Virginia Tech

Multimedia, Hypertext, and Information Access Capstone
(CS4624)

Blacksburg, VA 24061

Instructor: Dr. Edward Fox

Client: Dr. Francesco Ferretti

May 8, 2022

Shark Validator Game

Authors
Omar Kalbouneh
Madison Marshburn
David Carroll

Table of Contents

List of Figures

Executive Summary

The team was provided with a platform to update and implement new design changes to the original website for a project led by Dr. Francesco Ferretti, an Assistant Professor in the Department of Fish and Wildlife Conservation. His research interests include studying the impact of humanity on the Earth's oceans and conservation efforts. The website is built using WordPress with frontend CSS and HTML. RShiny provides the backend support to implement the Validation Monitor. We were tasked with converting the static framework of the website to a more dynamic and responsive framework and with improving on the current gamification scheme through a refined point rewards system and incentives.

The team was able to refine the current point reward system by integrating functionalities that will motivate users to validate sharks and support SharkPulse's research for the ecology and taxonomy of shark populations. These functionalities include awarding users who are able to recognize rare shark species, and if the shark species are labeled as "endangered" or "vulnerable" or "critically endangered" according to the IUCN red lists of ecosystems and threatened species. Most of the completed changes were on the backend for the validation monitor and on the frontend for the identification guide. The team has also improved the identification guide to be a more suitable web page to educate users about sharks. It was updated by adding more questions and adding an option to let the users select "I can't see" if they are unable to see shark characteristics from an image. Overall, the backend changes for assigning points based on if the user recognizes rare or threatened species is deployed. However, the project is still not complete, as the website still needs to be updated from being a static website to being a dynamic one. The rare species functionality could also be updated to improve the program's performance.

1 Introduction

SharkPulse is a mobile and web application designed to involve citizens in the documentation and monitoring of global wild shark populations. The main function of the application is to allow users to report shark sightings and to serve as a space for the image collection of shark sighting photos and videos in order to support research into the ecology and taxonomy of different shark populations [1]. The initiative also aims to support conservation research and increase public awareness of shark species.

According to the World Wildlife Fund, there are over 1,000 species of sharks and rays around the world that are threatened by overfishing, the illegal fishing of sharks for their fins, and accidentally getting caught in fishing gear set out for other types of fish [2]. These dangers are leading to the rapid loss of many species of sharks, which is why conservation and education efforts such as the SharkPulse initiative are so important.

Our team has been delegated the tasks of updating the website to display dynamic graphics and to improve the point system for the gamification of the site. This summer, Dr. Ferretti plans to host a big push event in an attempt to get the public more involved with SharkPulse. Our work will support this initiative, including making sure that the database tables have the capacity to deal with heavy transactional traffic. This should give users a reason to visit SharkPulse's website and play the game of validating sharks. The report will explain, in detail, what work was completed and what is currently in progress. It will also show the differences in the frontend and backend design for the old and new website.

Team Member Roles

Omar Kalbouneh: Tasked with coming up with new ideas for refining and improving the point system for validating sharks, gamifying the validation monitor, and assisting in updating the shark identification guide.

Madison Marshburn: Tasked with enhancing the frontend of the WordPress website by making figures and other elements dynamically connected to the database.

David Carroll: Tasked with improving the backend database portion of the website.

1.1 Requirements and Objectives

The objectives of this project are to dynamically update the frontend website and to provide better backend support for the database and image collection. We also aim to increase the gamification of the application to incentivize users to validate and interact with the shark images. This involves refining the point systems and identification guide schemes in order to track user progress. We have discussed these objectives with our client at length and determined the best course of action in completing them.

1.2 Client

Our client is Dr. Francesco Ferretti, an Assistant Professor in the Department of Fish and Wildlife Conservation. His research focuses on the alteration of marine ecosystems by human impact and the effect this has had on the ecology of these ecosystems. He has a particular interest in sharks and their relatives, which led him to begin the SharkPulse project [3].

1.3 Challenges

There were several challenges that we ran into as a team throughout the process of working on SharkPulse. The team was not familiar with the current stack being used. R Shiny and WordPress are both platforms that our team had no previous experience with. So, we were not able to have an immediate impact on the website because it took us time to learn the technology stack. Another challenge was that the validation monitor was crashing from too many connections between R and PostgreSQL, so we were not able to freely code in the backend R portion. Another challenge is that the code base is not designed using the best coding practices. For example, there is uncommented, complex code throughout the classes and file structure which makes it difficult to understand. Finally, there are also database obstacles involved in the gamification of the validation monitor. For example, figuring out how to extract data through R and then pass it to the HTML resubmission form to present it on the frontend, was a struggle.

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; see Figure 1. It is not required for users to be logged in to play the game, however, the users are provided with incentives to have an account and be logged in when validating sharks to level up by earning points (points in the old reward system) and various achievements and badges. When logging in, the users are presented with multiple options. They can either create an account with us or use their Google account to register.

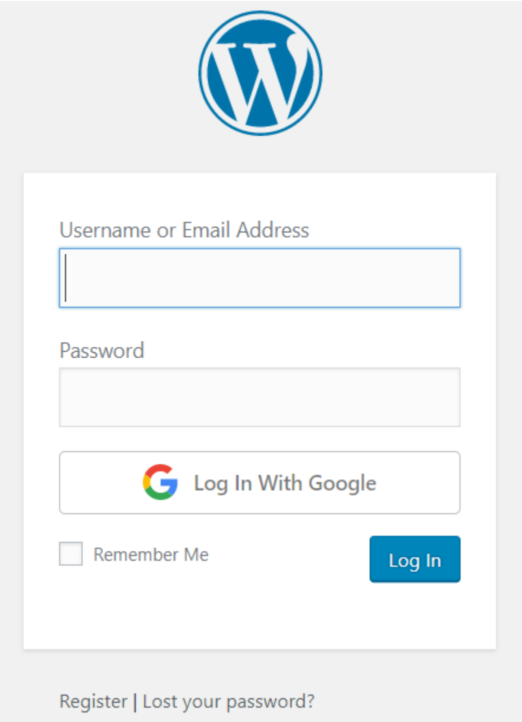


Figure 1: The User Login Interface.

3.2 Validation Monitor

The validation monitor is used to attract new users. The plan is to gamify it so that users can enjoy it and feel rewarded just like playing any other game. This is beneficial to the SharkPulse team because more images can be validated by humans. Below is an explanation of the monitor and what was changed on the user side. The next section explains about the backend changes.

- **How to use**

  First, when the user is logged in and clicks on the “validation monitor” button from the homepage, they are directed to a web page. The validation monitor is displayed and the user then chooses to start validating the images. The user does so by clicking on the blue bubble tags and then an image is displayed. This image is displayed next to some questions. The questions will ask about whether the user recognizes if the image is a shark, if the image is in an aquarium, the species name and the common name. The user will be assigned points depending on how many answers are inputted. It will also be assigned with regards to the new points system that will be explained later. In the case that the user is not sure about what the shark is, the user can take a look at the identification guide and figure out the shark. The identification guide was updated to have a “not sure” option and to present the user with more questions because we have found that some users did not have all the information required to answer the question. So, they ended up choosing a random answer. Further explanation will be presented in the backend section.

  It is not guaranteed that all of the images that the user gets are sharks, so the document will also demonstrate the different scenarios a user might face. Figure 3 demonstrates a scenario where the pop-up image is not a shark. The user will have to click “no” and then the data is sent to the machine learning algorithm.

3.2.1 Gamification

Users can earn points from validating sharks in several ways, as seen in Figure 3. If the users only identify if the image is a shark, 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. The additional functionalities that were implemented are:

1) The user is assigned 9 additional points if they are able to recognize rare species.
2) The user is assigned different points if they are able to recognize endangered species.

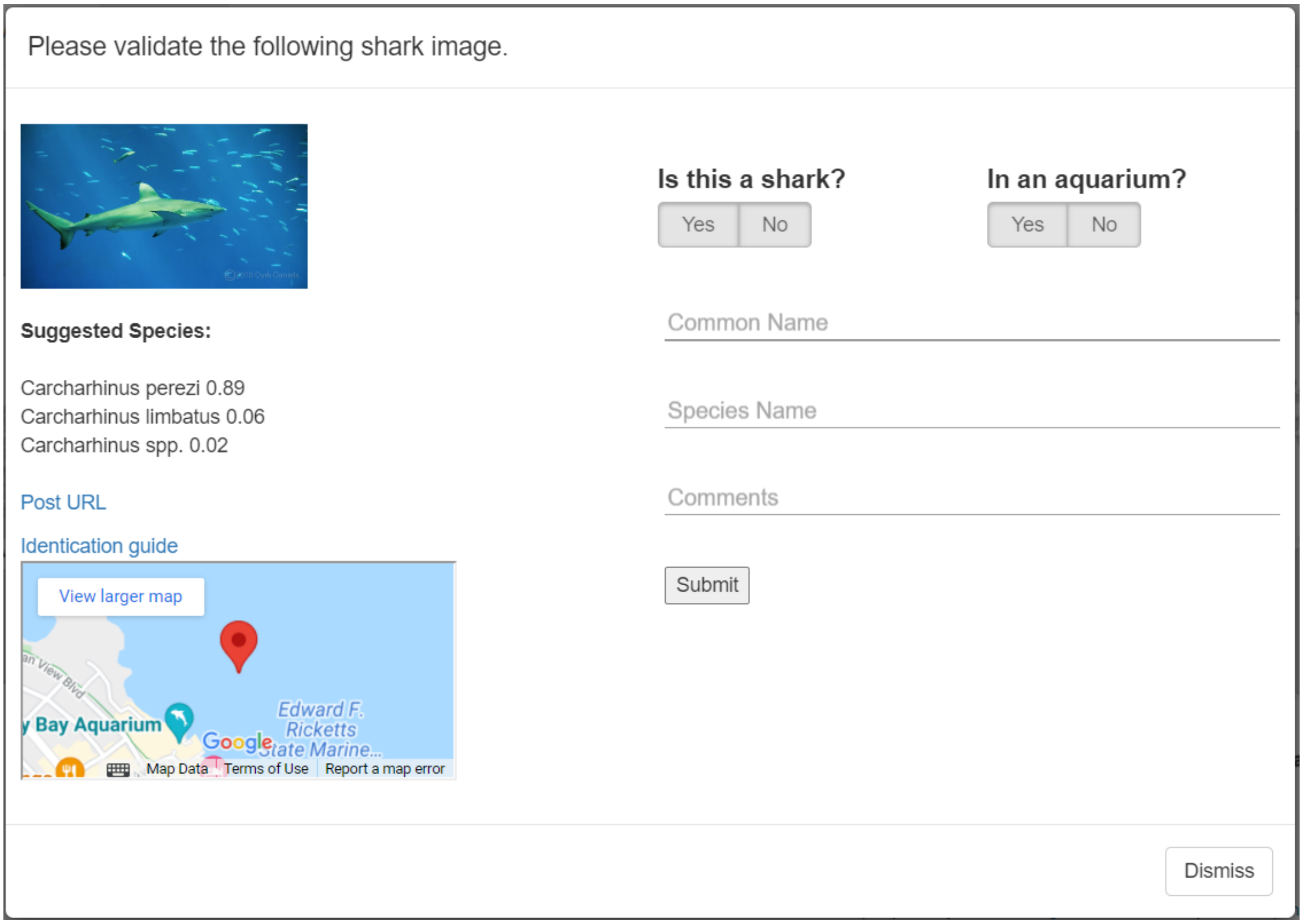
The endangered species is categorized into different brackets according to IUCN red lists of ecosystems and threatened species [6]:

