SHARK VALIDATION MONITOR

Authors
J. Jenrette
G. Chang
S. Gordon
M. Mulgrew
H. Debay

May 12, 2021
# Table of Contents

- **Table of Figures**  
  2

- **Table of Tables**  
  3

- **Executive Summary**  
  4

1 **Introduction**  
  5
  1.1 Requirements and Objectives  
    - Shiny platform  
    - Media Pipeline  
    - Gaming  
    - Back-end  
  1.2 Client  
  1.3 Challenges  

2 **Design**  
  8
  2.1 Methodology  
    - User type  
    - Tasks flowchart  
    - Specific services  
  2.2 Citizen Scientists as Users  
  2.3 Web server and PostgreSQL database  
  2.4 Shiny App  
  2.5 Project Plan and Timeline  

3 **Implementation**  
  15
  3.1 Testing  
  3.2 Evaluation and Assessment  

4 **User Manual**  
  16
  4.1 User Login  
  4.2 Validation and Scoring  
  4.3 Other Functionality  

5 **Developer's Manual**  
  23
  5.1 ELO Ratings and User Validations  
  5.2 Database  
  5.3 Display  

6 **Lessons Learned**  
  28

**Acknowledgements**  
  29

**References**  
  30
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gaming user logical task flow</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Knowledge user logical task flow</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>Image sourced from Facebook and validated by a citizen scientist as a Greenland shark.</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>Columns of data mining table in the Pelagic database.</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Main page for the Validation Monitor constructed from R Shiny and hosted with the Shiny server software package.</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>HTML form written in R Shiny to display image, date, identification app hyperlink, and radio/text questions.</td>
<td>14</td>
</tr>
<tr>
<td>7</td>
<td>Realized timeline for developing a shark validation monitor Shiny App within the sharkPulse webserver.</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>Current link to allow the user to login; Link embedded in the text “Login with Google to keep score!”</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>Page after clicking to login, “Log In With Google” is used for Google related login services; the Username/Password shown is for WordPress related login.</td>
<td>17</td>
</tr>
<tr>
<td>10</td>
<td>Validation point to be marked as &quot;No&quot;; Not a shark in the wild.</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>Information displayed after submitting a &quot;No&quot; response for a validation point.</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>Current scheme to award signed-in users with points depending on the action taken when validating.</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>Current leader-board totals with user jeremy as the leader with 222 VP and rank 2.</td>
<td>19</td>
</tr>
<tr>
<td>14</td>
<td>Form for uploading details of a shark sighting including the photo file, species name, location, date, and email.</td>
<td>20</td>
</tr>
<tr>
<td>15</td>
<td>Pulse Monitor map displaying details of a shark sighting contained in the validated database (date, common/scientific name, and source obtained).</td>
<td>21</td>
</tr>
<tr>
<td>16</td>
<td>Pulse Monitor data table displaying results from filtering for the date &quot;1/23/2016&quot;</td>
<td>22</td>
</tr>
<tr>
<td>17</td>
<td>sharkPulse identification app</td>
<td>22</td>
</tr>
<tr>
<td>18</td>
<td>Code snippet from action_page.php that is used to generate a thank you page upon submission of a validation point and contains relevant information about the submission.</td>
<td>25</td>
</tr>
<tr>
<td>19</td>
<td>Weekly Flickr query for the tag shark.</td>
<td>26</td>
</tr>
<tr>
<td>20</td>
<td>Code snippet from userbase.php that is used to update the players rank icon.</td>
<td>27</td>
</tr>
<tr>
<td>21</td>
<td>Code snippet from scoreboard.php that is used to generate the leader-board.</td>
<td>28</td>
</tr>
<tr>
<td>22</td>
<td>Code snippet that is used to update the user’s profile every 15 seconds.</td>
<td>28</td>
</tr>
</tbody>
</table>
Table of Tables

Table 1: Roles for each team member .................................................. 6
Table 2: Progression of Monitor services ................................................. 11
Table 3: Database tables and PHP action scripts ...................................... 24
Executive Summary

Abundance and observation data of global shark populations is necessary for effective conservation and management. While there are operative direct methods to retrieve this data from scientific surveys and fisheries monitoring, these methods can be convoluted, costly, and focused more on commercially relevant species such as billfish, tuna, and cod rather than sharks (Dulvy et al., 2008). Fortunately, there is an untapped plethora of data with potential to provide distribution and abundance data of shark species. This data can be mined from massive social media clouds such as Flickr or from news and blog posts, ecological sites and articles, and other forms of online media.

