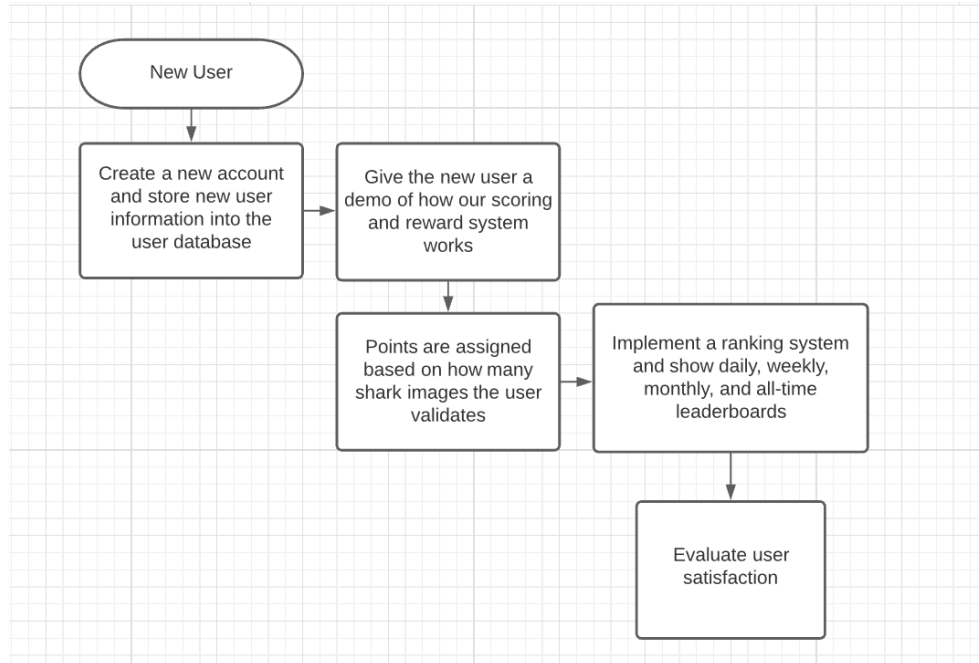


Figure 2.1: New User Workflow

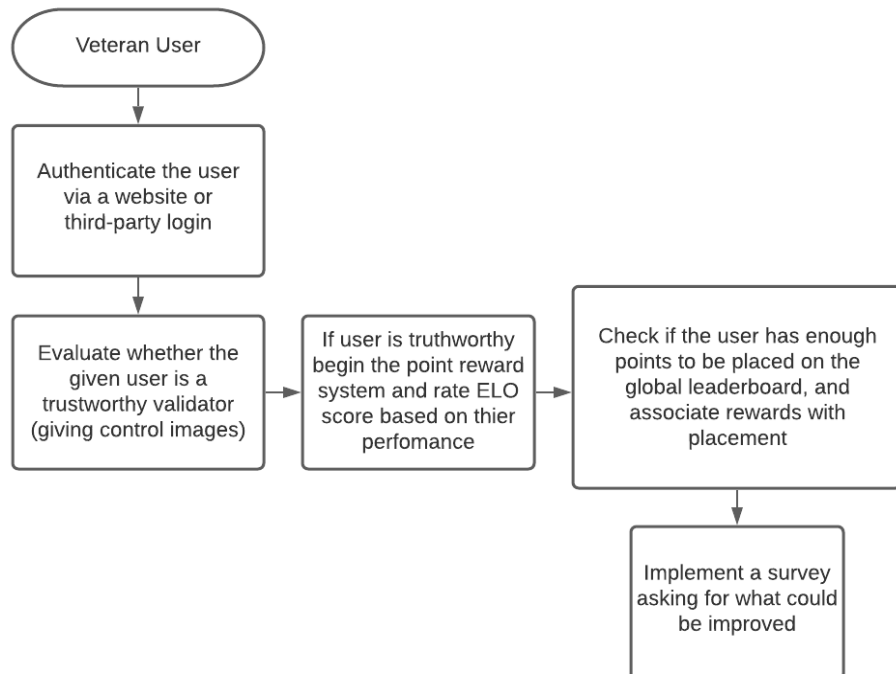


Figure 2.2: Veteran User Workflow

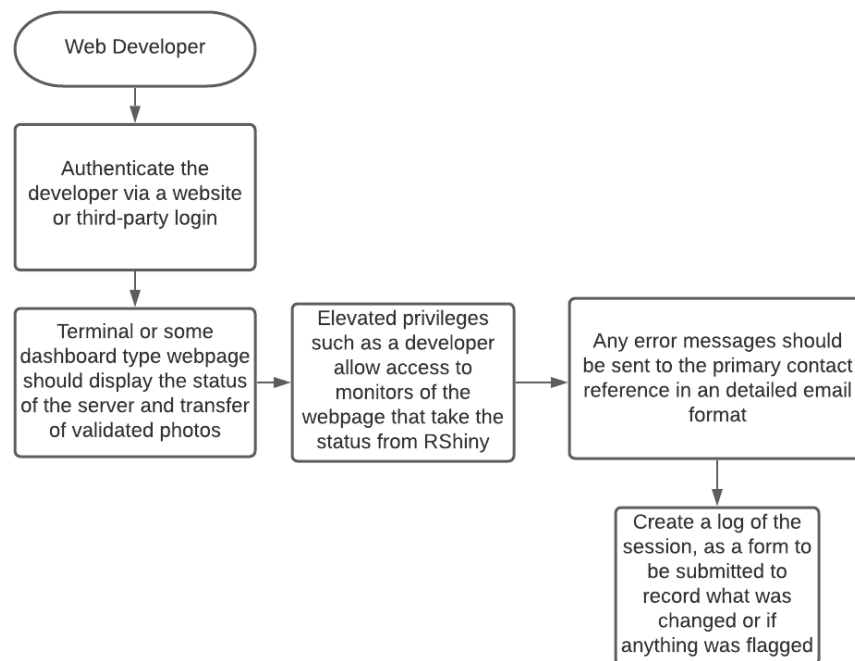


Figure 2.3: Web App Developer Workflow

New User Workflow (Figure 2.1): For the new users, we want to continuously evaluate user satisfaction to ensure that our product is well-designed and engages the users. First, we allow the new users to register an account. Once the account is created, it is populated in the user database with all scores and rewards set to 0 / None. Then, we give the users a demo of how to go about our game and application, along with how our scoring system works. Afterward, the users start validating the shark images and they start accumulating points and other rewards for correct validations. As users gain more points, they can have a chance to be displayed on our daily, weekly, monthly, or all-time leaderboards. The leaderboards will be based on how many points were accumulated by the user within that specified timeframe, and are sorted by the number of points.

Veteran User Workflow (Figure 2.2): Just like for the new users, we constantly want to improve the process for submitting and validating photos, so we will be asking for

what could be improved after each session. First, a veteran user will already have a login, so we will have to authenticate their credentials through our database or third-party authentication service. Next, as a precautionary measure, we will make the user validate some control images before assigning them points for their validations. If they do well, their skill rating will increase and they will receive points. We will need to check their points in case they have enough points to be on the global leaderboard. If they do, we need to associate the right prizes with their placement. Finally, we ask what could be improved in a short survey that they can skip if they choose.

Web App Developer (Figure 2.3): A web developer's main focus is to maintain the automated scripts/server in addition to scheduling times to move the validated images to the database. In order to ensure the developer can easily maintain the website, we need to be able to give elevated privileges to a specific account. However, first, it is essential that we authenticate the developer's account before displaying server data. Once the developer has been authenticated there will be a dashboard or terminal that shows the status of the photo transfers and the server. This data will be taken directly from RShiny for the servers and the service that transfers the photos to the database[1]. In addition, if the developer, online or offline, encounters an error, the error will be sent in a detailed formatted email. Finally, a log will be recorded that takes note of the status of all the processes, the changes made by the developer, and more generally, any flags.