- Critically Endangered: 12 points
- Endangered: 10 points
- Vulnerable: 9 points

3) The user will receive a boost of 40 points if they identify a rare and an endangered species.

Both of these functionalities require operations with PostgreSQL, R, and PHP. The rare functionality is implemented by first accepting a specific name for the species as an input from the user. Then the system will count the number of occurrences of the specific name of the species from the sharkpulse table (see Figure 15) in the database and then divide it by a baseline (total number of species in the database). This gives us an idea of whether the species is rare or not and helps us assign the correct number of points.

The functionality of recognizing endangered species is implemented in PHP. Detailed explanations of the implementation are in the developer's manual.



Please validate the following shark image.

Suggested Species:

Carcharhinus perezi 0.89
Carcharhinus limbatus 0.06
Carcharhinus spp. 0.02

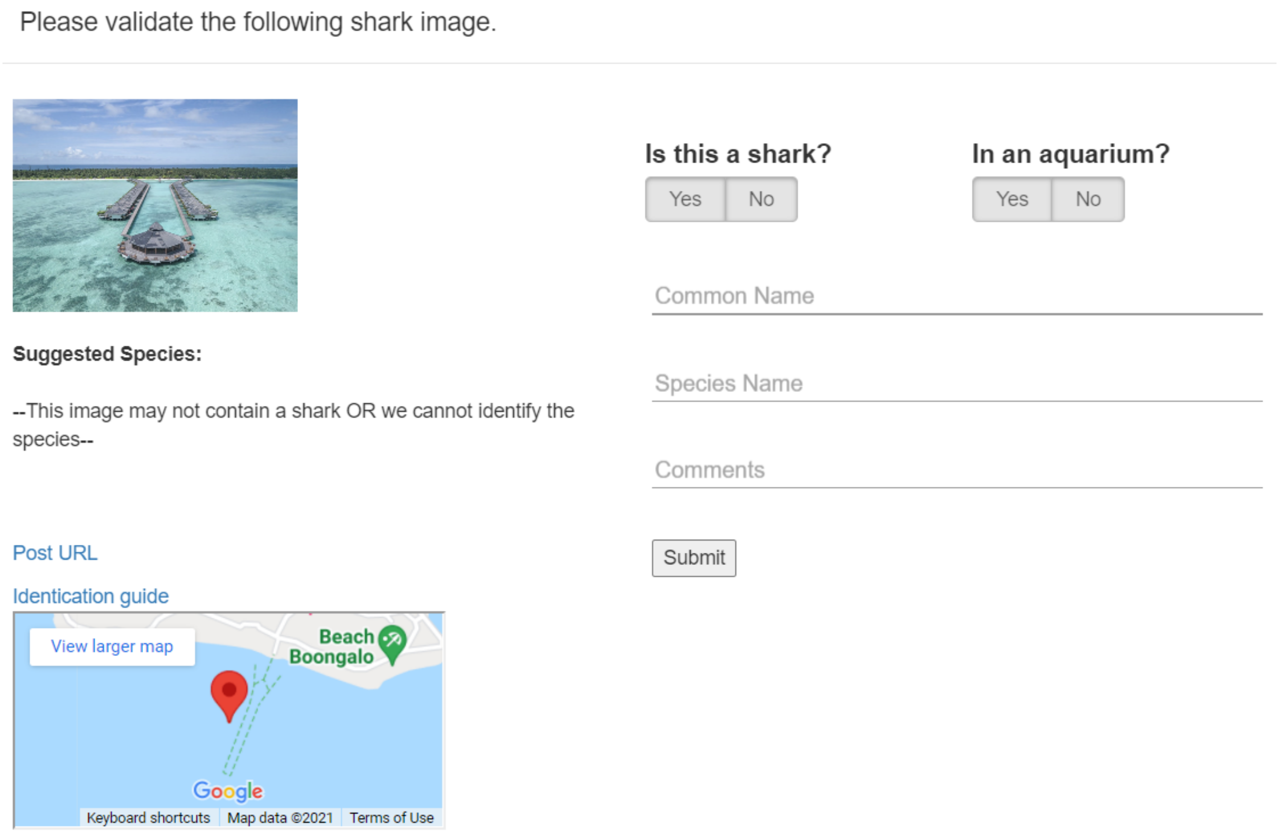
Post URL

Identication guide

Is this a shark?

In an aquarium?

Figure 2: Shark Validation Form (Shark In Image).



Please validate the following shark image.

Suggested Species:

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

Post URL

Identication guide

Is this a shark?

In an aquarium?

Common Name

Species Name

Comments

Figure 3: Shark Validation Form (No Shark In Image).

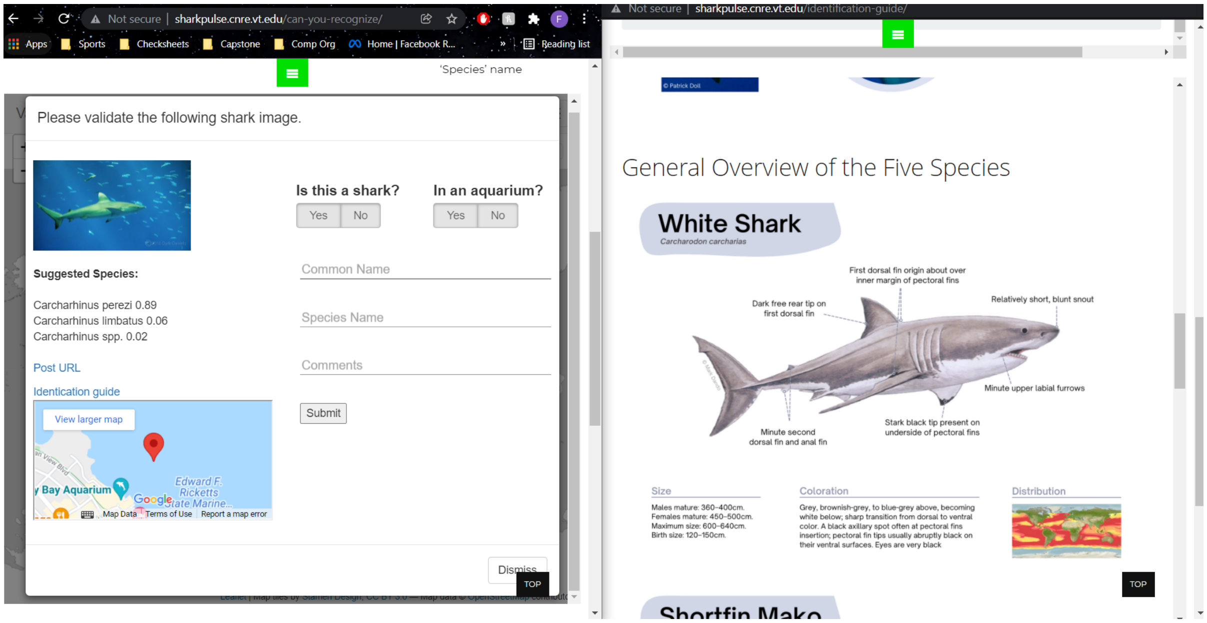
3.4 Identification Guide

The validation monitor is the most important tool for encouraging new users to visit the website and begin validating images. However, it is not enough to educate the users about sharks. Moreover, the new functionality for the validation monitor is hugely dependent on users to have a decent knowledge of shark species. So, how will new users, who are unfamiliar with sharks, be able to detect if the image includes rare shark species and fill in the species name from the top of their head? It is not a simple Google search. The answer for that is the Identification Guide.

The Identification Guide is offered as an option to every user when they are playing the Shark Validator Game. It serves to help guide users through validating a shark image, especially if the users are not familiar with shark species or if they reach a point that they may need assistance. Figure 4 shows how the users can optimize their validation experience by utilizing a split-screen mode with the Validation Monitor on one side, and the Identification Guide on the other, to actively learn more about the species on display in the image they are working on.

Figure 5 shows how users use the **old** version of the identification guide. If you look at the top left corner of Figure 5, the question asks the user if they recognize from the image whether the shark's mouth is in front of its eyes. Afterwards, if the user clicks yes, another "yes/no" question is presented. In the best-case scenario, the user would correctly answer the next question and then a shark species name will appear as shown in Figure 6. However, it is not a good practice to rely on best case scenarios. What if the user could not see if the shark's mouth is in front of its eyes in the image? Then, the user would be forced to guess the answer, which will make the user obtain an incorrect species name.

