Final Report: WildWatch

CS-4366 Senior Capstone Project

April 30th, 2025

Instructor: Victor Sheng

Team Members

Lauren Fagley  
Computer Science  
Texas Tech  
Ft. Worth, United States [lfagley@ttu.edu](mailto:lfagley@ttu.edu)  
  
  
Mary Anne Onyedinma  
Computer Science  
Texas Tech  
Mesquite, US   
[maonyedi@ttu.edu](mailto:maonyedi@ttu.edu)  
  
  
Avery Cone   
Computer Science   
Texas Tech   
Fort Worth, United States [avcone@ttu.edu](mailto:avcone@ttu.edu)   
  
  
Ethan Singer  
Computer Science  
Texas Tech  
Austin, United States  
[etsinger@ttu.edu](mailto:etsinger@ttu.edu)

Aayat Alsweiti  
Computer Science  
Texas Tech  
Lubbock, United States  
[aaalswei@ttu.edu](file:///C:\Users\ethef\Downloads\aaalswei@ttu.edu)

1. **Introduction**

The Online Animals and Plants Distribution System is a web platform that helps track and display information about animals and plants. Users can report sightings by providing details such as location, time, species name, images, and a short description. The system also uses artificial intelligence to check if the reported species name is correct. One key feature is the ability to track bird migration patterns, giving useful insights for researchers and conservation efforts.

Monitoring biodiversity is important for understanding and protecting ecosystems. Traditional methods of tracking species take a lot of time and effort. This system makes it easier by allowing people to share sightings, helping build a large and useful database. Other platforms like iNaturalist and eBird have shown that community-driven data collection is effective. Our system improves on these by using AI to verify species names, making reports more accurate and useful.

Another important part of the system is migration tracking. Bird migration affects ecosystems and helps scientists understand environmental changes. By collecting and analyzing reports, the system helps track bird movement, which is useful for research and conservation. By combining user participation, AI verification, and real-time data visualization, this system becomes a useful tool for documenting biodiversity, supporting research, and educating the public.

1. **Motivation**

Our motivation for this project comes from the importance of documenting animals and plants. The most important reason for this documentation is its value to scientific research. For example, tracking animals can help scientists learn more about health problems affecting animals and humans, leading them to develop safer medicine and new medical procedures to treat various diseases. As for the plant side, this project is especially beneficial since plant collection donations have steadily declined over the past 50 years. Without research samples, botanists cannot properly document plant biodiversity. Having people contribute to plant documentation will greatly help scientists gain a better understanding of how to protect the world’s plant biodiversity.

Additionally monitoring migration patterns helps scientists understand if the ecosystem is healthy because migration patterns are essential for maintaining ecosystem balance. Animal migration helps enhance plant diversity by helping with seed dispersal and pollination, helps various species gain essential resources they need to reproduce and survive, and regulates predator-prey dynamics ensuring a stable population. If migration patterns are disrupted, it will lead to major shifts in ecosystem and biodiversity health. For these reasons, it is extremely important for scientists to be up to date with migration patterns in order for them to take immediate action to protect wildlife and natural habitats.

Finally, our project has an educational purpose. Many people have a strong interest in wildlife and plant life but might not have the resources to learn about them. By creating this project we will be able to allow users the ability to report and learn about biodiversity and promote wildlife and environmental awareness.

1. **Problem Definition**

Accurately tracking plant and animal distributions is essential for ecological research, conservation efforts, and educational purposes. However, traditional data collection methods rely on time-intensive field studies, limiting the scale and frequency of observations. To address this, we propose developing a web-based platform that allows users to report sightings of local flora and fauna. This crowd-sourced data can then be used to study species distribution, migration patterns, and ecological changes over time.

However, this project presents several key challenges:

1. **User-Friendly Interface and Data Visualization**
   * The platform must be designed to be intuitive and easy to navigate for users of all experience levels.
   * A visually engaging educational section should provide species information, allowing users to learn about reported plants and animals.
   * An interactive map must be implemented to display real-time species distribution based on user reports, enabling users and researchers to track sightings geographically.
2. **Reliable Data Storage and Verification**
   * A scalable database is required to store and manage user-submitted reports alongside baseline species data sourced from authoritative references.
   * AI-based verification methods must be implemented to ensure accuracy and prevent false or misleading information from being included in the dataset.
3. **Scope Management**
   * Cataloging all species worldwide is beyond the scope of this project, requiring careful limitations on geographic regions and species categories.
   * A defined dataset of species must be selected, balancing comprehensiveness with feasibility.