2.1.3 Specific Services

Service/ Device	Input	Output	Type	Interaction by User	Dependen- cies
RShiny	Server form	HTML frame	R	Yes	Shiny, Shinyjs, RPostgreSql
Leaflet Map	Geolocation and image link	Leaflet markups on HTML frame	R	Yes	PostgreSQL connection
Modal Window	Image and validation form input	Modal window for leaflet markups	R/ HTML & CSS	Yes	Flickr URL, RShiny
Modal Window Input	Validation form answers	Change in database	PHP/ SQL	Yes	Web Server, PHP
Validation Points	User authentication	Adding to total points	PHP/ SQL	Yes	Web Server, PHP
Level System	User authentication	New level	PHP/ SQL	Yes	Web Server, PHP
Achievements & Badges	User validation	New achievement	PHP/ SQL	Yes	Web Server, PHP
User DB	User info	None	SQL	No	PostgreSQL

Table 2.1: Validation Monitor Services

Table 2.1 displays the services of the Validation Monitor functions. In addition, the table lists the library dependencies, input and output, and the type of each of the functions.

[8]

2.2 UI/UX Redesign

2.2.1 Validation Monitor

The Validation Monitor serves as a form of validating sourced images via crowdsourcing. The end goal for the Validation Monitor is to transition from being a crowdsourcing validation tool, into a fun and competitive validation game. To achieve this transformation, the previous form and layout have been redesigned. Currently, the validation monitor is based on the RShiny plugin of the JavaScript Library LeafletJS [11]. LeafletJS allows you to put markers on specific geolocations on the planet, which can also hold HTML when clicked upon. Previously, the validation forms were attached to these markers as distinct HTML entities, however, some bugs made the validation forms appear off the map (**Figure 2.4**). To improve the User Experience (UX), we changed how we formatted and attached the validation forms. To ensure that each of the validation forms is visible, we transitioned from LeafletJS Marker Popups to Modal Boxes that were activated based upon a marker click [11]. The Modal Box approach made each form always center on the middle of the screen, instead of relying on the location of the LeafletJS Marker Popups. In addition, the layout of the HTML form has been turned into a modernized format to improve the UX of submitting the forms (**Figure 2.5**).

In the newest version of the updated UX, we have added two new features to the validation modal box. The first feature is a sorted list of the top three shark predictions based on the image from a customized SharkPulse machine learning algorithm [8]. The entries on the list are displayed on the middle-left side of the modal box, formatted as such: scientific name of shark species, and confidence percentage of the prediction (**Figure 2.7**). This information is displayed in order to assure the user of their submission, or to help the user start their search. The second feature added to the modal box is a more detailed geolocation Google Maps image. This image is a more accurate depiction of the longitude and latitude location of the web sourced image. The image is also accompanied by a Google link that takes the user to a more detailed page for the location (**Figure 2.8**). The purpose of this feature is for the user to be able to click on the link and determine whether the location is an aquarium or not. This serves as a way to ease the difficulty of the game, in addition to adding an extra layer of validation to the form.

To demonstrate the extended functionalities of the redesigned validation monitor we will walkthrough a simulated validation. To begin, the user is prompted to sign into or register a new account, to save their progress and points. After authentication, the user will have a global map displayed with multiple geolocation markers, pointing to web-sourced shark sightings (**Figure**

2.4). The user will click on any marker and the associated marker data will be displayed in a modal box form. This box will contain the following: web-source shark image, top three shark species predictions, detailed Google Maps geo marker image/link, identification link, and input form[11], [14]. This input form will prompt the user to determine whether the image displayed was a shark, whether the picture was taken in an aquarium, and naming conventions for the shark, alongside having a prompt for additional comments (**Figure 2.7**). If the user cannot get a sufficient view of the image, the user may hover over the image in order to resize the image to fit the entire modal box. If the user requires aid to identify the shark that is shown, they may use the displayed machine learning predictions alongside the identification guide to narrow the choices. To ensure that the data is validated correctly the user can click on the Google Maps image link to investigate whether the location is an aquarium or not. Finally when the user has entered data in all the fields they will click submit and be redirected to a submission response page (**Figure 2.6**). If they prefer to validate more images they can click the “return to map” button.

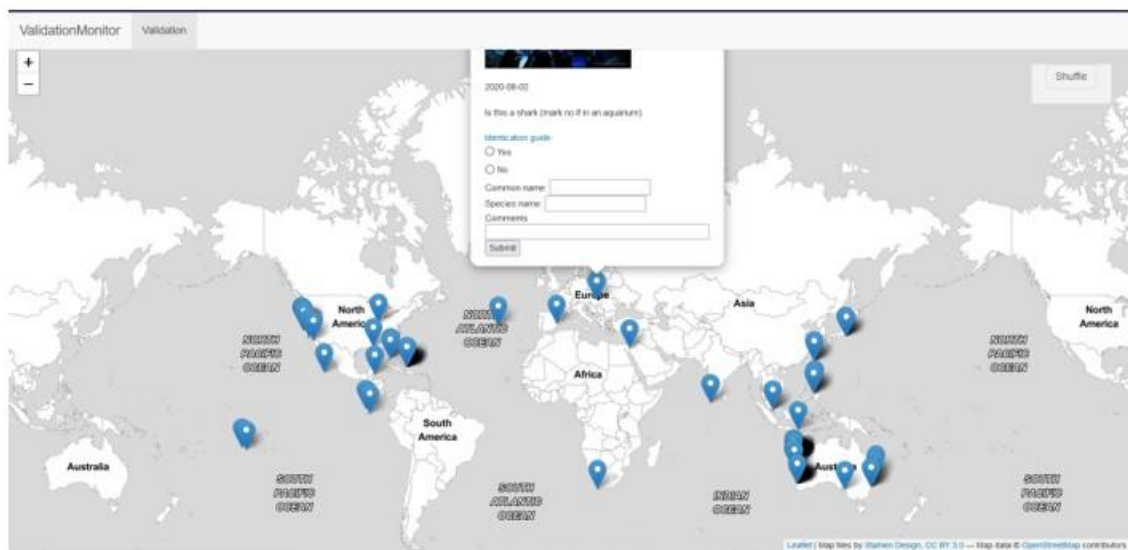



Figure 2.4: Validation Form (Previous) [14]

ValidationMonitor
Validation

+
-

NORTH PACIFIC OCEAN
SOUTH PACIFIC OCEAN

Please validate the following shark image.



Is this a shark?
In an aquarium?

Yes No
Yes No

Common Name

Species Name

Comments

16401
Is this a shark (mark no if in an aquarium)
Identification guide

Submit

Dismiss

Figure 2.5: Validation Form (Updated)

Thank You

Here is the information you have submitted:

1. Shark: yes
2. Common Name:
3. Species Name: *Ginglymostoma cirratum*



4. image ID: 39879
5. email:

Return to Map

Figure 2.6: Validation Form (Submitted Response) [8] []

ValidationMonitor

Validation

+

-

Canada


United States of America

Mexico

Be

EL

Please validate the following shark image.



Is this a shark?

Yes

No

In an aquarium?

Yes

No

Common Name

Species Name

Ginglymostoma cirratum

Comments

Submit

Suggested Species:

Ginglymostoma cirratum 0.38

Carcharhinus melanopterus 0.12

Heterodontus portusjacksoni 0.11

Post URL

Identification guide

View larger map

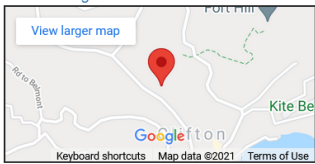


Figure 2.7: Validation Form (Newest Version)

Figure 2.8: Google Maps Link

17

2.2.2 Timeline

To ensure that we were able to meet the deliverables on time, we constructed a monthly-based timeline (**Figure 2.9**). The timeline below translates into a four-step process: Familiarize, Improve, Update, Finalize. By following this timeline as a guideline, we stayed on track to meet our deliverables on time.