To remedy this, we have added functionality that allows users to choose a "I cannot see" option to move the guidance process along as shown in Figure 7. If the user clicks on the "I cannot see" option, then the identification guide will present alternative questions that were not present in the old guide. This approach provides flexibility to the user and narrows down the *correct* answers. In the rare case when the guide is unable to produce a concrete answer, it will list out multiple different species that the shark is likely to be, as displayed in Figure 6. By utilizing these various tools provided in the guide, users are able to stay engaged and fully interact with the validation game even if they were to require assistance in identifying a specific shark species. Implementation details will be explained in the developer's manual.


Not secure | sharkpulse.cnre.vt.edu/can-you-recognize/
Apps
Sports
Checksheets
Capstone
Comp Org
Home | Facebook R...
Reading list
'Species' name

Please validate the following shark image.

Is this a shark? Yes No

In an aquarium? Yes No

Suggested Species:

Carcharhinus perezi 0.89
Carcharhinus limbatus 0.06
Carcharhinus spp. 0.02

Common Name

Species Name

Comments

Post URL
Identication guide
View larger map
Bay Aquarium
Edward F. Ricketts State Marine...
Google
Map Data
Terms of Use
Report a map error
Submit
Dismiss
TOP
Not secure | sharkpulse.cnre.vt.edu/identification-guide/
© Patrick Doll

General Overview of the Five Species

White Shark
Carcharodon carcharias

First dorsal fin origin about over inner margin of pectoral fins
Dark free rear tip on first dorsal fin
Relatively short, blunt snout
Minute upper labial furrows
Stark black tip present on underside of pectoral fins
Minute second dorsal fin and anal fin

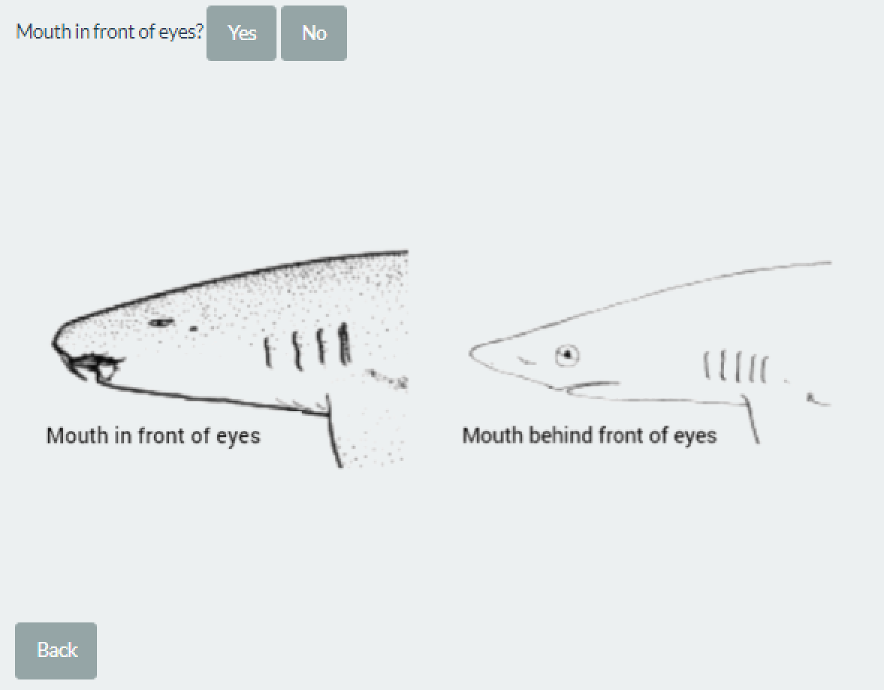
Size

Males mature: 360–400cm.
Females mature: 450–500cm.
Maximum size: 600–640cm.
Birth size: 120–150cm.

Coloration

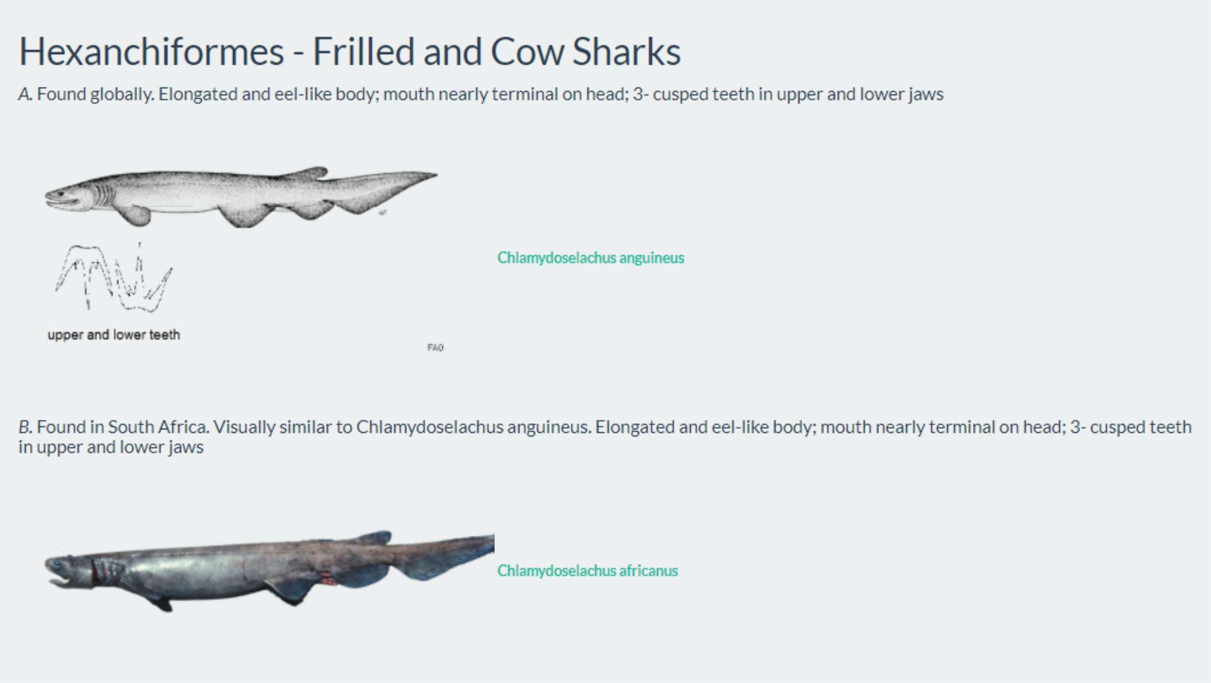
Grey, brownish-grey, to blue-grey above, becoming white below; sharp transition from dorsal to ventral color. A black axillary spot often at pectoral fins insertion; pectoral fin tips usually abruptly black on their ventral surfaces. Eyes are very black

Figure 4: Identification Guide (General Shark Species Information).



Mouth in front of eyes?

Figure 5: Old Identification Guide.



Hexanchiformes - Frilled and Cow Sharks

A. Found globally. Elongated and eel-like body; mouth nearly terminal on head; 3- cusped teeth in upper and lower jaws

B. Found in South Africa. Visually similar to Chlamydoselachus anguineus. Elongated and eel-like body; mouth nearly terminal on head; 3- cusped teeth in upper and lower jaws

Figure 6: Identification Guide Results.