By addressing these challenges, this project aims to create a functional, scientifically valuable, and user-friendly platform that supports both researchers and the general public in biodiversity tracking.

**IV. Approach**

*Back End*: We will be using mySQL as our database management system and hosting the database locally on one of our devices. We will use node.js to connect to it to our website and be able to query and draw data from it from the website instead of having to use the console that we currently are. This gives us more flexibility on the entire database then hosting it on a server like MongoDB.

At the moment, we plan for two tables within our database: a table we fill out with info about all the species that are in Lubbock and a table that is there for user submissions.

The table that we are filling out ourselves will have four fields: an ID for the species, and this ID will be both unique, the primary, and be how animal and plant are told apart. Animals will have an even ID number while plants will have an odd ID number. There will also be a tiny text species name that has the common name of the species such as 'Northern Mockingbird'. There will also be a short description of the bird using either medium text or long text that is shown when the creature is searched up alongside a photo that is stored as a link to a photo rather then the whole photo in the database.

Meanwhile the table that is for user submissions doesn’t allow any NULL categories- all the sections of the database need to be filled out and this will be secured by needing every box of the post page of the site to be filled out to even send it over to the AI. That way a NULL tag won’t break our database by having a NULL where it knows there cannot be one.

This table will have 5 categories to it. It will have a unique primary tab called SubmissionID, this one will autoincrement by 1 with every posting into our website to guarantee that none of them overlap in number. We use this rather than a userID as we don't use logging in, everyone is a guest on our website so there are no userIDs to keep track of. Instead the whole website will be accessible to everyone just by accessing it. Our next category is the location. It is the Json type as it is an array: [Latitude, Longitude] and this is required to be known as it is how we will display where the creatures were seen on a map. Without knowing where it was seen, we would have no way to map it out for better viewing. Next, we have the photo of the creature, just like the other table this one is stored as a link to a photo instead of the entire photo in our database so it's a medium text type. Next we have a singular bit to decide if it's a plant or an animal: this is for faster querying and because outside of foreign keys we don't have any way to tell if they are a plant or an animal. Finally, we have a foreign key of ID which is the primary of the other table. This is to identify what kind of animal it is, we can get it's name from the personal table which is what it is for.

As for our AI, we plan on using Dragoneye which has an already made API for JavaScript that works with Node.js. It is able to see the species of animals and plants in a photo that it is input. This is exactly what we need for our verification tools and it is at an affordable price of 5 dollars per 1000 uses. With 5 of us splitting the price, that is 1 dollar per 1000 uses which is not a big deal especially not at this stage of our project where we will have only so many visitors to our site, so even though this is a free website so any costs will be out of our pocket: the costs are manageable.

We will have a search bar for users to look into the data we have on plants and animals: such as where they were reported, facts about them, migratory patterns if they’re one of the 664 Texan birds, and images of the animals. While this search bar will look in the names for anything that matches the chain, even if it’s in the middle or end of the name, it will not be trying to find closely related words in the database and asking you if you meant them.

Due to the size and number of creatures and plants in the world, we will be limiting the scope of our app and database for now to just the city of Lubbock. We will be doing this for a more manageable set of data and a more manageable time for the project as Texas alone has 664 species of birds and many more creatures. It makes it more possible to do the database as MongoDB due to their storage limits.

Overall, our backend will be done in JavaScript for ease of integration with the frontend.

*Front End:* To develop the front end of the Online Animals & Plants Distribution System, we will follow a structured approach that ensures an interactive, user-friendly, and visually appealing experience. The system's primary functions include allowing users to report sightings of animals and plants, view an interactive map displaying distribution data, track migration patterns of birds, and receive automatic name correction suggestions for species identification. To achieve this, we will use HTML, CSS, and JavaScript as the core languages, along with frameworks and libraries like Leaflet.js for interactive mapping, Chart.js for data visualization, and possibly Bootstrap or Tailwind CSS for responsive design.