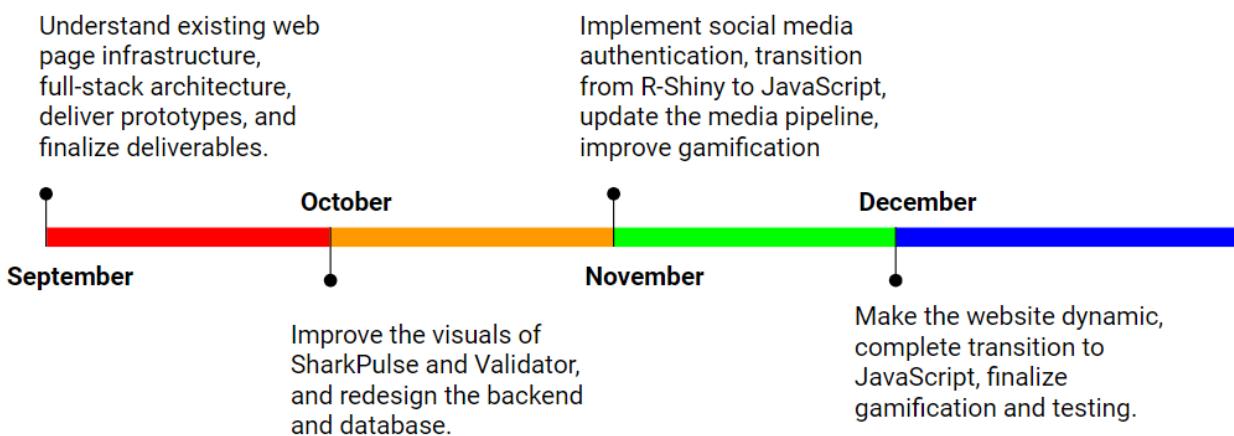


Figure 2.9: Timeline and Milestones

2.3 Structural Design

2.3.1 RShiny and JavaScript

The Validation Monitor is the output of an RShiny script that uses the R rendition of LeafletJS. The issue with using RShiny in this sense is that we are using it as a web host and we are not using R for its intended purpose, which is statistical analysis [2], [3]. Due to this, we do not have the full range of manipulation of the DOM (Document Object Model), and therefore we cannot manipulate the UI or functionality of the Leaflet Map or Markers. It would be beneficial to transition from RShiny to Native JavaScript [5].

2.3.2 Leaflet Map

Figure 2.10 displays how the Leaflet Map is being rendered in RShiny. RShiny is able to render leaflet markers and maps through a ported LeafletJS library to RShiny. Utilizing a ported library leads to limited flexibility, as opposed to developing it in its native language.

Figure 2.11 shows how each of the Modal Boxes in the map display the top three classifications that the machine learning classifier resulted in, along with the accuracies for each species provided. **Figure 2.12** shows a small portion of the code which displays all the other information provided in the Modal Boxes.

```
# Render a map
leaflet(data = dat) %>%
  addProviderTiles(providers$Stamen.TonerLite) %>%
  addMarkers(layerId = 1:length(dat$id), clusterOptions = NULL)
# markerClusterOptions(showCoverageOnHover = FALSE)
})
```

Figure 2.10: Leaflet Map Render

```
s = which(is.na(dat$top1_species))
dat$top1_species[s[1:length(s)]]='--This image may not contain a shark OR we cannot identify the species--'
dat$top2_species[s[1:length(s)]]=''
dat$top3_species[s[1:length(s)]]=''

dat$top1_probability=round(as.numeric(dat$top1_probability),2)
dat$top2_probability=round(as.numeric(dat$top2_probability),2)
dat$top3_probability=round(as.numeric(dat$top3_probability),2)

p = which(is.na(dat$top1_probability))
dat$top1_probability[p[1:length(p)]]=''
dat$top2_probability[p[1:length(p)]]=''
dat$top3_probability[p[1:length(p)]]=''
```

Figure 2.11: Modal Box Machine Learning Classification

```

output$sharkmap <- renderLeaflet({

  observeEvent(input$sharkmap_marker_click, {
    id = input$sharkmap_marker_click
    index = as.numeric(id$id)

    showModal(modalDialog(
      title = "Please validate the following shark image.",
      size = "l",
      HTML(
        '<!DOCTYPE html>
        <html>
        <head>
        <script>
          document.addEventListener("change", function(event) {
            let element = event.target;
            if (element && element.matches(".form-element-field")) {
              element.classList[element.value ? "add" : "remove"]("-hasvalue");
            }
          });
        </script>
        <style>
        .switch-field {
        display: flex;
        margin-bottom: 10px;
        overflow: hidden;

```

Figure 2.12: Modal Box HTML

3 User Manual

3.1 User Login

SharkPulse users can log in on the Validation Monitor to start earning points for each shark image they validate. It is not required for users to be logged in to play the game, however, we provide the users with incentives to have an account and be logged in when validating sharks to level up by earning points, and earn various achievements and badges. When logging in, we provide the users' multiple options. They can either create an account with us or use their Google account to register.

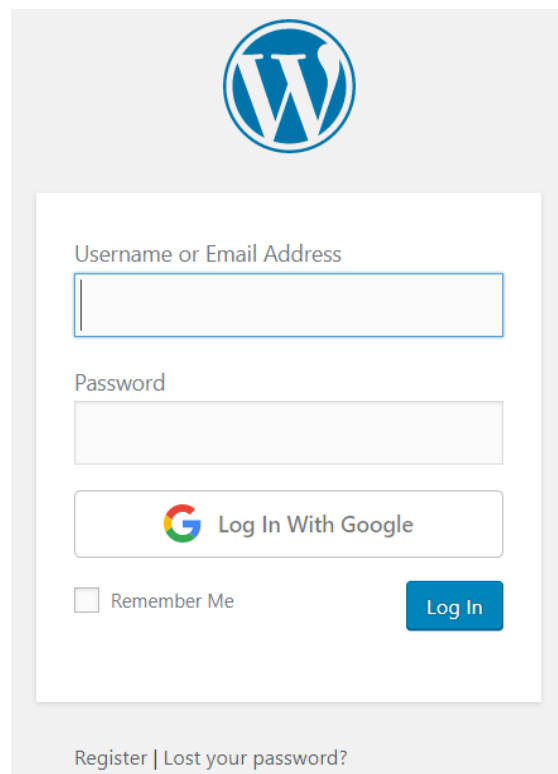
The image shows a user login interface. At the top center is a circular logo with a blue 'W' and a shark fin. Below the logo is a white rectangular form. Inside the form, there are two input fields: the first is labeled 'Username or Email Address' and the second is labeled 'Password'. Below these fields is a button with the Google 'G' logo and the text 'Log In With Google'. Underneath this button is a checkbox labeled 'Remember Me'. To the right of the checkbox is a blue button with the text 'Log In'. At the bottom of the form, there are two links: 'Register' and 'Lost your password?'. The entire form is set against a light gray background.

Figure 3.1: The User Login Interface [14]

3.2 Validation Monitor

Validation Monitor is the game aspect of this web application. Users are directed to the web page as shown in **Figure 3.2** when they click on Validation Monitor. This page displays the

current user and the validation map that users can play the game through. **Figure 3.3** shows what the user is shown when they click on one of the pop-up bubbles from the map. They are asked ‘Is this a shark?’ and ‘Is it in an aquarium?’ If the user gets stuck trying to validate the sharks, they can refer to the identification guide that is provided to them on the validation form. That helps the user in identifying the shark in terms of their ‘Common’ name and ‘Species’ name. Refer to **Figure 2.5** to view an example of how the user would answer if the image being displayed does not include a shark. When the image is a shark but is in an aquarium, we don’t want the users to try to validate it. We collect that information to send it to our machine learning classifier as input data to help increase its accuracy in identifying sharks. **Figure 3.3** also displays the current accuracies of the machine learning algorithm classification. It ranks the top three options of suggested species that the computer detected from the shark image. **Figure 3.4** shows what the validation form displays if the image being shown does not contain a shark, or if the algorithm is unable to use the image to identify it. The machine learning classifier was developed by Jeremy Jenrette, who is a previous student of CS 4624 and one of the members of the team that was responsible for developing SharkPulse.