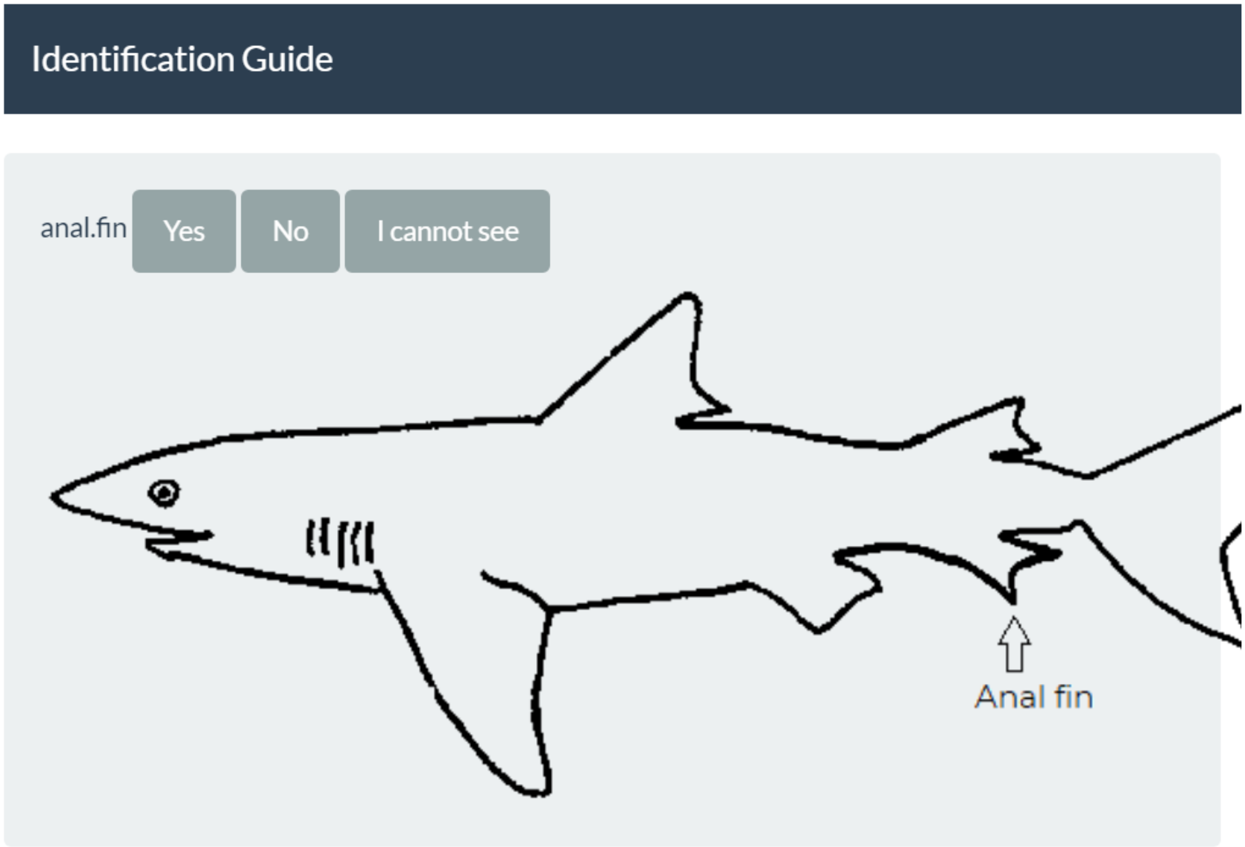


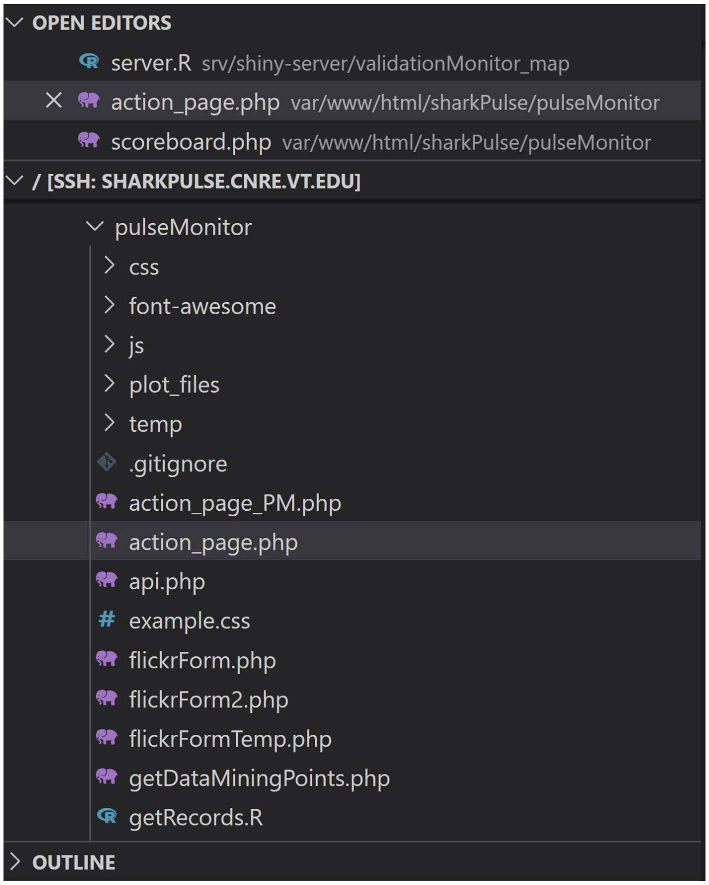
Figure 7: New Identification Guide.

4 Developer's Manual

4.1 Backend Design

- **Files Changed**

  The team's work was implemented through VScode and the file structure will be explained according to how they were accessed through VScode.
  Most of the validation monitor changes took place in action_page.php. Figure 8 shows the directories path to the updated files.

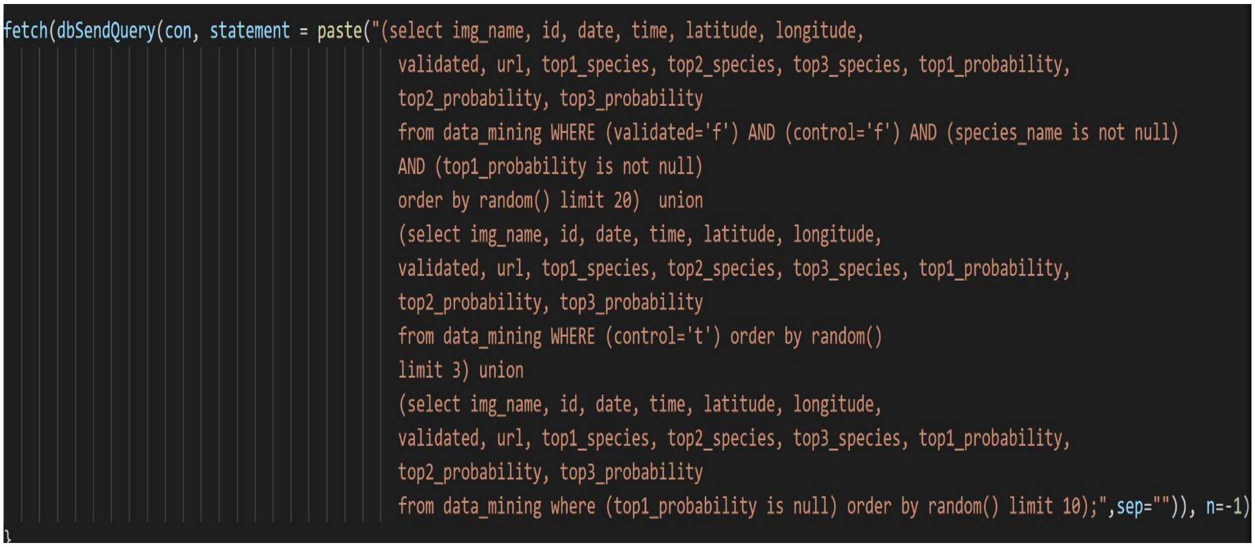


```
OPEN EDITORS
    server.R  srv/shiny-server/validationMonitor_map
  × action_page.php  var/www/html/sharkPulse/pulseMonitor
    scoreboard.php  var/www/html/sharkPulse/pulseMonitor
/ [SSH: SHARKPULSE.CNRE.VT.EDU]
    pulseMonitor
      > css
      > font-awesome
      > js
      > plot_files
      > temp
      .gitignore
      action_page_PM.php
      action_page.php
      api.php
      example.css
      flickrForm.php
      flickrForm2.php
      flickrFormTemp.php
      getDataMiningPoints.php
      getRecords.R
> OUTLINE
```

Figure 8: File Structure

- **Rare Functionality Implementation**

Implementing the functionality to recognize whether the user input for the species name is rare was originally planned to be integrated in the backend file, server.R shown at the top of Figure 8. The plan was that server.R would interact with the database and then calculate whether the species is rare or not. However, the team has been experiencing issues on the validation monitor because there are too many connections between Postgresql and R. So, adding another query to fetch data from the database is not scalable/ideal. Figure 9 shows an example of a fetch request in server.R.



```
fetch(dbSendQuery(con, statement = paste("(select img_name, id, date, time, latitude, longitude,
                                           validated, url, top1_species, top2_species, top3_species, top1_probability,
                                           top2_probability, top3_probability
                                           from data_mining WHERE (validated='f') AND (control='f') AND (species_name is not null)
                                           AND (top1_probability is not null)
                                           order by random() limit 20)  union
                                           (select img_name, id, date, time, latitude, longitude,
                                           validated, url, top1_species, top2_species, top3_species, top1_probability,
                                           top2_probability, top3_probability
                                           from data_mining WHERE (control='t') order by random()
                                           limit 3) union
                                           (select img_name, id, date, time, latitude, longitude,
                                           validated, url, top1_species, top2_species, top3_species, top1_probability,
                                           top2_probability, top3_probability
                                           from data_mining where (top1_probability is null) order by random() limit 10);",sep="")), n=-1)
}
```

Figure 9: Fetch request in R

In order to implement this functionality without overloading the connections, the team has decided to implement the functionality in action_page.php instead of server.R. action_page.php is the frontend file (HTML form submission) for the validation monitor.

Throughout the file, the functionality was implemented by first checking if the user input for the species name and/or the common name exists (not null). If it does, then a query is sent to the sharkpulse (see Figure 15) table to count:

1) The number of occurrences of the species/common name in the sharkpulse table in the database.

2) The total number of species names that exist in the sharkpulse table.