The development process will begin with wireframing and UI design, using tools like Figma to create a clear layout for the website. The key pages will include a Home Page for navigation, a Report Sighting Page where users can submit their observations, an Explore Page for users to explore sightings, a Distribution Map Page displaying wildlife reports, a Migration Patterns Page visualizing bird movements, and a Species Guide Page to allow users to view and explore different species in Lubbock. Once the design is finalized, we will proceed with building the website structure using HTML while styling it with CSS, ensuring a modern and responsive layout. JavaScript will be used to enhance interactivity by implementing form validation, dynamic image previews, and smooth scrolling.

Further enhancements will include integrating Leaflet.js to create an interactive map that displays real-time or previously reported sightings, making it easier for users to track biodiversity in their area. Additionally, Chart.js or D3.js will be used to dynamically visualize bird migration patterns based on available data. To ensure an engaging user experience, animations, tooltips, and interactive elements will be added where necessary. Lastly, rigorous testing will be conducted across different browsers and devices to ensure full responsiveness and accessibility. Once tested and refined, the front-end will be deployed using platforms such as GitHubPages, Netlify, or Vercel, making it accessible to users. This approach ensures that the front-end of the system is intuitive, interactive, and easy to use, aligning with the project's goal of encouraging users to contribute to wildlife tracking and conservation efforts.

*Testbed*: We have not reached the part of our project to be using a testbed yet, so we have not decided on which to use. Our first option is Mocha which works for both front and backend as well as has debugging for Node.js. It also includes Cross Browser and Asynchronous testing. The other option is Jest which is what Facebook uses. It uses faster testing and is good for testing the performance of websites as well as end to end testing.

Even though there is some overlap to what the testbeds do, they both also do different things so there is a chance that we will try to use both to test both debugging and performance testing. This would only be if we have enough time, but Jest does have the benefit of being a fast tester.

**V. UML Diagrams**

A diagram of a diagram

AI-generated content may be incorrect.

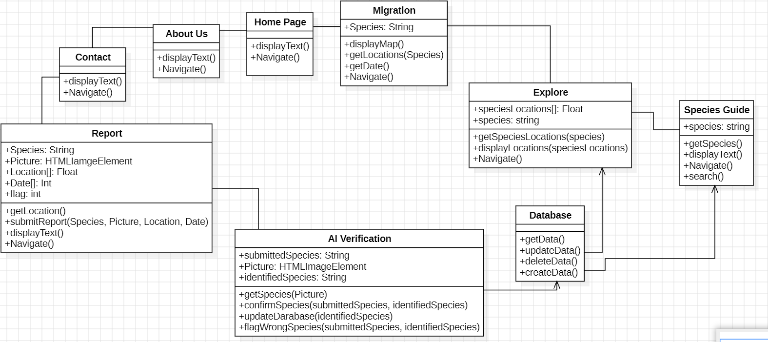
**Figure 5.1 Use Case Diagram**

This Use Case Diagram for WildWatch provides a visual representation of the core functionalities and user interactions within the online animal and plant distribution tracking system. The primary actor in this system is the User, who can access and interact with various features without the need for login or registration. The diagram outlines the user’s ability to navigate across the main pages of the website, including the Home Page, Report Sighting Page, Explore Page, Migration Patterns Page, Species Guide Page, About Page, and Contact Page. These pages are designed to facilitate user engagement and provide educational and interactive content regarding local biodiversity.

A key use case is the "Submit Sighting Report" function, which allows users to contribute to the system by reporting sightings of animals or plants. Each report requires users to provide critical details like the species name, time, location, a brief description, and a mandatory image upload to support accuracy and verification. Once a report is submitted, the system automatically invokes an AI-powered validation module, represented in the diagram as the use case "AI Validates and Suggests Corrections." This AI feature plays a vital role in ensuring data quality by analyzing the submitted information, such as checking if the species name matches the image or known species data, and providing suggestions or corrections if the user is incorrect.

Additional user functionalities include viewing submitted sightings in the Explore section, which promotes community-driven data sharing and environmental awareness. Users can also view consistently updated visualizations of bird migration patterns, supported by real-time data, and explore a comprehensive Species Guide to learn more about the flora and fauna in their area. The Contact the Team use case provides a communication channel for users to reach out for support, provide feedback, or ask questions about the platform.