3.2.1 Gamification

Users can earn points from validating sharks in several ways, as seen in **Figure 3.2**. If the users only identify if it’s a shark and if it’s in an aquarium, then they earn 2 points total. They earn 3 points total if they also provide either the ‘Common’ name or the ‘Species’ Name. They can earn 5 points if they answer all of the questions provided on the validation form. The users will receive boosts on their points if the shark has already been validated. If the database already has the correct answers stored for an image that has been manually validated, the user can receive up to 50 points. The aim is to motivate our new users by providing them with already validated shark images. For veteran users, we want to focus more primarily on the images that are directly sourced from the Flickr pipeline. Not all of the images sourced from the pipeline are guaranteed to be sharks in the ocean, so the aim is to have the veteran users help validate those images because they have proven to be well-educated on the subject matter. Users are also able to keep track of their current ValPoints and level while seeing the top performers on SharkPulse, as shown in **Figure 3.2** and **Figure 3.5**.

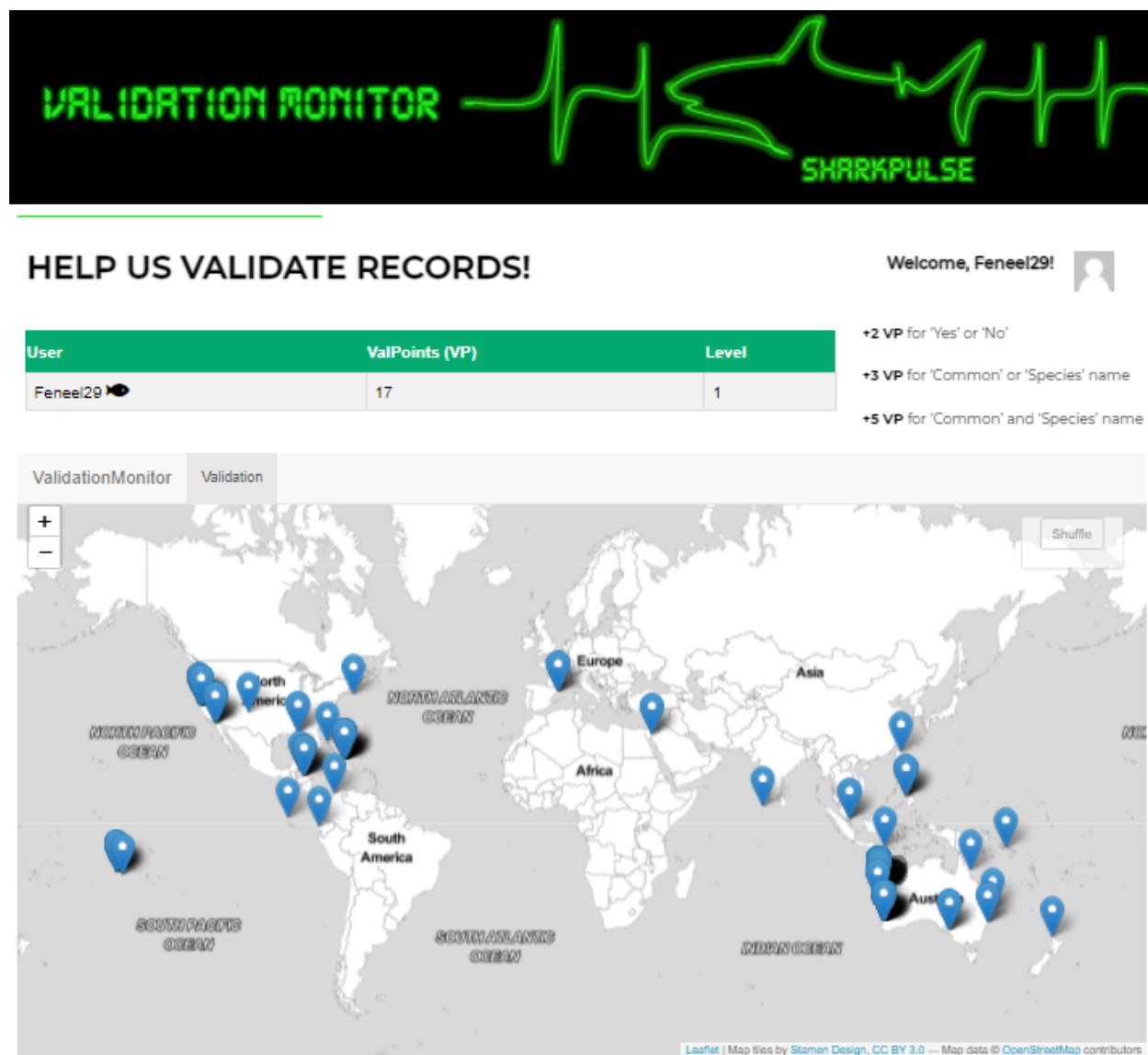



Figure 3.2: Validation Monitor Web Page [14]

Please validate the following shark image.

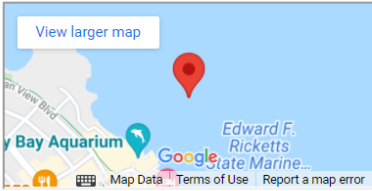


Suggested Species:

Carcharhinus perezii 0.89
Carcharhinus limbatus 0.06
Carcharhinus spp. 0.02

Post URL

Identification guide



Is this a shark?

In an aquarium?

Common Name

Species Name

Comments

Figure 3.3: Shark Validation Form (Shark In Image) [14]

Please validate the following shark image.

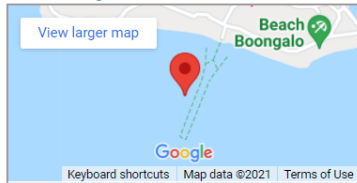


Suggested Species:

--This image may not contain a shark OR we cannot identify the species--

Post URL

Identification guide



Is this a shark?

Yes

No

In an aquarium?

Yes

No

Common Name

Species Name

Comments

Submit

Figure 3.4: Shark Validation Form (No Shark In Image) [14]

LEADERBOARD




User	ValPoints	Level
jeremy 	1093	5
Dhruv Dharamshi 	761	4
Francesco Ferretti 	442	3

Figure 3.5: All-Time Leaderboard Sorted By ValPoints [14]

3.3 Pulse Monitor

Pulse Monitor holds all of the validated images. SharkPulse users can explore the world map and go through different locations to see various shark images and classifications. For those interested in specific shark images, they can filter by 'Common' name or 'Species' name. Pulse Monitor also provides a data table, which displays the data records that currently exist in the database, as shown in **Figure 3.7**.

PULSE MONITOR

Explore the map and database



Figure 3.6: Pulse Monitor Map [13]



Figure 3.7: Pulse Monitor Pop Up Data [13]

PULSE MONITOR

Explore the map and database

SharkPulse Monitor

Map

Data Table

Species name:

☒ Scientific name

All

☐ Common name

Table

Show

10

entries

Search:

	date	species_name	common_name	location	source
1397	2017-03-28	Alopias superciliosus	Bigeye thresher	Argolic Gulf, Greece	argolikeseldhseis.gr
1391	2017-02-22	Cetorhinus maximus	Basking shark	Mortorio Island, Italy	Operazione Squalo Elefante
1507	2017-02-22	Alopias vulpinus	Thresher	Castellammare del Golfo, Italy	post on SharkPulse
6839	2018-08-06	Prionace glauca	Blue shark	Palavas Les Flots, France	20minutes.fr
405	2017-05-10	Cetorhinus maximus	Basking shark	Istria, Croatia	24sata.hr
31886	2017-10-26	Carcharodon carcharias	Great white shark	Rimini, Italy	4Fishing Facebook
6896	2018-06-19	Cetorhinus maximus	Basking shark	Gloucester, Massachusetts	7seaswhalewatchFB
6420	2018-01-17	Carcharhinus leucas	Bull shark	Georges River, Sydney, Australia	9news
1534	2017-12-29	Hexanchus griseus	Bluntnose sixgill shark	Mersin, Turkey	aa.com.tr
6307	2017-10-17	Negaprion brevirostris	Lemon shark	Darwin, Australia	Abc.net.au

Showing 1 to 10 of 32,562 entries

Previous

1

2

3

4

5

...