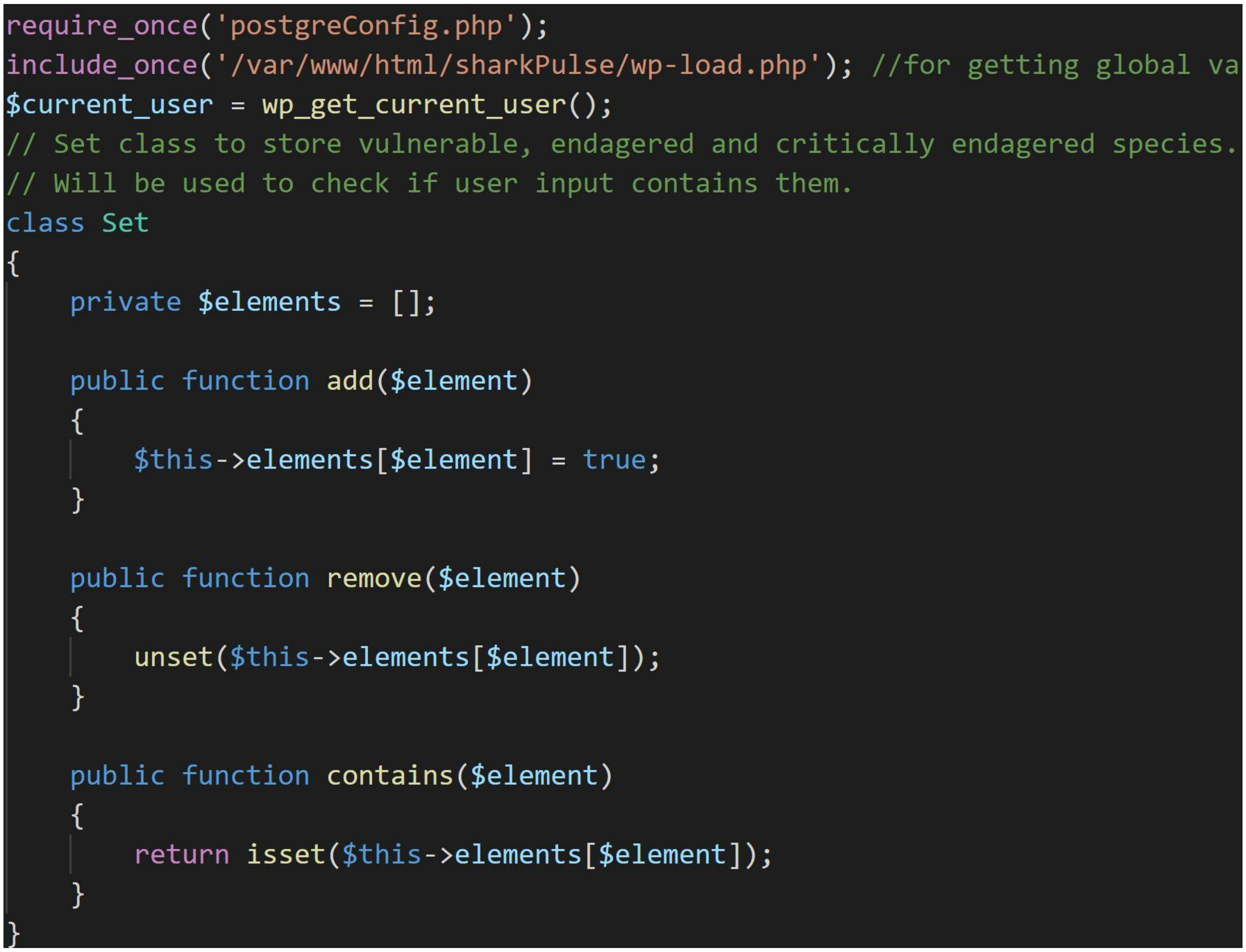
Figure 15 shows the tables inside the database. The validated images from the database from the users are stored in the datamining table. The validated images that have been double checked by the SharkPulse team are stored in the sharkpulse table. We have chosen to count the number of entries in the sharkpulse table instead of the datamining table to have more authentic and precise values.

Both of these values are retrieved and then the number of occurrences of the user input is divided by the total number of species that exist in the table. Afterwards, this number is compared against a certain threshold that can be adjusted by the developer. The threshold is set at 30% for now. Anything less than 30% is considered rare and therefore the user will gain an additional 9 points. This will happen by retrieving the user from the userbase table by their email and updating the row entry by adding 9 points. We also thought about the edge case of the user trying to cheat the rules of the game by inputting a word that does exist in the database. To solve this case the functionality checks that user input must exist in the database to be awarded points.

- **Endangered Species Functionality**

The endangered functionality (explained in the user manual), was originally planned to also be implemented in server.R by integrating a package called “rredlist”, which assists in categorizing the shark species according to the IUCN red lists of ecosystems and threatened species [6]. However, due to the fact that the SharkPulse team believes that the validation monitor crashes are directly related to too many requests between R and PostgreSQL, we adopted a similar strategy to the rare functionality explained above.

The team took an approach to build an in-memory data structure to store the vulnerable, endangered and critically endangered species inside action_page.php. Figure 10 shows the implementation for the data structure.



```
require_once('postgreConfig.php');
include_once('/var/www/html/sharkPulse/wp-load.php'); //for getting global va
$current_user = wp_get_current_user();
// Set class to store vulnerable, endagered and critically endagered species.
// Will be used to check if user input contains them.
class Set
{
    private $elements = [];

    public function add($element)
    {
        $this->elements[$element] = true;
    }

    public function remove($element)
    {
        unset($this->elements[$element]);
    }

    public function contains($element)
    {
        return isset($this->elements[$element]);
    }
}
```

Figure 10: Set structure

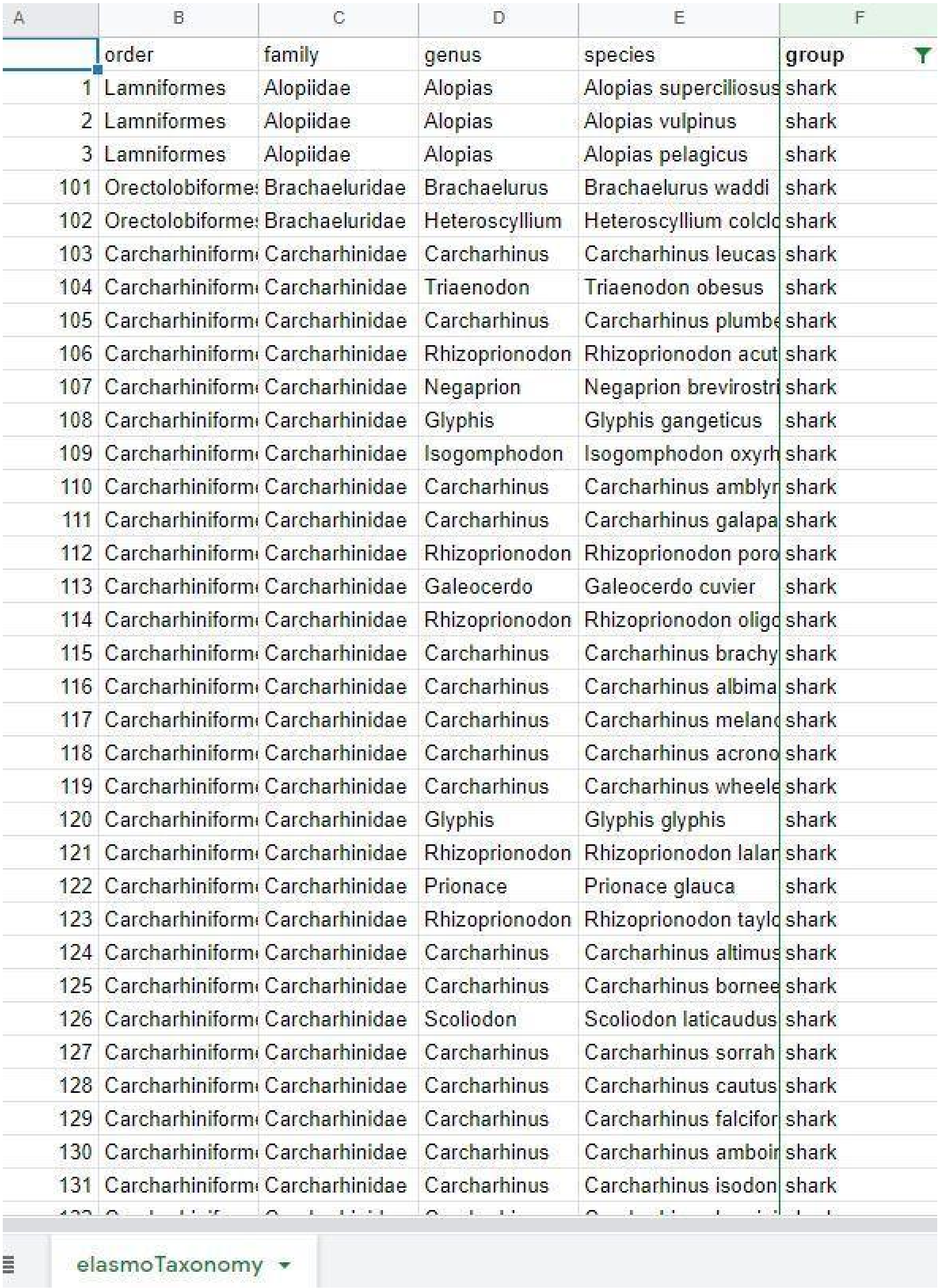
The implementation utilizes an object oriented structure and builds on top of existing functionality for sets in PHP. The structure only includes three methods, add, remove, and contains. The reason we chose to implement it, is because we wanted future developers, who might not be familiar with the built-in functionality of sets in PHP, to be able to work and understand our design. The plan is to assist developers from all backgrounds to understand our additions.

We have initialized three different sets: vulnerable, endangered, and critically endangered. Then we filled these lists with the species obtained from the IUCN red lists of ecosystems and threatened species [6].

After the user input is retrieved from the backend and sent by a POST request to the action_page.php, the user input will then be checked if it exists in one of the three sets by calling the *contains(element)* method. The time complexity for this operation is O(1) which achieves the best performance. If the method returns true, then the user will gain more points by updating the userbase table in the database.