Overall, this diagram describes the user centralized design and intelligent backend capabilities of the WildWatch system, highlighting its dual goals: to empower the public to contribute to biodiversity monitoring and to ensure the integrity of reported data through AI-driven support. The use case diagram serves as both a technical and conceptual map of how WildWatch functions as an interactive, intelligent web platform for local ecological engagement.



**Figure 5.2 Class Diagram**

Figure 5.2 depicts the class diagram. The pages of the website include the home page, migration page, about us page, contact page, explore page, species guide page, and report page. All of these pages are associated with one another and can be easily navigated through. Each page has access to switch to another page with the navigate operation. Additionally, each page displays text with the display text operation.

The report page allows the user to submit a reported sighting of a plant or animal species. It uses a string variable called species, an HTMLImage Element variable called picture, an array of float points called Location, an array of integers called date, and an integer called flag. Once the user submits the report, these variables store the information reported by the user that is respective to their names. The flag variable stores whether or not the user is reporting a plant or an animal. A plant will be flagged as 0 and an animal as 1. Furthermore, the report page contains a getLocation() operation. This operation will ask permission to access the user’s location. If given permission, the location of the user will be recorded and stored in the location variable. Otherwise, the user can manually insert the location of the species. This option should be used if the user is reporting the species in a separate location from where they saw the species. All variables are required to be reported.

Once the user submits the report and the user submissions are stored within the variables, these variables are verified by AI. Specifically, the AI takes the user submitted species name, stored in the submittedSpecies variable, and the user submitted photo, stored in the picture variable. It scans the photo itself and determines what it believes the species to be. Once it determines the species, it stores it in the identifiedSpecies variable. It then checks whether the submittedSpecies and identifiedSpecies are the same. If they are, it uploads the data submitted by the user to the database. Otherwise, it notifies the user that the species may be incorrect and the suggested species. It then submits the corrected species and the rest of the user submitted information to the database.

The database contains basic create, read, update, and delete operations. The AI verification has a directed association to the database and the database has a directed association to the explore page and species guide page. This is because the AI verification directly communicates with the database and the database directly communicates with the explore page and species guide page. However, the database does not communicate back with the AI verification and the explore page does not communicate back to the explore page.

The explore page provides data visualization for the data submitted by users. It contains a map of Lubbock and a search bar. The user can use the search bar to search for a particular species that they are interested in. Once they select this species, dots will appear on the map where this species has been spotted. To achieve this, the explore page uses a getSpeciesLocations operation to retrieve the species’ location data from the database. It then uses the displayLocations operation to display the locations.

Similar to this, the species guide page uses data from the database to display information for the user to read. It has a search bar where the user can search for a specific species and see information about that species. Otherwise, the user can scroll through the listed species and read short descriptions about them and see photos of them.

Lastly, the migration page takes the desired bird species through a search bar similar to that of the explore page. It then displays the migration pattern of the selected bird species. It uses a chart to show when the birds are within Lubbock and when they are not.

A diagram of a system

AI-generated content may be incorrect.

**Figure 5.3 Sequence Diagram**

This diagram is our UML sequence diagram. What it shows is how our WildLife sighting system works. At a basic level, the website allows users to submit a wildlife sighting, which is then gets validated in our WildWatch System. There are three main participants involved in this sequence which would be: the User - the entity providing the input data (species sighting), the System - the entity acting as the intermediary and interface for the user, and an AI Validator - the entity that processes and analyzes the data to make sure that the data is accurate.

The process begins when the user opens the Submit Sighting Report page on the system. The User then enters details about the sighting which includes the species type (either plant or animal), location, and date. Next, the User uploads an image of the plant or animal that they saw. After all that is done the User clicks “Submit”. Once the User clicks “Submit” all the information entered including the image is sent to the System.

The System then forwards all this data to the AI Validator. The AI Validator will now analyze the sighting details provided by the user. The AI Validator then checks the accuracy of the information entered with the sighting picture. If needed the AI Validator will suggest corrections if the information is not accurate for the picture uploaded. But if all the information is correct the System displays a success message. This system is very efficient since we are combining user input and an AI validator which will help to ensure data reliability. Since the user will have to put in information about the sighting to the best of their knowledge.