3257

Next

Figure 3.8: Pulse Monitor Data Table [13]

3.4 Identification Guide

The Identification Guide was already implemented in SharkPulse prior to our team working on it by one of the previous teams. The Identification Guide is offered to every user when they are playing the Shark Validator Game. Its main purpose is to help guide users through validating a shark image, especially if the users are not as familiar with shark species names or if they get stuck and require further assistance. **Figure 3.8** shows how the users can use their systems to set up a split-screen with the Validation Monitor on one side, and the Identification Guide on the other to actively learn more about whichever species the image currently displays. It is currently displaying the general overview of the White Shark. **Figure 3.9** shows how the users can use the tool that asks several questions about the shark, such as the type of snout the shark has, the number of gill slits, if the mouth is ahead or behind the eyes, etc. to help the guide identify

exactly which type of species the shark is. If the guide is unable to come up with a concrete answer, it will list out multiple different species the shark is most likely to be, as displayed in **Figure 3.10**. By using these various tools from the guide, the users stay engaged and remain interactive with the validation game, even if they were to get stuck on a specific shark image and need further assistance.

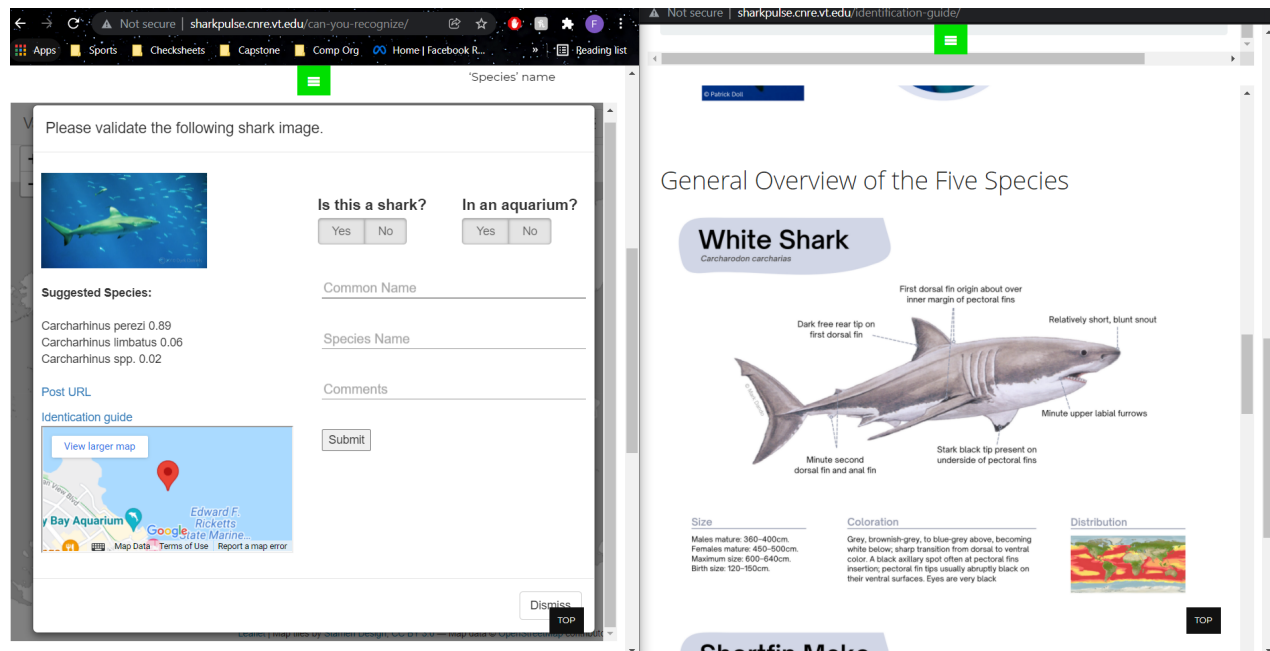


Figure 3.9: Identification Guide (General Shark Species Information) [14]

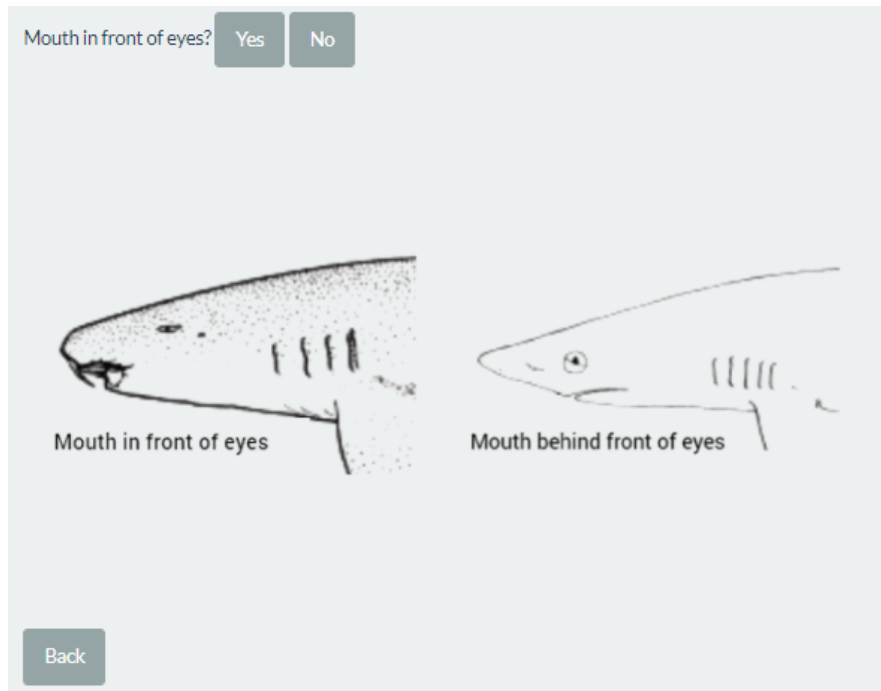


Figure 3.10: Identification Guide Tool [14]

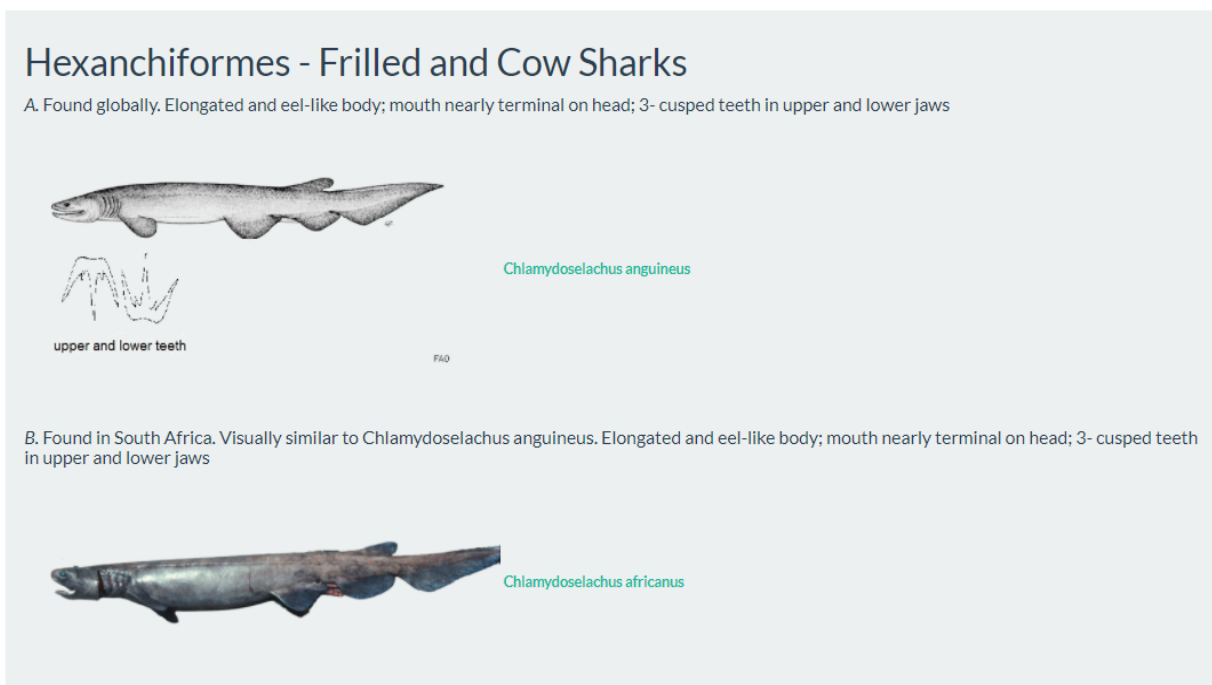


Figure 3.11: Identification Guide Results [14]