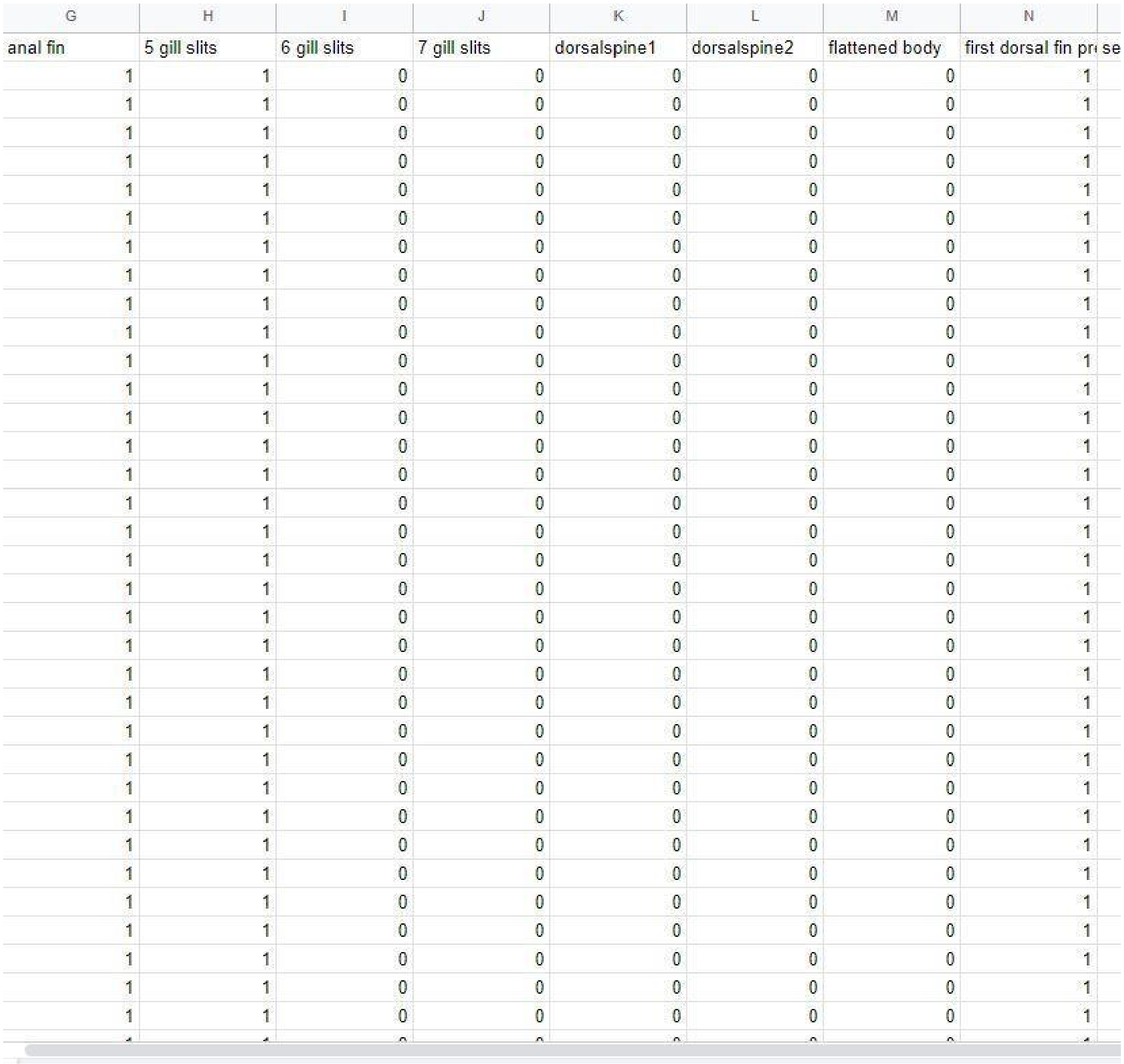
- **Identification guide**

  As explained in the user manual, the ultimate goal is to present the user with the maximum assistance for a decision. Figure 11 and Figure 12 display a spreadsheet table containing over 500 entries for shark species, where each column is a physical shark characteristic. In Figure 12, “1” denotes that the shark includes this characteristic and “0” denotes that the shark does not include this characteristic. For example, in column G, any shark that has “1” there means that the shark has an anal fin. In the case when the user can see the anal fin, the guide filters and chooses all of the results that contain a “1”. Then, specific questions are asked about how many slits the user can see. If the user is not able to count the number of slits from the image, new questions will be asked about other characteristics present in the columns. This was implemented by utilizing RShiny to develop a new script. Figure 13 shows the code for parsing the spreadsheet and building the UI for the guide. Lines 5-8 parse through the CSV file and lines 12 - 46 implement the headers for the GUI. The logic behind Figure 13 and 14 is that the webpage presents the user with shark characteristics questions. Then, if the user answers “yes” or “no”, the script will parse the CSV file and extract the entries that match the physical characteristics by checking if the column is equal to 0 or 1. If the number of extracted entries is less than 10, they would be presented to the user (usually the number of entries is less than 10). In the case, where the user selects “I cannot see”, the question will be replaced with an alternative question for the shark characteristic.



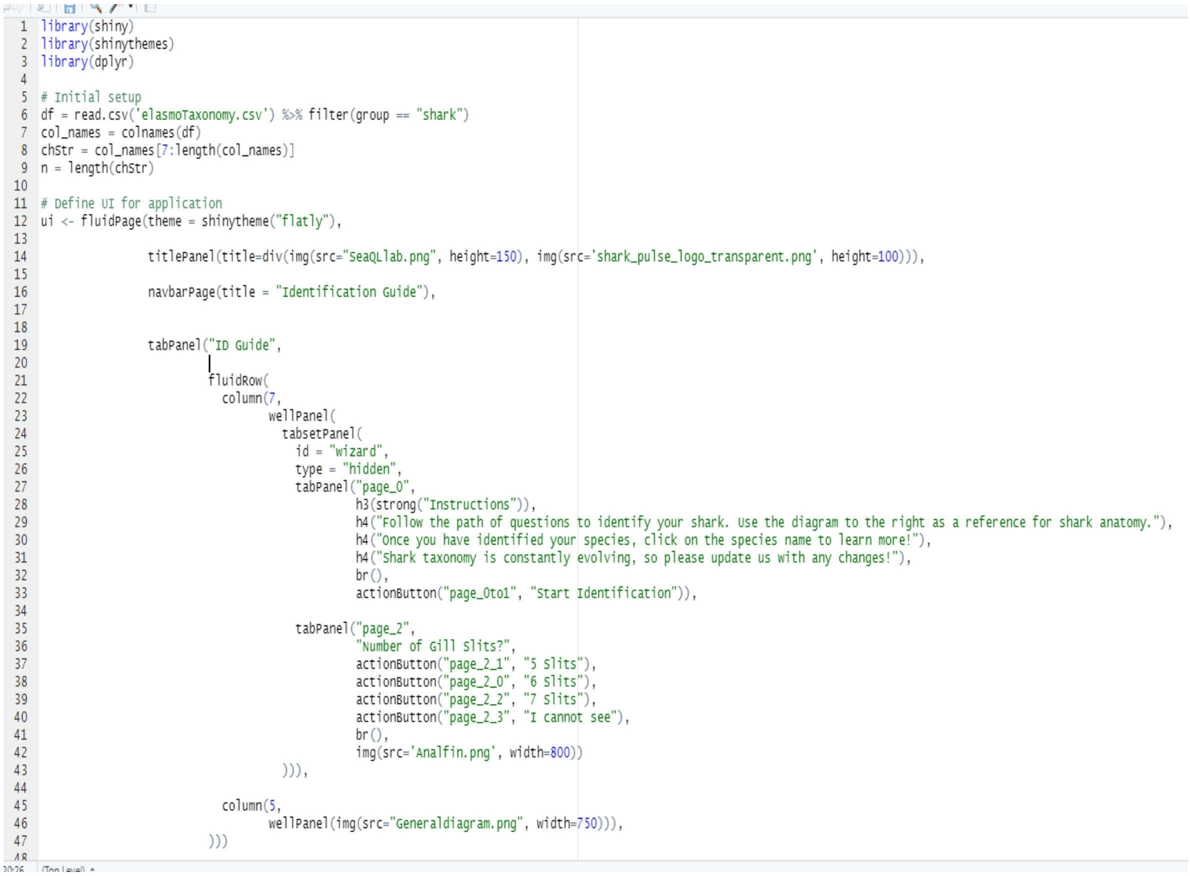
A	B	C	D	E	F
	order	family	genus	species	group
1	Lamniformes	Alopiidae	Alopias	Alopias superciliosus	shark
2	Lamniformes	Alopiidae	Alopias	Alopias vulpinus	shark
3	Lamniformes	Alopiidae	Alopias	Alopias pelagicus	shark
101	Orectolobiforme:	Brachaeluridae	Brachaelurus	Brachaelurus waddi	shark
102	Orectolobiforme:	Brachaeluridae	Heteroscyllium	Heteroscyllium colclo	shark
103	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus leucas	shark
104	Carcharhiniform	Carcharhinidae	Triaenodon	Triaenodon obesus	shark
105	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus plumbe	shark
106	Carcharhiniform	Carcharhinidae	Rhizoprionodon	Rhizoprionodon acut	shark
107	Carcharhiniform	Carcharhinidae	Negaprion	Negaprion brevirostri	shark
108	Carcharhiniform	Carcharhinidae	Glyphis	Glyphis gangeticus	shark
109	Carcharhiniform	Carcharhinidae	Isogomphodon	Isogomphodon oxyrh	shark
110	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus amblyr	shark
111	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus galapa	shark
112	Carcharhiniform	Carcharhinidae	Rhizoprionodon	Rhizoprionodon poro	shark
113	Carcharhiniform	Carcharhinidae	Galeocerdo	Galeocerdo cuvier	shark
114	Carcharhiniform	Carcharhinidae	Rhizoprionodon	Rhizoprionodon oligo	shark
115	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus brachy	shark
116	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus albima	shark
117	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus melanc	shark
118	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus acrono	shark
119	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus wheele	shark
120	Carcharhiniform	Carcharhinidae	Glyphis	Glyphis glyphis	shark
121	Carcharhiniform	Carcharhinidae	Rhizoprionodon	Rhizoprionodon lalar	shark
122	Carcharhiniform	Carcharhinidae	Prionace	Prionace glauca	shark
123	Carcharhiniform	Carcharhinidae	Rhizoprionodon	Rhizoprionodon taylo	shark
124	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus altimus	shark
125	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus bornee	shark
126	Carcharhiniform	Carcharhinidae	Scoliodon	Scoliodon laticaudus	shark
127	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus sorrah	shark
128	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus cautus	shark
129	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus falcifor	shark
130	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus amboir	shark
131	Carcharhiniform	Carcharhinidae	Carcharhinus	Carcharhinus isodon	shark