Overall combining human input with AI validation helps ensure data accuracy. This is a great way to make sure the data is accurate, since there are basically two entities checking information. Additionally, If the user gets information wrong then the AI will be able to always catch it and inform the user that they have entered incorrect information. Not only will it inform the user of any errors, but it will also suggest what the species might be based on the image. Furthermore, this interface is designed to guide the user step-by-step through the process, making it accessible even for those without technical or biological expertise. Lastly the main benefit of this system is how it will be able to contribute to many scientists’ initiatives by ensuring the data collected is both easy to submit and scientifically reliable. Because as mentioned before there is a need (specifically plant samples) because plant collection donations have steadily declined over the past 50 years. And without research samples, botanists cannot properly document plant biodiversity. This website will help scientists obtain the essential data they need to ensure plant biodiversity. Additionally, science also needs animals’ samples so they will be able to learn more about health problems that affect animals and humans which would lead them to develop safer medicine and new medical procedures to treat various diseases. So, because of this website scientists will be able to obtain the plant and animal samples they need to ensure plant biodiversity and to be able to create better medicine.

This diagram illustrates a semi-automated process that combines user input with AI assistance to ensure accurate wildlife data reporting. It demonstrates how our system balances human observations with machine validation to improve both the quality and reliability of submitted data. By guiding users through a simple submission process and using AI to verify the accuracy of the information, the system makes wildlife reporting accessible to the general public while maintaining scientific integrity.

This approach is especially valuable at a time when plant and animal data collection is declining. With accurate, AI-verified reports, researchers can gain access to much-needed information to support biodiversity studies, conservation efforts, and even medical research. Overall, the diagram captures a thoughtful integration of technology and user engagement to support real-world environmental and scientific needs.

**VI. Problem Solutions**

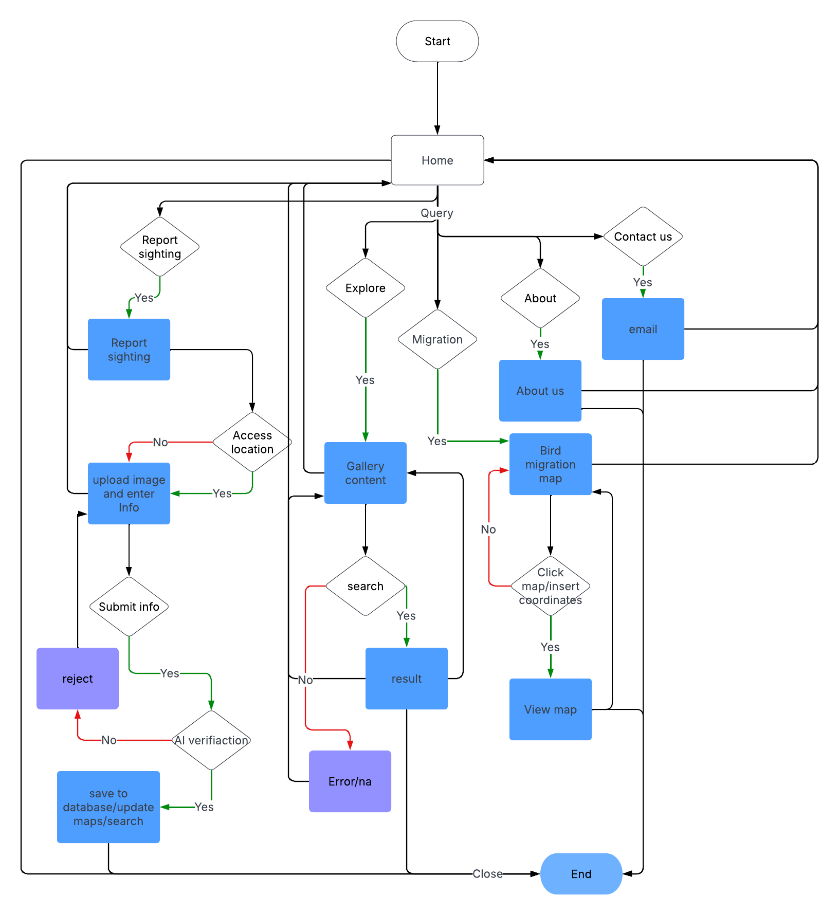
This section explains how our WildWatch system is designed to solve the main problems we identified earlier. We wanted the platform to be easy for users to navigate, reliable when it comes to reporting sightings, and efficient in handling and verifying data. The Flowchart breaks down how users move through the site depending on what they choose to do, while the System Architecture shows how everything works behind the scenes—from the user interacting with the website, to the backend handling requests, to the AI helping verify what species was seen. Both parts work together to make sure the system is accurate, easy to use, and ready for future improvements.