4 Developer's Manual

4.1 Backend Redesign

Originally the connection between the front and back end was superfluous with static HTML content being echoed from scripts in the backend and rendered on the front-end with no knowledge of content around it. We found it impossible to implement dynamic functionality using JavaScript in the current model. Further, there was internal fragmentation among the scripts in the backend with one not knowing the other. Information was calculated and re-calculated or pulled from the database which increased the load average usage of the CPU. [1] Thus, we decided to entirely redesign the backend using AJAX scripts, taking advantage of this functionality embedded into the WordPress backend organization. [4] [10]

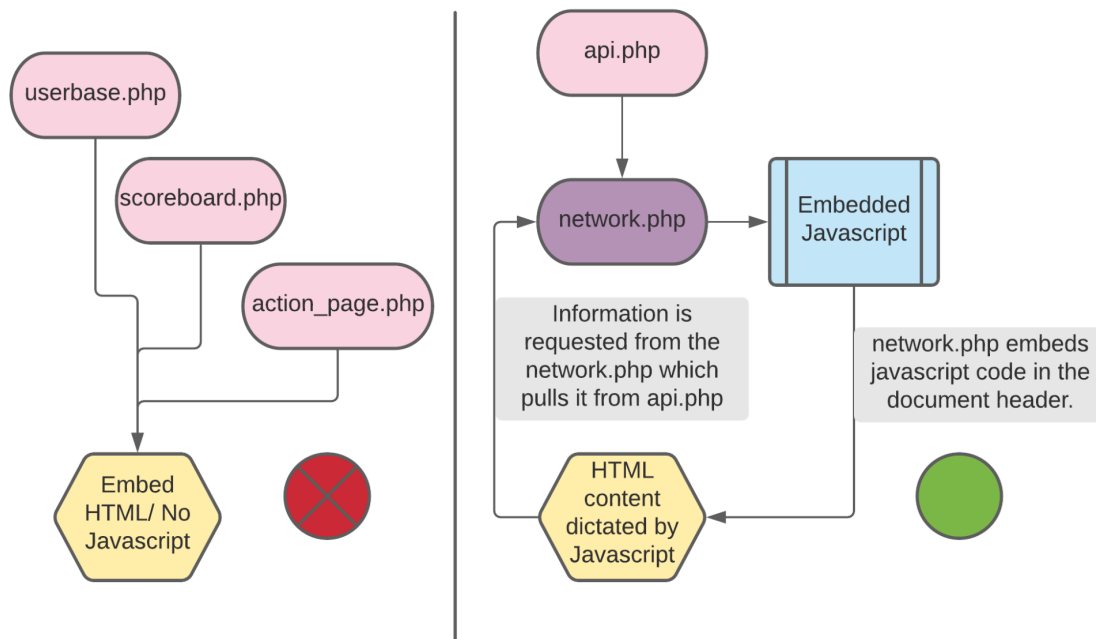


Figure 4.1: Backend connection flow redesign

Figure 4.1 shows a high-level implementation of a network model where information is requested and received by the front end using a network-flow model. All the HTML content is dictated by the JavaScript embedded into the document header when the page is loaded for dynamic updates. Further, instead of having separate scripts for each element, all the HTML

content is generated by a single *api.php* such that it has contextual information about the rest of the page. This entire network model is implemented using AJAX callback functions which can update information on the page dynamically without the need to reload the entire page.

Let us dive deeper into the implementation of AJAX calls with WordPress. To alter the WordPress content dynamically, we needed to gain access to the DOM elements generated by WordPress. To that end, we insert our custom *network.php* plugin **into the functions.php file of the controlling theme**. One major drawback of this method is that this step needs to be repeated every time the WordPress theme is updated or changed. WordPress erases any custom code from the theme's controlling *functions.php* every time it updates. [4]

```
// -----Custom Plugin-----  
if(!@include($_SERVER['DOCUMENT_ROOT']. "/pulseMonitor/network.php"))  
    throw new Exception("Failed to include 'script.php'");
```

Figure 4.2: Custom plugin embedded into the controlling theme's *functions.php*

```
1 <?php  
2 include_once('api.php');  
3  
4 add_action( 'wp_enqueue_scripts', 'theme_name_scripts' );  
5 function theme_name_scripts() {  
6     if (is_front_page()) {  
7         wp_enqueue_script( 'script-name', '/pulseMonitor/homepage.js', array('jquery'), '1.0.0', true );  
8         wp_localize_script( 'script-name', 'MyAjax', array(  
9             // URL to wp-admin/admin-ajax.php to process the request  
10            'ajaxurl' => admin_url( 'admin-ajax.php' ),  
11            // generate a nonce with a unique ID "myajax-post-comment-nonce"  
12            // so that you can check it later when an AJAX request is sent  
13            'security' => wp_create_nonce( 'my-special-string' )  
14        ));  
15     } elseif (is_page('can-you-recognize')) {  
16         wp_enqueue_style('table-format', '/pulseMonitor/example.css', array(), '1.0.0');  
17     }  
18 }  
19  
20 // The function that handles the AJAX request  
21 add_action( 'wp_ajax_my_action_ok', 'my_action_callback' );  
22 add_action( 'wp_ajax_nopriv_my_action_ok', 'my_action_callback' );  
23 function my_action_callback() {  
24     check_ajax_referer( 'my-special-string', 'security' );  
25     $whatever = get_sharkpulse_records();  
26     echo $whatever;  
27     die(); // this is required to return a proper result  
28 }
```

Figure 4.3: Excerpt from *network.php*

Once the plugin is embedded into the controlling theme's *functions.php* we can use WordPress inbuilt **wp_enqueue_script** and **wp_enqueue_style** functions to add custom JavaScript code and CSS styling to the *head* of the page's DOM structure. This placement would allow us to control the styling and functionality of the entire page through JavaScript and CSS code written in the backend. Furthermore, we can define dynamic content based on user interactions using JavaScript, rendered on the user's browser without additional strain on the server. This model allows us to delegate tasks to the user's JavaScript engine instead of relying solely on the server's computational capabilities. One thing to note is that since the AJAX functionality is derived from the WordPress source code, the JavaScript code needs to be localized using the **wp_localize_script** function call to give it access to the AJAX/jQuery code. [4]

```
1  <?php
2  require_once('postgreConfig.php');
3  // include_once('/var/www/html/sharkPulse/wp-load.php');
4  function get_sharkpulse_records()
5  {
6      global $dbconn;
7      $sql = "SELECT count(sp_id) FROM sharkpulse;";
8      $resArr = pg_query($dbconn, $sql);
9      $result = pg_fetch_row($resArr);
10     return $result[0];
11 }
12
13 function scoreboard()
14 {
15     global $dbconn;
16     $sql = "select username, email, points, level from userbase order by points desc limit 3;";
17     $result = pg_query($dbconn, $sql);
18     $resultArr = pg_fetch_all($result);
19     $return_val = "<table class=\"points-table\">
20     <tr>
21     <th>User</th>
22     <th>ValPoints</th>
23     <th>Level</th>
24     </tr>";
25
26     foreach ($resultArr as $array) {
27         $badge = get_badge($array['level']);
28
29         $return_val .= '<tr>
30             <td>' . $array['username'] . ' ' . $badge . '</td>
31             <td>' . $array['points'] . '</td>
32             <td>' . $array['level'] . '</td>
33             </tr>';
34     }
35
36     $return_val .= '</table>';
37 }
```

Figure 4.4: Excerpt from api.php

The *api.php* allows us to dictate the content generated for each separate information request through a common API that has contextual information about the rest of the functions described for the current page. It adopts a model where a central controlling function calls different private functions in the API to provide information about the current state of the database and DOM elements required to render the page correctly. This protects the implementation from vulnerabilities while providing accurate information about the current state. One good example of how the *api.php* allows us central control is by comparing it to the *userbase.php* and *scoreboard, PHP*. Both the latter scripts contained similar computation code that called for the points and achievements of users. A change in one source code would not be reflected in the other file. This creates a disconnect in the application of our model. *api.php* solves this conflict by collecting all the source code into a single function that is shared by the requests for information for the userbase or the scoreboard. All the JavaScript code is written using jQuery sourced from WordPress for uniformity, and vanilla JavaScript. [4]

4.2 Database

The Validation Monitor for Sharkpulse is hosted on a PostgreSQL database, an open-source relational database management system (RDBMS) that focuses on SQL compliance and scaling. [6] For our project, we have completely redesigned the database structure based on user-flow analysis, to obtain a coherent and robust model that offers significant advantages when scaling upwards towards a large number of database records.