Available to the public, these clouds of data can be mined, filtered, and presented on the website, sharkPulse.org (Ferretti, 2015). SharkPulse is a web application and database created to warehouse global shark sightings. To establish public involvement, this project deals with the user interactive platform of sharkPulse, the Validation Monitor. The Monitor is a global map display of records that have been geolocated and timestamped, but have yet to be validated as genuine shark sightings. Our project, and aims for the Monitor, involve i) increasing public awareness, ii) educational opportunities, and iii) the spatiotemporal resolution of shark species.

To do this, we have designed many elements that compliment the Web server, database, and dynamic website that sharkPulse is built on. We have incorporated a Shiny App server which is a tool for building interactive web pages in HTML and JavaScript, but is written in R (R Core Team, 2020; RStudio Inc., 2013). Shiny uses port forwarding and iframes to display the app in WordPress (WordPress, 2021). We use Elementor as a page-editing platform to facilitate the app (Elementor, 2021). Login to keep score takes users to the sites login page. Users may login to the Validation Monitor page via Google Login or with WordPress registration. Once logged in, a user can begin validating images they believe to be sharks or non-sharks. Within the Shiny App, a global map displays popup balloons. Currently, each balloon represents an image scraped from Flickr, a media-sharing social network (Butterfield & Fake, 2021). The images are stored in the sharkPulse server and are displayed in the app by querying the local PostgreSQL database, Pelagic. Among other things, this database contains a data mining table which stores the Flickr image URLs in a column. The app can specify an HTML form upon clicking the map balloons, which renders the image URLs along with some questions regarding that image.

To entice users and motivate them to stick with the validation process, this team has incorporated a gaming aspect. We want the users to control their level of play within the Monitor. A ranking and point system is dependent on their involvement. Discussing with our client, we have identified the rules of this game, how to implement them, and a sound baseline for new users. We developed the back and front-end of the Shiny app to accommodate for a competitive or relaxed gaming style.
1 Introduction

Archiving records can be a tedious chore for any area of research. In conservation and ecology, records often consist of spatiotemporal, taxonomic, and user-appended metadata. The goal of sharkPulse is automating archiving of Elasmobranchii media for better defining population characteristics for conservation and management. Currently, sharkPulse houses 32,586 photo records across 367 species; more than 25% of them are labeled vulnerable or endangered to some degree. SharkPulse is composed of sighting statistics, identification apps, picture submissions, social media presence, and publications that have benefited from sharkPulse’s efforts. To facilitate filtering out unrelated media, sharkPulse has also incorporated machine learning and computer vision applications to crop and extract frames of sharks from videos as well as classification schemes to discern shark media from non-shark media. Taxonomic classification of sharks is constantly debated for many species because they are morphologically similar to one another and there is little data that facilitates species-specific identification (Ebert & Stehmann, 2013). Classification models continue to incorporate convolutional neural networks for image filtering, and object detection algorithms for spotting sharks via underwater or UAV footage.

SharkPulse is a relatively new creation (2015); however it shows excellent potential to become a focal point for shark and ray conservation research. Our team has been tasked with helping expand the reach of sharkPulse to citizen scientists via a gaming framework which rewards users with points-based prizes and statuses. 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 a gamified monitor, citizen scientists will be able to seamlessly interact with the platform, validate shark images, and participate in inflating species-specific data repositories of this charismatic group of marine animals.

Furthermore, citizen scientists can actively participate in big data mining by uploading sightings from nature. SharkPulse relies on many different ways to extract reliable Elasmobranch sightings, and user uploads have proved a valuable resource. Users can also comment on potentially valid records to share opinions, experiences, and personal expertise on the matter which are not ubiquitous among other citizen scientists or even experts.

To gain a comprehensive view of this project and the efforts of sharkPulse, we (Table 1) have outlined the seven team members and two mentors working on either/both the taxonomic modeling aspect or the main focus of this project, the Validation Monitor. Jeremy is leading both efforts as a Master’s student and is advised by the client, Dr. Francesco Ferretti. Both the client and Dr. Edward Fox have mentored the team on many steps of these projects. The Validation Monitor team members have moved from one to several roles, but have identified one specifically that they have dedicated the most time to.
1.1 Requirements and Objectives

We discussed clear and obtainable objectives with our client that will transform sharkPulse’s out-of-date and no longer supported Validation Monitor into a sleek, interactive, and streamlined updated version. The main requirements for our project include i) building a baseline Shiny App (because sharkPulse already incorporates a Pulse Monitor for displaying warehoused records, this can be used as a template), ii) organizing a media flow into the Monitor, iii) gamifying the Monitor, and iv) updating the Web server with user input. These objectives were addressed and achieved through this project.