Figure 11: Spreadsheet 1



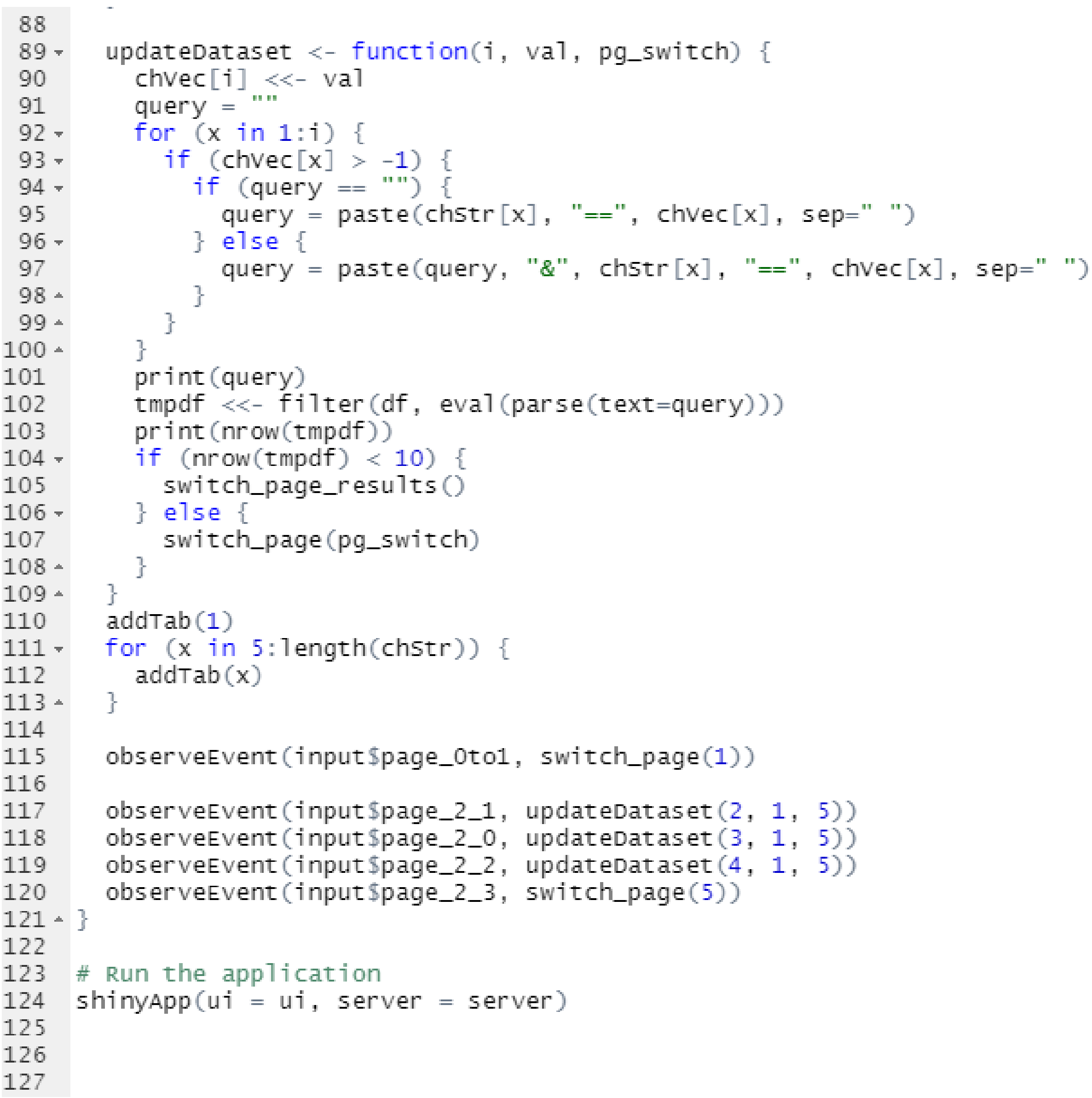
G	H	I	J	K	L	M	N	
anal fin	5 gill slits	6 gill slits	7 gill slits	dorsalspine1	dorsalspine2	flattened body	first dorsal fin pr	se
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	
1	1	0	0	0	0	0	1	

Figure 12: Spreadsheet 2



```r
library(shiny)
library(shinythemes)
library(dplyr)

# Initial setup
df = read.csv('elasmoTaxonomy.csv') %>% filter(group == "shark")
col_names = colnames(df)
chStr = col_names[7:length(col_names)]
n = length(chStr)

# Define UI for application
ui <- fluidPage(theme = shinytheme("flatly"),

                titlePanel(title=div(img(src="SeaQLlab.png", height=150), img(src='shark_pulse_logo_transparent.png', height=100))),

                navbarPage(title = "Identification Guide"),


                tabPanel("ID Guide",

                         fluidRow(
                           column(7,
                                  wellPanel(
                                    tabsetPanel(
                                      id = "wizard",
                                      type = "hidden",
                                      tabPanel("page_0",
                                               h3(strong("Instructions")),
                                               h4("Follow the path of questions to identify your shark. Use the diagram to the right as a reference for shark anatomy."),
                                               h4("Once you have identified your species, click on the species name to learn more!"),
                                               h4("Shark taxonomy is constantly evolving, so please update us with any changes!"),
                                               br(),
                                               actionButton("page_0to1", "Start Identification")),

                                      tabPanel("page_2",
                                               "Number of Gill Slits?",
                                               actionButton("page_2_1", "5 Slits"),
                                               actionButton("page_2_0", "6 Slits"),
                                               actionButton("page_2_2", "7 Slits"),
                                               actionButton("page_2_3", "I cannot see"),
                                               br(),
                                               img(src='Analfin.png', width=800))
                                    ))),

                           column(5,
                                  wellPanel(img(src="Generaldiagram.png", width=750))),
                         )))
```