The flowchart for WildWatch outlines the navigation paths and decision points that guide users through the system. The flow begins at the Home Page, where users are presented with five main options: Contact Us, About Us, Migration, Explore, and Report Sighting. These options are displayed clearly to allow users to quickly choose their intended destination on the platform.

If the user selects Contact Us, the system takes them to an informational page that displays the team’s contact details, including emails and other relevant communication links. From this page, the user may either return to the Home Page or end their session by closing the page. Similarly, selecting About Us directs the user to a page that introduces the purpose of the WildWatch platform, outlines its mission, and provides an overview of its key features. This page also allows the user to navigate back to the Home Page or exit the system.

When the Migration option is selected, the user is taken to a page that displays bird migration patterns. On this page, the user is given the option to either view the interactive migration map or insert coordinates manually. If the user selects the map option, the system shows relevant bird migration data based on the user’s chosen location. If the user does not select either option, they remain on the migration page. From here, users may choose to return to the Home Page or close the session.

Selecting Explore navigates the user to a gallery of wildlife and plant reports submitted by others. From this gallery, users are provided with an optional search feature. If the user enters a valid search query, the system displays relevant results. If no results are found, or the query is invalid, the system displays an error message or a “Not Available” notice. Regardless of the outcome, the user may return to the gallery, return to the Home Page, or exit the system entirely.



**Figure 6.1 Flow Chart**

The Report Sighting option takes the user to a submission page where they are asked to upload an image and enter key details, including the species name, location, time of sighting, and a brief description. Within this process, the user is prompted to allow the system to obtain a more accurate location. If the user agrees, the system retrieves location data directly from the device. Once the information is submitted, the system uses the Dragoneye API to verify the image against the provided species name. If the AI confirms the match, the report is saved to the database and becomes accessible to other users through the Explore and Species Guide pages. If the AI identifies a mismatch, the system notifies the user and returns them to the submission page for correction. From this point, users may either resubmit, return to the Home Page, or close the session.

This flowchart represents the logical flow of WildWatch, where users are guided step-by-step through the platform. Each path is designed to offer clarity, control, and flexibility, ensuring users can navigate the system with ease while contributing valuable data to the broader biodiversity database.

The system architecture for WildWatch is composed of several interconnected components that work together to manage user interactions, process data, and ensure accurate species verification. At the core of this architecture is the relationship between the user interface, server-side logic, database storage, and the AI verification tool.

A diagram of a web architecture

AI-generated content may be incorrect.

**Figure 6.2 System Architecture**

The system begins with the user, who interacts with the WildWatch platform through a frontend built using HTML, CSS, and JavaScript. This interface allows users to navigate the website, report sightings, browse existing data, and explore features such as migration tracking and species guides. The frontend is designed to be responsive and accessible, providing a smooth user experience across various devices. When a user submits information or performs an action that requires backend processing, the request is sent to the Node.js server, which serves as the backend of the application. The backend is responsible for handling incoming data, managing user requests, and coordinating between the frontend, database, and external APIs. It ensures that data is validated, formatted correctly, and securely passed between components.

The backend communicates with a MySQL database, which stores all user-submitted sightings, species details, migration data, and verified entries. This database supports standard create, read, update, and delete operations, allowing the system to keep all content organized and accessible. The structure of the database is designed to support efficient searches, filtering, and geographic queries, enabling features such as the Explore Map and the Species Guide.