1.1.1 Shiny platform

To begin, our client and our team came to a consensus that a Shiny App was the optimal method to display records that had not been validated. Shiny supports HTML, CSS themes, widgets, and JavaScript actions (RStudio Inc., 2013). Shiny does not require development skills or Web server knowledge. The purpose of this project is to easily build a focal point for citizen scientists to aid in our research. Users will be able to manipulate data points by validating them, share their progress with point and ranking systems, and help each other identifying shark species with comment boards (this process also helps the researchers identify any discrepancies with the record).

1.1.2 Media Pipeline

Completed, but with room for refinement, the media pipeline is a data mining architecture tasked with pulling shark-related images from Flickr and storing them in the temporary table of the sharkPulse database. This table is home to records that have yet to be validated. To refine this application, we are looking into adding the image classification AI scheme so images are pre-
processed for shark images only. To eliminate redundancy, we are also looking into implementing a duplicate search.

### 1.1.3 Gaming

Requirements for gaming attributes of the Validation Monitor include ranks and point systems for participation and correct validation. Achievements will also be implemented to encourage users to focus on endangered species, species that have low spatiotemporal resolution, or geographical points of interest. The gaming interface will be included within the Shiny App as well as PHP displays of user profiles and leaderboards.

### 1.1.4 Back-end

Most of this project will encompass configuring the Web server to interact with the Shiny App and the database synchronously. Validation points and new users must be generated and updated in the database according to user input and the media pipeline. Gaming attributes will effect user and validation status as well. If a user is constantly validating shark images and participating in archiving these records, they should be rewarded with marks, but also tangible rewards. Our client discussed allocating funds for prizes and this is something to consider when developing a user interface.

### 1.2 Client

Our client is an Assistant Professor in the Fisheries Management and Conservation Department at Virginia Tech. Dr. Francesco Ferretti is a quantitative and computational marine ecologist and the creator of sharkPulse. He specializes in marine conservation, fisheries sciences, population dynamics, and quantitative ecology with an interest in sharks and rays. Dr. Ferretti wants to augment data-processing of non-validated shark sightings on his platform and speed up archiving of these records with the help of citizen scientists.

### 1.3 Challenges

Before deliverables were addressed, we defined the most notable obstacles that stood in the way of providing this platform. First, there are many layers to this project that have already proven convoluted in the design process. The server, website, database, and web application all need to work in concert for a suitable final product to be realized. Objects defined in the application need to be migrated to the server and the database for processing and possible manipulation (depending on user input). This involves using Unix-variants, PHP, HTML, Python, R, JavaScript, and SQL for database management. Second, Shiny is not a widely popular resource among computer scientists and is usually advertised to people with less experience in developing. Our group has had to test and gain experience with Shiny and the supporting language, R. Third, in order to grant points and ranks to users who have correctly validated a shark image, we need pre-existing knowledge of those images (control images). A large challenge to this project is determining where and how to integrate a validation network for the users’ taxonomic knowledge, so that the input is both meaningful and ecologically relevant.
2 Design

2.1 Methodology

To gain a better sense of our roles in the project as well as the roles of the users, we outlined two sets of user experiences. The purpose of giving users an interactive platform on the Validation Monitor is to motivate them to check sightings as real or not. This requires a Gaming User experience as well as a Knowledge User experience.

2.1.1 User type

**Game User**: This user will be using the shark validation monitor as a game. They will be shown sharks and be asked if the picture displayed is a shark and what kind of shark is being displayed to them. Our goal for this user is for them to quickly get the hang of what the game is by making an intuitive and clear interface (see Figure 1). We also want to encourage the user to continue playing the game to gather more information. That is why we also need to add encouragement like achievements or a leaderboard which will add a sense of accomplishment to the user.