Figure 13: Identification Guide script 1

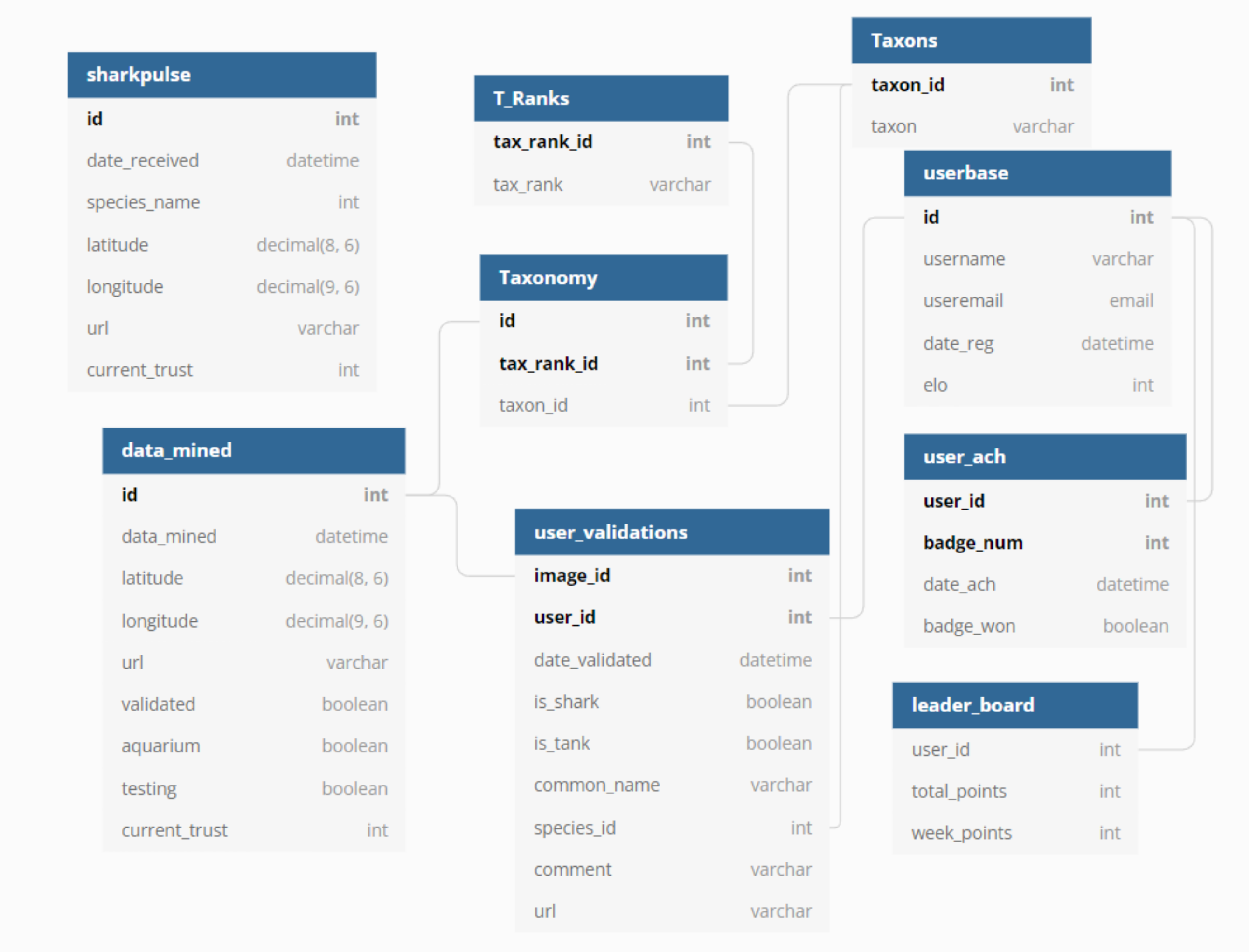


```
  updateDataset <- function(i, val, pg_switch) {
    chVec[i] <<- val
    query = ""
    for (x in 1:i) {
      if (chVec[x] > -1) {
        if (query == "") {
          query = paste(chStr[x], "==", chVec[x], sep=" ")
        } else {
          query = paste(query, "&", chStr[x], "==", chVec[x], sep=" ")
        }
      }
    }
    print(query)
    tmpdf <<- filter(df, eval(parse(text=query)))
    print(nrow(tmpdf))
    if (nrow(tmpdf) < 10) {
      switch_page_results()
    } else {
      switch_page(pg_switch)
    }
  }
  addTab(1)
  for (x in 5:length(chStr)) {
    addTab(x)
  }

  observeEvent(input$page_0to1, switch_page(1))

  observeEvent(input$page_2_1, updateDataset(2, 1, 5))
  observeEvent(input$page_2_0, updateDataset(3, 1, 5))
  observeEvent(input$page_2_2, updateDataset(4, 1, 5))
  observeEvent(input$page_2_3, switch_page(5))
}

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

Figure 14: Identification Guide server logic

4.2 Database Changes

As seen in Figure 15, in the leaderboard table (on the bottom right), the team added a “week_points” column to record the points accumulated for the past week to get the highest ranked users. All of the mined records from the new functionality are stored in the database table data_mined. This table includes all of the identifying information about the images, such as their location and the species type so that the image may be readily located in the future if needed. The userbase table contains the identifying information for all the users of the SharkPulse platform. It stores the user input from the Pulse Monitor and Validation Monitor to validate the images.



sharkpulse

id	int
date_received	datetime
species_name	int
latitude	decimal(8, 6)
longitude	decimal(9, 6)
url	varchar
current_trust	int

T_Ranks

tax_rank_id	int
tax_rank	varchar

Taxons

taxon_id	int
taxon	varchar

userbase

id	int
username	varchar
useremail	email
date_reg	datetime
elo	int

Taxonomy

id	int
tax_rank_id	int
taxon_id	int

data_mined

id	int
data_mined	datetime
latitude	decimal(8, 6)
longitude	decimal(9, 6)
url	varchar
validated	boolean
aquarium	boolean
testing	boolean
current_trust	int

user_ach

user_id	int
badge_num	int
date_ach	datetime
badge_won	boolean

user_validations

image_id	int
user_id	int
date_validated	datetime
is_shark	boolean
is_tank	boolean
common_name	varchar
species_id	int
comment	varchar
url	varchar

leader_board

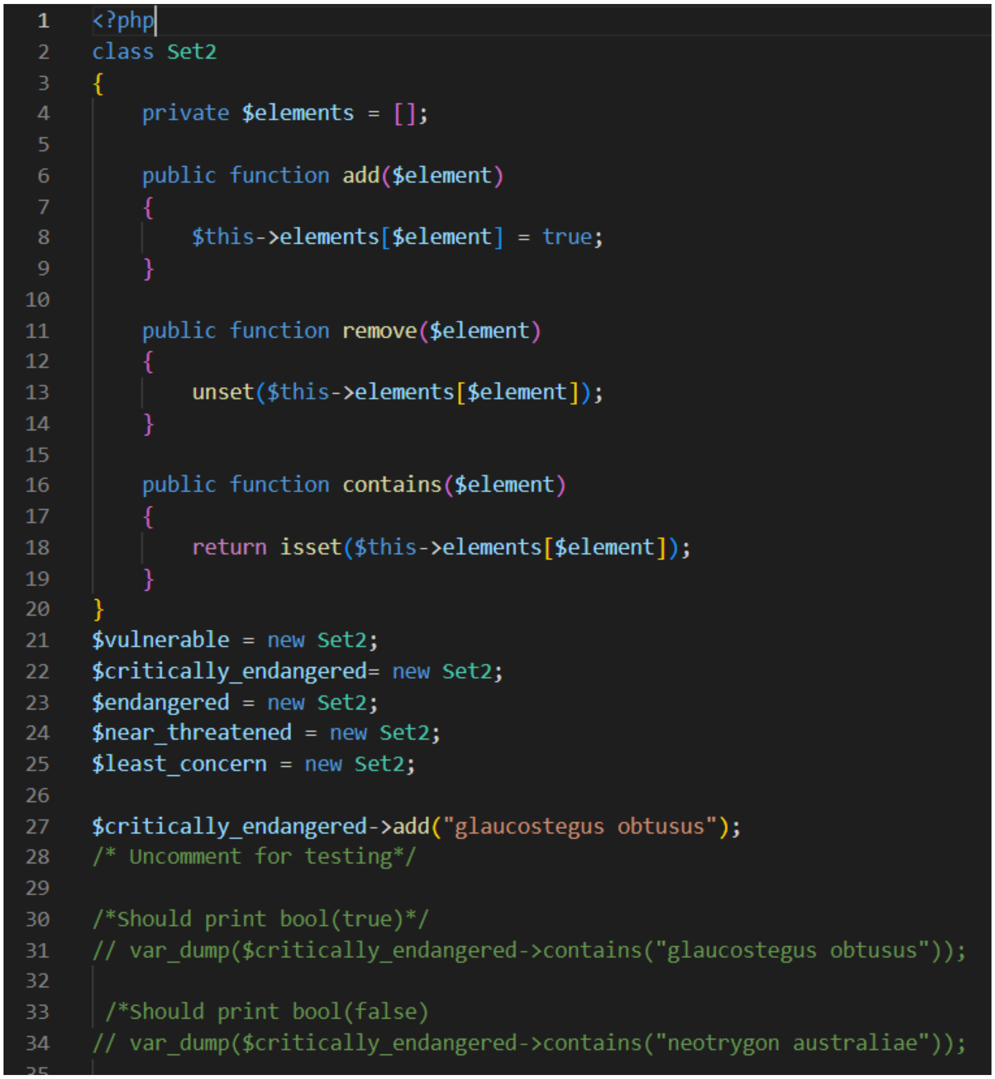
user_id	int
total_points	int
week_points	int

Figure 15: Database table design.

5 Implementation

- **Testing**

The number of points/points a user gains from the new functionality was manually checked from the leaderboard or database when validating a rare and/or an endangered species. The identification guide was manually tested by running the new script in Rstudio and verifying that we get the expected results. Figure 16 is a testing script that tests the Set data structure that was written for the endangered species functionality. Descriptive comments are included to help the developer recognize if the Set functionality is working.

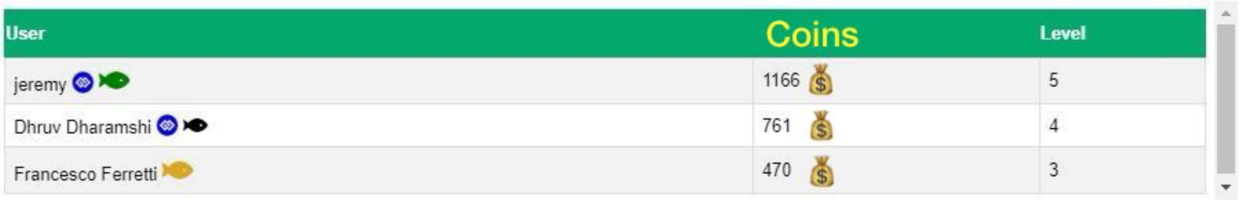


```php
<?php
class Set2
{
    private $elements = [];

    public function add($element)
    {
        $this->elements[$element] = true;
    }

    public function remove($element)
    {
        unset($this->elements[$element]);
    }

    public function contains($element)
    {
        return isset($this->elements[$element]);
    }
}
$vulnerable = new Set2;
$critically_endangered= new Set2;
$endangered = new Set2;
$near_threatened = new Set2;
$least_concern = new Set2;

$critically_endangered->add("glaucostegus obtusus");
/* Uncomment for testing*/

/*Should print bool(true)*/
// var_dump($critically_endangered->contains("glaucostegus obtusus"));

 /*Should print bool(false)
// var_dump($critically_endangered->contains("neotrygon australiae"));
```

Figure: 16: Script for testing Set

6 Future Work & Projections

An update to the current implementation of finding the rare functionality is to compute the values with a daily cron job. We can add two columns in the sharkpulse table. One column would record the distinct species name in the table and the other column would record the count of the species in the table. At the end of the column, a row could be added to record the total number of species names. This way, only two values would be needed to be fetched and then divided. This could improve the performance of the website. Another addition that can be made to the Validation Monitor is to fully integrate a timer mode into the game. The time-based mode would rank the players based on who validated the shark images within a specific amount of time (for example, 15 minutes). This would encourage people to actively participate in validation monitor games. Additional functionalities that we did not have time to implement is providing the highest ranked user within a specific number of days (for example, getting the highest scoring user for the past week or month). As in Figure 15, only a table called "week_points" was added to the leader_board, but the backend infrastructure was not built to support it. The leaderboard could also show the highest scoring users for the past week instead of the all-time high users. This can motivate users to validate images because they will have a chance to see their name on the scoreboard and compete with family and friends. We only designed a wireframe for a new leaderboard, but did not have time to update the code. Figure 17 shows the new frontend wireframe. Instead of rewarding users with points, rewarding them with "Coins" will make them feel rich and prompt them to start validating.



User	Coins	Level
jeremy	1166	5
Dhruv Dharamshi	761	4
Francesco Ferretti	470	3

Figure 17. New Leaderboard sorted by points

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

Graduate Student Expert: Jeremy Jenrette, CS4624 Spring 2021, jjeremy1@vt.edu

Undergraduate Student Expert: Aman Kothari, CS4624 Fall 2021, amank@vt.edu

References

[1] Ferretti, F. (2015). What is SharkPulse? http://sharkpulse.cnre.vt.edu/ (Accessed April 2022).

[2] World Wildlife Fund. (2022). Species, Shark. https://worldwildlife.org/species/shark#:~:text=risk%20of%20extinction-,More%20than%20one-third%20of%20all%20sharks%2C%20rays%2C%20and,Threatened%20Species%20extinction%20risk%20status. (Accessed April 2022).

[3] Virginia Tech. (2022). Assistant Professor Francesco Ferretti. https://fishwild.vt.edu/faculty/ferretti.html (Accessed April 2022).

[4] Kothari, A., Patel, F., Raya, R., Shroff, T., Tiwari, A. (2021). VTechWorks. CS4624 team term project, December 2021, Virginia Tech, Blacksburg, VA, http://hdl.handle.net/10919/103254 (Accessed April 2022).

[5] 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 April 2022).

[6] IUCN. 2021. The IUCN Red List of Threatened Species. Version 2021-3. https://www.iucnredlist.org. Accessed on April 2022.