As seen in **Figure 4.5**, all the records mined (sourced from Flickr, Instagram, etc.) are stored in the **data_mined** table with all their identifying information to locate the picture again, if required. [1] This table is also the primary table that will be used to train our Neural-Network model to identify sharks and their taxonomic information. This table is the primary table whose *id* connects the validations to all other tables. The **userbase** table contains identifying information for all the users of this validation game. This table may be expanded in the future when authentication through social media platforms is implemented. The **user_validations** table connects the **data_mined** table to the **userbase** table for user validations. It stores the user input of *species_name*, *common_name*, *comments*, *is_shark* among other information to validate any image. This separate structure allows separate individual components to reduce the load on database transactions. Another feature introduced in this database redesign is the **Taxonomy** table that provides a robust framework to input all hierarchical levels of taxonomic information.

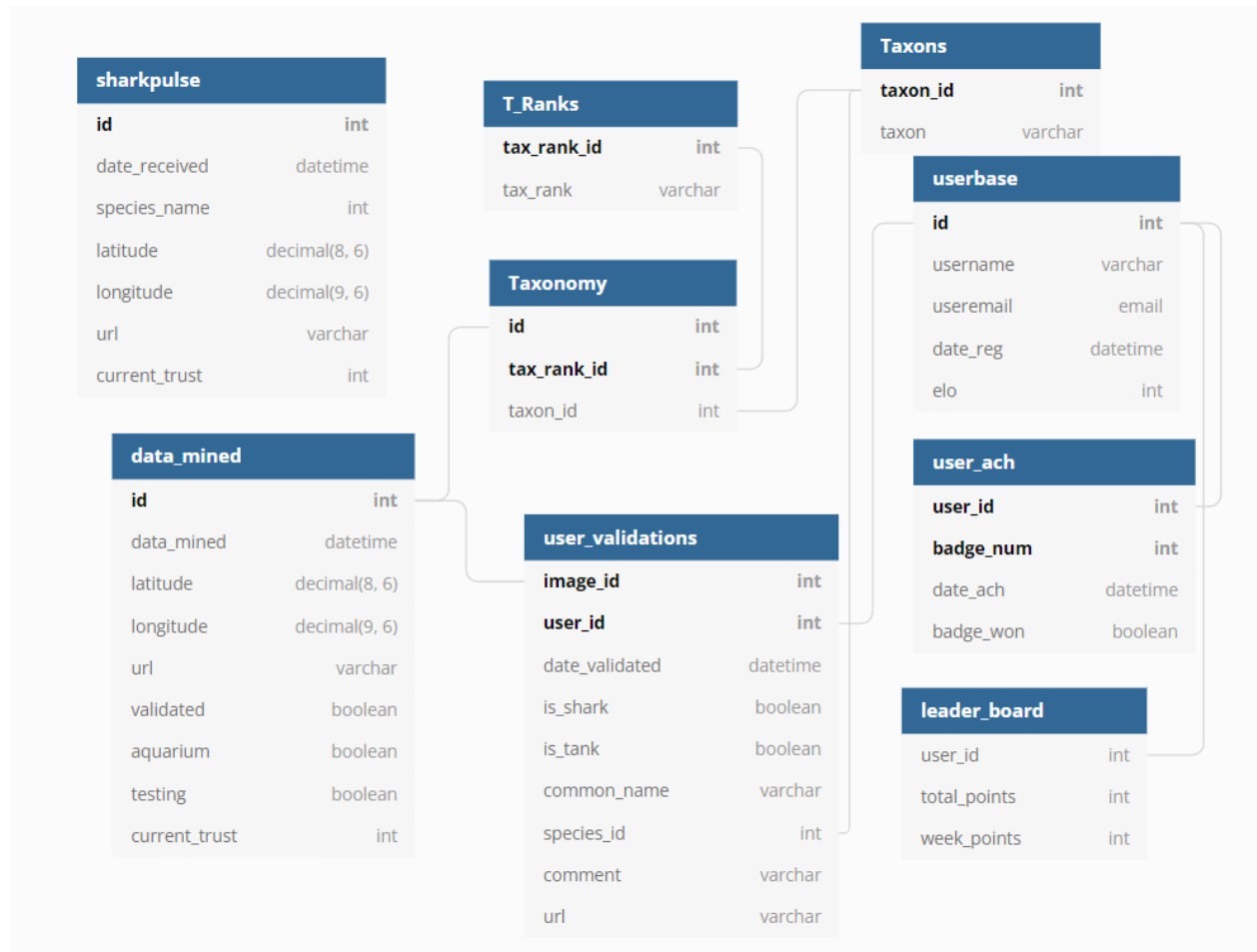


Figure 4.5: Redesigned *Pelagic* database and the relational model.

Currently, our model only takes *species names* as taxonomic information input. However, this redesign offers the ability to expand that to all the taxonomic levels if the need arises. The string taxons are stored in the **Taxons** table to optimize memory savings in case of repeated strings. Furthermore, to implement our new aspect of gamification and user achievements, we have introduced the **user_ach** and **leader_board** tables that add a new dynamic to the gamification. The **leader_board** table holds points based on overall and weekly achievements to allow users to compete between themselves. Refer to **Table 4.1** for comprehensive usage of these tables in the backend.

DB Table Name	Description	Backend action
SharkPulse	All validated records.	-
data_mined	All sourced images with identifying info.	Sourced by Validation Monitor on the map
userbase	All user records	Individual score + leaderboard
user_validations	All user validation input for each image	Sourced when the image is marked validated
Taxonomy	All user input taxonomy	-
T_Ranks	Defining all Taxonomic Ranks	-
Taxons	All users input Taxon strings as key-value	-
user_ach	Holds user achievements	Sourced on user homepage
leader_board	Holds user weekly points	Sourced for the weekly leaderboard

Table 4.1: Database table description and Backend action/connectivity.

4.3 RShiny Redesign

As mentioned previously, the Validation Map sourced its content from an RShiny script in the backend. [2] [3] This R script, named *server.R*, initially connects to the PostgreSQL database and sources 200 images (including control images that have already been validated previously). These images are rendered on the map as markers whose modals show up when clicked upon. However, during testing, the connection to the Validation Map was frequently overwhelmed. [1] We were able to identify this problem to be the way the server loads the script every time one image on the map is validated by a user.