**Knowledge User**: This user will be using the shark validation monitor both to challenge the knowledge they have, gain better knowledge of shark sightings, as well as contribute to the research of shark sightings. The goal for this user is to make sure the user is aware of the progress they are making both as someone who is learning and challenging their knowledge but also as someone who is helping further the research of shark sightings (see Figure 2). Adding user profiles that save and mark down progress as well as the achievements mentioned for the Game User are all starts to make a user feel like they are progressing.
2.1.2 Tasks flowchart

Figure 1: Gaming user logical task flow
For the Game user, the task is to evaluate the motivation of users. SharkPulse also uses a derived version of the ELO system created by Arpad Elo, hereinafter referred to as ELO. This is done by first allowing the user to create a user login via Google login. Once the user account is created, it is populated in the database with a predefined ELO and zero points, as well as zero achievements. The user is then allowed to validate images, some of which are “true positives”, meaning that they are already known to be sharks. If the user chooses the right answer for a true positive, they receive a boost to their ELO ranking. Any validation that the user does, also awards points. After performing a certain amount of validation, or other quantifiable feats, the user can receive achievements and be ranked on a global leaderboard.

For the Knowledge user, the purpose of the application is a lot less game-oriented and much more focused on the collection of shark knowledge and identification. A media pipeline is created from Flickr in which images that may or may not be sharks can be streamed into the game. True positives are also added to the database by these knowledge users in order to increase the number of quality evaluations that are made. Conditionals for correctly validating the type of shark are also added. All of this leads to the better evaluation and collection of user input with regard to the shark information that is being fed into the application.
2.1.3 Specific services

Table 2: Progression of Monitor services

<table>
<thead>
<tr>
<th>Service/device</th>
<th>Input</th>
<th>Output</th>
<th>Type</th>
<th>Interaction by user</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shiny App</td>
<td>Server and UI</td>
<td>UI - web page</td>
<td>R-language</td>
<td>yes</td>
<td>shiny, RPostgreSQL, RE2, shinyR, ConnectPelagisR3E</td>
</tr>
<tr>
<td>Leaflet map</td>
<td>Flickr geolocation and image URL</td>
<td>popup balloons on world map</td>
<td>R-language</td>
<td>yes</td>
<td>PostgreSQL connection</td>
</tr>
<tr>
<td><em>popup</em></td>
<td>image URL, taxa questions</td>
<td>user form</td>
<td>RHTML</td>
<td>yes</td>
<td>Flickr URL</td>
</tr>
<tr>
<td><em>point system</em></td>
<td>user input, form completion</td>
<td>database change</td>
<td>PHP/SQL</td>
<td>yes</td>
<td>LAMP stack web server, PHP</td>
</tr>
<tr>
<td><em>rank system</em></td>
<td>user auth validation</td>
<td>additive points</td>
<td>PHP/SQL</td>
<td>yes</td>
<td>LAMP stack web server, PHP</td>
</tr>
<tr>
<td><em>achievement system</em></td>
<td>user specific validation</td>
<td>new rank</td>
<td>PHP/SQL</td>
<td>yes</td>
<td>LAMP stack web server, PHP</td>
</tr>
<tr>
<td>build user database</td>
<td>user info (name, points, ELO)</td>
<td>n/a</td>
<td>SQL</td>
<td>no</td>
<td>PostgreSQL</td>
</tr>
</tbody>
</table>

Table 2 depicts specific inputs and outputs of the Validation Monitor as well as program and library dependencies. The webserver primarily sources Flickr for data-driven shark sightings and is built on the LAMP stack software (Linux, Apache, MySQL, PHP & Python).

2.2 Citizen Scientists as Users

The focal point of this project is tailoring a shark validation platform around users interested in identifying shark images. Our design should allow easy interaction for the user to check in, validate what they wish, and check out, recording progress and any points or achievements collected. Figure 3 depicts the results of the validation process.

![Image sourced from Facebook and validated by a citizen scientist as a Greenland shark.](image)

9/1/2015 - Eastern Canadian Arctic

**Common name:** Greenland shark  
**Scientific name:** Somniosus microcephalus  
**Source:** shark-referencesFB

Figure 3: Image sourced from Facebook and validated by a citizen scientist as a Greenland shark.

2.3 Web server and PostgreSQL database

The Web server is composed of an Ubuntu 18.04 LTS host machine which serves a LAMP stack software bundle. The sharkPulse website is built with WordPress and Elementor (a WordPress
construction plugin). The website is accessible through Virginia Tech’s hosting platform. The host name is [sharkpulse.cnre.vt.edu] and is redirected from [sharkpulse.org] through the Gandi hosting services.

The sharkPulse database is a PostgreSQL relational database system. The database, pelagic, is composed of many tables. For this project, we are only interested in the data mining table which contains all of the current sightings that have yet to be validated. Data contained and its constrained type is displayed in Figure 4.

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>integer</td>
</tr>
<tr>
<td>date</td>
<td>date</td>
</tr>
<tr>
<td>time</td>
<td>time without time zone</td>
</tr>
<tr>
<td>species_name</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>latitude</td>
<td>numeric(5,2)</td>
</tr>
<tr>
<td>longitude</td>
<td>numeric(5,2)</td>
</tr>
<tr>
<td>img_name</td>
<td>text</td>
</tr>
<tr>
<td>notes</td>
<td>text</td>
</tr>
<tr>
<td>users_email</td>
<td>text</td>
</tr>
<tr>
<td>validated</td>
<td>boolean</td>
</tr>
<tr>
<td>is_shark_1</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>is_shark_2</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>users_email_1</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>users_email_2</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>common_name_1</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>common_name_2</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>species_name_1</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>species_name_2</td>
<td>character varying[50]</td>
</tr>
<tr>
<td>control</td>
<td>boolean</td>
</tr>
<tr>
<td>yes_elo</td>
<td>integer</td>
</tr>
<tr>
<td>no_elo</td>
<td>integer</td>
</tr>
<tr>
<td>comment</td>
<td>character varying[350]</td>
</tr>
<tr>
<td>url</td>
<td>character varying[350]</td>
</tr>
</tbody>
</table>

**Figure 4:** Columns of data mining table in the Pelagic database.

In this project, we need to retrieve all columns except IP address and time. These columns will not be necessary for constructing a user-interface validation process. Instead of IP, we will use user email_1 as indication of the unique user that validated the specific image. The most important input data is derived from the ID, date, img_name, and latitude and longitude. This is important for the HTML form by displaying the full scope of ecological information we have on the image. The most important output data is logged in the validated and is_shark columns. This is where user input is stored.

The Web server host and Pelagic database work in concert by taking in and receiving validation data from the data mining table, manipulating table data, and updating ecological information.

**2.4 Shiny App**

R Shiny is the backbone of this project and acts as the conduit between the Web server and PostgreSQL database, the sharkPulse website, and the user interface. The Shiny App consists of its own pocket front-end and back-end and is composed of server.R and ui.R configuration scripts. The former script controls back-end data processing functions, database connection and querying strings, as well as rendering the leaflet map. The latter script controls front-end UI
aesthetics such as the Validation tab, title, and box size. The image within the HTML forms can be enlarged by hovering over it with a cursor. To resize back to normal, simply remove from hovering. Because the form is vertically long, you may to readjust the map center by dragging the map with the cursor.

**Figure 5:** Main page for the Validation Monitor constructed from R Shiny and hosted with the Shiny server software package.
Figure 5 presents the Monitor page as seen when the user jeremy (developer) is logged in. When not logged in, a username will not be displayed. Figure 6 shows the forms that are generated from clicking the leaflet map popups. This allows the user to inspect the image metadata (image and date) and consult the sharkPulse identification app if need be. Then they are instructed to discern whether the image is a shark, and if so, the suspected common and species name.

2.5 Project Plan and Timeline

To deliver on client-specified functions and develop a user system for validating shark media, we have implemented all facets of the Design section before the end of the term (approximately 4 month timeline). We have been on track to deliver a fully functional validation monitor which can distinguish a userbase, record validation effort, and log genuine sightings in the sharkPulse database.
The capstone of this project, however is the gaming aspect. Without a system to motivate citizen scientists, validation effort will wane. To fulfill our timeline goals, depicted in Figure 7, we constructed a gaming aspect before the end of April, which rewards users and facilitates competition to refine shark occurrence resolution.

3 Implementation

3.1 Testing

Most testing was done on the RStudio integrated development environment. Here the Shiny app was tested and developed for server and database interactivity. To integrate users, we tested login/logout and ease-of-access on WordPress. We experimented with several plugins designed for One-Click logins such as Google Login. To connect front-end user profiles with the user database within sharkPulse, we tested PHP pathways which received user input from the HTML forms in the Validation Monitor. All information was processed and functions were implemented within one PHP script to prevent a cumbersome back-end (this was a distinct problem with the previous Monitor).