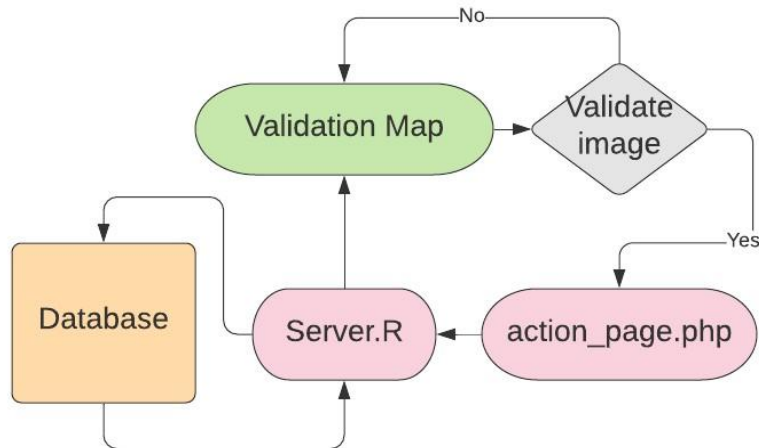


Figure 4.6: The problem with reconnections to the database in RShiny

As seen in Figure 4.6 every time an image is validated on the Validation Monitor it is redirected to the *action_page.php* script which instead of going back to the validation page sources the *server.R* script again. This leads to the creation of a new database connection and images being sourced from the database again. This model quickly overwhelms the database with the incoming connections and stalls the server. [1]

To counter this problem we propose, as an immediate short-term solution, to introduce an essential hack, preserving the model that calls on the *action_page.php* script but instead redirecting its output to an iframe located in the modal's HTML content. We have not been able to implement this feature due to the testing period. However, we strongly recommend this be input into the script to handle a greater number of connections to the website more efficiently.

5 Implementation

5.1 Testing

We used the RStudio IDE for doing most of our development especially for certain fixes on the frontend. [2] This is where we tested the functionality of our Shiny app for its server and database interactivity. [3] As of now, since we have not completely made a JavaScript version of User Authentication, we are sticking with the WordPress login/logout feature integrated with our app.

Further, for connecting the user profiles with the backend database, we made use of PHP. To make this connection within SharkPulse, we used PHP pathways to receive user input from the HTML forms in the Validation Monitor. To avoid a clumsy backend, we managed to process all the information and implemented all functions within just one PHP script. We also tested out everything on the live server using PHP. For tracking user-specific data like Validation Points, Ranks, metadata, and appendages, the PHP action script was tested and implemented. Finally, for being able to connect with the local Pelagic database, all PHP scripts included the `postgreConfig.php` script which inputs the client master user and password for logging into the database.

We have tested the most recent version of the website with about 30-40 people. During this testing phase, we came across certain issues with the server. One of the major ones was that the server was not able to keep up with so many connection requests simultaneously. Due to this, the server would just keep crashing. It was difficult to have so many users log into the system at the same time due to this reason.

We isolated the problem and determined the core reason for this crash. The way the Shiny script was written, it would reload the validation page over and over again. This led to sourcing data repeatedly from the database. Thus, the database was not able to keep up with so many connection requests and the server would eventually crash.

Regarding the front end, we got a positive response. We received feedback on how the Shark Validation process and the entire game were a fun experience overall for the users. The modal

box was easy to navigate and aesthetically pleasing. It was a nice learning experience for the users.

5.2 Evaluation and Assessment

The evaluation and assessment of this project are solely based on the user experience and the amount of data that is collected for validated images. For this iteration of sharkPulse, a lot changed. However, we still do not have enough user data and have gone through the testing phase with only about 30-40 people. Over time, a lot more feedback should be gathered with more users testing out this app.

6 Future Work

Although SharkPulse has come a long way since it was initially created, there are still many ways it can be improved. In terms of gamification, there can be more achievements, badges, and rewards available for players to earn to keep encouraging the player base to continue playing the game. These rewards can be made more lucrative by implementing user profiles so that other players can view exclusive rewards earned by certain players to motivate them to also play the game more to earn those rewards. Expanding upon the current leaderboards would also help motivate players by displaying the top players weekly, monthly, and all-time in terms of score and rank.

One major addition that can be made to the Validation Monitor is to fully integrate the timer mode into the game. We want to allow the users to be able to choose between a casual mode and a time-based mode. The time-based mode would give the player 5 or 10 minutes to try to validate the most shark images relative to the other players. SharkPulse can hold weekly, or monthly, competitions for players where exclusive rewards and badges are awarded to those who perform well in those events. The aim is to encourage existing players to continue visiting the application on a regular basis, and challenge them along the way, so the most dedicated and knowledgeable players can have something to show their commitment. New users will likely still be more attracted to the casual mode as there is not as much stress, due to there being no time limit.

We have completely redesigned the backend to use an AJAX model with all information being requested from an API. This model allows for a lot of flexibility in the data being sent to the client. [10] However, we were not able to write an extensive API that checks for robustness and efficiency. We have provided a solid foundation to be expanded upon and that should be one of the foremost tasks moving forward. Furthermore, we have implemented the Validation Map in JavaScript using the LeafletJS library, if the need arises to switch away from RShiny.[11] We strongly recommend moving away from RShiny as JavaScript offers much more modularity and dynamic rendering to be done on the page. It gives us end-to-end control over what is displayed on the front end. However, we acknowledge the client's need to stick with RShiny and so also propose a solution to make the server more robust and efficient, as mentioned in Section 4.3. Further, as a long-term plan to improve robustness and efficiency, we propose to use database-pooling to manage connections to the database. There is evidence to suggest that a database-pooling approach provides increased efficiency, manageable connections to the

database, and robust concurrent accesses from the database. [12] It organizes connections in a way to realize our target within our limited resources and abstract it from the user.

As we move forward, this project will also need a lot of changes to handle the user requests that it is not able to as of now. This would mean shifting to JavaScript for a better server to database connection and a smooth backend overall. It has been one of the major changes that have to be implemented in this project. Further, this project would also encompass efforts from collaborators of the client to expand sharkPulse via media attention and research output.

Acknowledgments

Client: Dr. Francesco Ferretti, Assistant Professor, Department of Fish and Wildlife Conservation, SharkPulse creator, ferretti@vt.edu

Mentor: Dr. Edward Fox, Professor, Department of Computer Science, fox@vt.edu

References

- [1] Ferretti, F. (2015). SharkPulse organization. <http://sharkpulse.cnre.vt.edu/>
(Accessed September 2021.)
- [2] R Core Team. (2020). R: A language and environment for statistical computing.
<http://www.Rproject.org/> (Accessed September 18, 2021.)
- [3] RStudio Inc. (2013). Easy web applications in R. <http://www.rstudio.com/shiny/>
(Accessed September 18, 2021.)
- [4] WordPress. (2021). WordPress website builder. version 5.7.1. <https://wordpress.com/>
(Accessed September 27, 2021.)
- [5] Mozilla. (2005). JavaScript <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
(Accessed October 3, 2021)
- [6] The PostgreSQL Global Development Group. (1996) PostgreSQL
<https://www.postgresql.org/> (Accessed October 13, 2021)
- [7] Elementor. (2021). Elementor code reference. v 3.2.1. <https://code.elementor.com/>
(Accessed October 28, 2021.)
- [8] Jenrette, J., Chang, G., Gordon, S., Mulgrew, M., Debay, H. (2021). VTechWorks. CS4624
team term project, May 2021, Virginia Tech, Blacksburg, VA,
<http://hdl.handle.net/10919/103254> (Accessed November 11, 2021)
- [9] Ferretti, F. Virginia Tech Department of Fish and Wildlife Conservation.
<https://fishwild.vt.edu/faculty/ferretti.html> (Accessed November 11, 2021)
- [10] Mozilla. (2005). "Ajax - Developer Guides: MDN." *Developer Guides | MDN*,
<https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX> (Accessed November 19, 2021)

- [11] Leaflet. (2010). "An Open-Source JavaScript Library for Interactive Maps."
<https://leafletjs.com/> (Accessed November 21, 2021)
- [12] Shaikh, Sohel S, and Vinod K Pachghare. "A Comparative Study of Database Connection Pooling Strategy." *International Research Journal of Engineering and Technology (IRJET)*, vol. 04, no. 05, May 2015, pp. 230–233.,
<https://doi.org/https://www.irjet.net/archives/V4/i5/IRJET-V4I539.pdf>. (Accessed November 10, 2021)
- [13] Ferretti, F. (2015). SharkPulse organization. Pulse Monitor. <http://sharkpulse.cnre.vt.edu/>
(Accessed September 2021.)
- [14] Ferretti, F. (2015). SharkPulse organization. Validation Monitor.
<http://sharkpulse.cnre.vt.edu/can-you-recognize/> (Accessed September 2021.)