Testing was also performed on the live server primarily with PHP. The action script was tested to ensure users were being tracked for validation progress, validation points, ranks, metadata appendages, and ELO calculators. To connect to the local Pelagic database, each PHP script included the postgresConfig.php script which logs into the database with the client master user and password.
3.2 Evaluation and Assessment

The evaluation of all aspects of this project are directly correlated with user satisfaction. Because sharkPulse is a relatively new creation and website, there are not many users to provide feedback. Fortunately, this project extends beyond the CS4624 class and the client. This project also encompasses efforts by collaborators of the client to expand sharkPulse awareness via media attention and research output. The Publication section of sharkPulse shows the value of recording species biodiversity of Elasmobranchii, and in the future, will reflect upon citizen science validation efforts. In the meantime, evaluation and assessment will be coordinated by our client, Dr. Francesco Ferretti, members and collaborators of his direct research group, our project Mentor, Dr. Edward Fox, and any users accessing the website who wish to provide feedback.

4 User Manual

4.1 User Login

The user can login to the sharkPulse monitor in order to save progress that has been made through validating points on the leaflet map. However this isn’t required for a user to begin validating points. In order to make the login experience easier, we have chosen a one-click login system that uses a Google related account. This not only allows the user to login faster, but can ultimately be a safer transaction. After loading the Validation Monitor, a user will be prompted to login by clicking the link displayed above the Monitor (Figure 8).

Figure 8: Current link to allow the user to login; Link embedded in the text “Login with Google to keep score!”

After navigating to the login page, the current system will prompt the user to either login through the WordPress website, or the user can click on the “Log In With Google” button to be sent to Google’s sign in options (Figure 9). After completing login through either method the user will be redirected back to the Validation Monitor where it will display the text “Logged in as:” along with the name associated with the account that was used to login and the score that is currently on record for that account (see Figure 5).
4.2 Validation and Scoring

While it is recommended for a user to sign-in before validating points on the Validation Monitor, it isn’t required, but any score accumulated will not be recorded if the user fails to sign-in. Once the user has decided to sign-in or not, they can then select any point on the map to start validating. The user can also zoom in and drag around the map to more clearly see and select points. When the user has selected a point on the map, if the picture in question clearly isn’t a shark or is a shark in captivity then the user can select “No” and leave any comments that they think could be helpful. For example in Figure 10 we can clearly see that this shark is in a public aquarium, and as such should be marked “No” (the comments aren’t required). This would also be the case if it were a picture of something else other than a shark (objects, people, etc.). After submitting, the information that the user submitted will be displayed, and the user can navigate back to the map by clicking the “Return to Map” button (Figure 11).
**Figure 10:** Validation point to be marked as “No”; Not a shark in the wild.

Here is the information you have submitted:

1. Shark: no
2. Common Name: 
3. Species Name: 

4. image ID: 4029
5. email jeremy1@yt.edu

**Figure 11:** Information displayed after submitting a “No” response for a validation point.

This type of “Yes/No” submission will grant the user +2 Validation Points (VP). Points in the current system keep track of a user’s progress, and will be given to a user based on different interactions. In Figure 12, the current scenarios that a user will be awarded points are shown, with each requiring different input. As in this first example the user will have been given +2 points for simply validating a point with a “No”. +3 VP will be awarded if the “Common” or “Species” name is given, and +5 VP if both are given. The user will be able to see changes to their score above the map such as shown in Figure 5 where jeremy has 192 VP. If the user has scored enough VP then their name will be added to the leader-board as shown in Figure 13.
4.3 Other Functionality

While some of the other features on the website weren’t a main focus of this project, the user may still wish to explore and use these features. Most notably, the site allows for the user to search through already validated sightings through the Pulse Monitor, and allows willing citizen scientists to upload their own shark sighting photos.

While on any page of the website, clicking the hamburger menu icon on the top right of the screen will open the navigation menu to allow the user to access these other features. Clicking on “The App” button will take the user to the page that allows them to directly upload a photo with various information such as location and email (Figure 14). If the user would prefer to use other means it is possible for the user to email these photos to sharkbaselines@gmail.com or simply upload to their favorite social media using the hashtag #sharkPulse. Uploading with this specific tag will allow the information to be scraped when collecting new data into the database.
Figure 14: Form for uploading details of a shark sighting including the photo file, species name, location, date, and email.

Clicking on the “Pulse Monitor” button from the navigation menu will allow the user to search through the validated shark sightings on the map. This map works in a similar manner to the Validation Monitor, but allows the user to filter the points shown on the map based on the various common or scientific names. It also allows the user to display these points in an aggregate or disaggregate, clustering many points together until zoomed closer to a specific point or displaying each point individually. Clicking on one of these points will display the photo, date of the sighting, the common and scientific name, and the source in which this sighting was obtained (Figure 15).
Figure 15: Pulse Monitor map displaying details of a shark sighting contained in the validated database (date, common/scientific name, and source obtained).

At the top of this map the user can select the option to display all of these points in a data table. This data table will allow the user to search for a specific point based on the information displayed in the table (date, species name, common name, location, and source) as in Figure 16. There are other controls that allow the user to display more entries per page or to search from a list of either common or scientific names.
Figure 16: Pulse Monitor data table displaying results from filtering for the date “1/23/2016”.

Users can also consult the identification app within the Validation Monitor if they are struggling to identify a specific record (see Figure 17). Acting as a media dichotomous key, the identification app takes users through a morphological guide of general shark features that may or may not be present in the images they are identifying. The hyperlink for this guide can be found on the validation form and can be opened alongside the map view to facilitate identification. The guide can be found at [http://sharkpulse.cnre.vt.edu/identification-guide/] or on the HTML form.

Figure 17: sharkPulse identification app
5 Developer’s Manual

5.1 ELO Ratings and User Validations

The sharkPulse validation monitor needed a system to increase the likelihood of users validating sharks correctly, as the database could easily be polluted should a user answer incorrectly on a frequent basis. To this extent, an ELO system was added to allow the system to autonomously ensure that results were of the highest quality. A fresh user starts with 1000 ELO, an arbitrary amount that is standard across all users to ensure uniformity. A set of “controls”, or images that have been pre-validated by the sharkPulse team, exist within the data mining table. Users will not be made aware of whether they are looking at a control or not. When a user answers a control, their ELO is shifted up or down by 50 points, based on whether they answered correctly or not. These control images will appear intermittently, and will allow for each user to be automatically gauged on their abilities throughout their time playing the game. The data mining table contains two columns, labelled elo_yes and elo_no. When a user votes on an image, their ELO will be added to the elo_yes or elo_no column, depending on their vote. This creates a system where multiple users can vote on the same image to ensure the validity of the votes. It also allows for a user who has proven their ability to validate images correctly to carry more weight in their votes than someone who is a novice, while also giving the novice the chance to potentially raise their ELO if they are able to consistently answer images correctly. When checking the data mining table to see which images are sharks, the developer needs to query all images where the elo_yes value minus the elo_no value returns a positive number. If the developer wishes to see all images that are not sharks, query all images where the elo_yes value minus the elo_no value returns a negative number.

A join table between the user_base and data_minning tables has been created called user_validations which stores each interaction a user makes with an image by storing the current user’s email against the image ID of any image that they validate. This can be used within the R Shiny app to ensure that users do not get the same image twice.

5.2 Database

The Validation Monitor deals heavily with the data_minning table within the Pelagic database (see Table 3). Here, shark sighting records are added, information appended to these records is fine-tuned, and user input is recorded alongside the image ID they have validated. The userbase table stores all new users and tallies their specific validation activity. The user validation table simply records the user along with the image ID they have validated to append that image with their ELO, acting as a proxy for validation confidence. The action page PHP script is the control center for these tables. Once a user clicks Submit, this script records the radiochoice yes no question, the common and/or species name, as well as any comments the user decides to input. All of this information is then added to the image’s metadata within the data_minning table. The script also adds validation points and gives or takes away ELO depending on if they have correctly validated a control image.
Table 3: Database tables and PHP action scripts.

Regarding an example of what the action_page.php script does, here is a snippet. A PostgreSQL connection is made, followed by grabbing global user information as well as checking for POST activity. This process prevents overhauling the PHP environment if there is no posting activity. If there IS posting activity, the script grabs information from the data_mining table that correspond to the image ID being validated (where id=$id). Then we can generate the ‘THANK YOU’ HTML page, present the image information along with the completed form that the user submitted. Finally, we can provide a redirection button back to the Validation Monitor if they choose to continue validating.

<table>
<thead>
<tr>
<th>DB table name</th>
<th>Description</th>
<th>PHP script 1</th>
<th>PHP script 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>userbase</td>
<td>Stores all user information</td>
<td>userbase.php - generates user profile</td>
<td>scoreboard.php - generates leaderboard</td>
</tr>
<tr>
<td>user_validation</td>
<td>Records all images validated and their validators</td>
<td>action_page.php - appends image with user ELO</td>
<td></td>
</tr>
<tr>
<td>data_mining</td>
<td>Stores all image information including validation progress</td>
<td>action_page.php - changes records based on user input</td>
<td></td>
</tr>
</tbody>
</table>
Figure 18: Code snippet from action_page.php that is used to generate a thank you page upon submission of a validation point and contains relevant information about the submission.

Furthermore, the data_mining table is continuously updated with shark sightings sourced from Flickr. Figure 19 shows the .sh script responsible for running every 7 days and querying Flickr posts for the tag shark. This cronjob activates the flickrSearchShark.py Python script to search within a specified geom box between a specific date. The script will compile results into a table
identical to the data mining table to allow easy addition to the database. Next, the script will download the images into the sharkPulse shark detector. This is an image classification model built with convolutional neural networks, and is able to discriminate between shark and non-shark images with 92% accuracy. Using the classification model enables better filtering of social network queries and reduces the noise for users to validate.

```bash
media@sharkpulse-one:/flickr$ cat cronjob.sh
dt=$(date +%Y-%m-%d);
echo "$dt"
datespan=$(date --date="${dataset_date} -${date_diff} 7 day" +%Y-%m-%d)
echo "$datespan"

cd '/home/media/flickr/'
python3 flickrSearchShark.py 'shark' $dt $datespan '-180,-90,180,90' 0

cd '/home/media/General_Shark_Model/model_predictor/'
python3 Flickr_Inference.py
```

**Figure 19:** Weekly Flickr query for the tag shark.

### 5.3 Display

User profiles are displayed using the userbase.php script. The Leaderboard is displayed using the scoreboard.php script. These scripts are sourced with HTML iframes through the Elementor plugin on the Validation Monitor front page [http://sharkpulse.cnre.vt.edu/can-you-recognize/](http://sharkpulse.cnre.vt.edu/can-you-recognize/).

The Shiny App grabs 200 random non-validated records from the data mining table and has a 1 in 8 chance of grabbing 10 random control images. There is a shuffle button in the top right corner to give the user a shuffling feature when they no longer wish to look at the current displayed dataset. Currently, whenever a user validates an image and returns to the map, the dataset is automatically shuffled. We did not have a chance to fix this bug.

The userbase.php script generates user profiles. PostgreSQL fetching and global email objects comprise this script. However, to generate user profiles, this information is conditional on the user's validation progress. We need to know their validation points and their rank. If the user has none of this, they need to be registered (using the current data PHP function to record when the user was first registered). Next, we retrieve the user's information and depending on their points, a rank is assigned, and depending on their rank, an icon is assigned. The icon is generated with the Font Awesome icon toolkit. In this case, we use a standard fish icon and grant different colors and sizes based on the user's rank ([Fonticons, Inc] 2018).
Figure 20: Code snippet from userbase.php that is used to update the players rank icon.

Rank icons are awarded within the HTML conditionals of the userbase.php script. This is identical to how the Leaderboard is generated and displayed.

The scoreboard is generated by grabbing the top three performers in the userbase table and displaying their usernames and scores alongside the same font awesome icon.
Figure 21: Code snippet from scoreboard.php that is used to generate the leader-board.

Currently, the user profile and the Leaderboard are updated every 15 seconds and 150 seconds respectively. This was done because we could not find a solution to automating updates of the userbase and scoreboard PHP scripts with HTML form submissions. So, instead, the display scripts are refreshed as in Figure 22.

```php
$sql = "select _username, _email, points, _rank from _userbase order by _points desc limit 3;";
$result = pg_query($dbconn, $sql);
$resultArr = pg_fetch_all($result);

Figure 22: Code snippet that is used to update the user’s profile every 15 seconds.
```

6 Lessons Learned

The project has tested our knowledge in Webserver applications and the localization of several different scripting languages and moving parts. R Shiny proved to have a steep learning curve, but with the proper tutorials and testing, we were able to write a strong Validation app which talks to the database through PostgreSQL R packages. The app is most importantly displayed on the sharkPulse website using HTML iframes. Creating HTML forms in R proved to be an interesting experience as you must abide by the language rules of both HTML and R. Fortunately, R objects could be passed as hidden values in HTML and pushed to the PHP action scripts for image ID-specificity as well as image date and URL.

PHP testing was a smoother experience, but required diligence and temperament. PHP errors are hard to identify if not outright shown in log errors or PHP commands (which they usually aren’t). These errors usually took the form of missed quotations or semicolons. Integrating PHP with WordPress was a simple enough process because WordPress provides many Shortcode options as well as HTML iframes for sourcing the scripts themselves. Overall, this process was easy to learn and manipulate.
Acknowledgements

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

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