In addition to the database, the backend is also connected to the Dragoneye API, which performs image-based species verification. When a user submits a report, the backend sends the uploaded image to the API, which returns a predicted species name based on AI analysis. If the identified species matches the user’s input, the backend saves the report to the database. If the names do not match, the system informs the user and prompts them to revise their entry before resubmitting. This verification step helps maintain the accuracy of the data and supports the scientific credibility of the platform.

Together, these components form a reliable and modular system that supports real-time data collection, verification, and exploration. The architecture ensures a seamless connection between users and backend operations, while maintaining data accuracy and performance. By integrating external AI tools and a structured database, WildWatch is able to offer a powerful yet accessible platform for monitoring biodiversity.

**VII. Timeline**

Our development followed a structured timeline that allowed for organized progression and balanced responsibilities across the team. In January, we focused on brainstorming, defining our goals, and researching potential technologies that would best support the system we envisioned. We evaluated several ideas before narrowing down our concept to a wildlife and plant tracking tool, as this project best aligned with our interests in community impact, environmental awareness, and real-world applicability. By February, we had officially launched frontend development. We divided tasks for UI wireframing using Figma, finalized the layout of core pages such as Home, Explore, Report, and Migration, and began writing foundational HTML and CSS. Simultaneously, we discussed backend requirements, sketched out our MySQL database schema, and planned API integrations. In March, implementation ramped up: backend logic was constructed in Node.js, the MySQL tables were created and populated, and Dragoneye API trials began. We refined our architecture and flowcharts and focused heavily on debugging, reviewing use case diagram scenarios, and validating AI results. April was largely dedicated to connecting the frontend and backend, ensuring smooth data flow between user submissions and backend storage and logic. We resolved responsiveness issues, ensured all forms and pages worked reliably, and prepared demo-ready test cases. By May, we finalized documentation, addressed the remaining bugs, created visual assets for the presentation, and packaged everything for submission.

**VIII. Team Roles & Contributions**

Each team member contributed to multiple areas. Roles were divided as follows:

* **Lauren Fagley:**

Lauren was one of the lead backend developer. She implemented the AI validation flow, ensuring every user-submitted image is sent to the validator, which then returns a “correct” or “incorrect” response. Lauren integrated that feedback into the submission workflow so that, regardless of the AI’s verdict, an appropriate message is relayed back and the report is processed accordingly. She also vigorously help to debug the code.

* **Mary Anne Onyedinma:**.

Mary Anne helped with both the front end design and backend integration. On the front end, she created the initial wireframes. Mary Anne also wrote a JavaScript script for form handling that captured user inputs photo files, GPS coordinates, species name, and date encapsulated them into variables, and sent them to the AI. If the AI approved the submission, it was forwarded to the database for storage; if not, nothing was stored and the user was prompted to review their information. Additionally, Mary Anne assisted with defining and documenting the API routes, implemented client-side validation to catch errors before sending requests, and collaborated with the backend team to ensure consistent data formats. She also wrote end-to-end tests to verify the full submission workflow and debugged cross-browser compatibility issues, guaranteeing a seamless user experience.

* **Avery Cone:**

Avery was the lead front-end developer; she helped to create the code for the front end. She implemented and built out the UI components for both reporting sightings and exploring the map. She developed the Home page, Report Sighting page, Contact Info page, Migration page, and Explore page, and she helped to debug code throughout. Additionally, Avery optimized responsive layouts for desktop and mobile, integrated interactive map markers and clustering logic, wrote JavaScript event handlers for user interactions, and collaborated with designers to refine page flows and styling.

* **Ethan Singer:**

Ethan also took a lead on the backend. He took the lead on creating the Database. He helped to design and maintain the MySQL schema for species information and user submissions. He also wrote all the server-side logic that processes the information that was sent from the front-end JavaScript code of valid information; parsing all of that information and then executing the appropriate SQL queries to insert or update those records in the database.

* **Aayat Alsweiti:**

Aayat contributed to the backend development and AI integration for species identification within the WildWatch system. She also contributed in the creation of system documentation and technical architecture diagrams to ensure a clear understanding of the platform’s structure and workflow. She assisted with debugging and enhancing backend functionalities as needed throughout the project lifecycle.

**IX. Limitations and Known Issues**

Despite meeting our functional objectives, WildWatch has several current limitations that could be improved in future iterations. **First**, the Dragoneye API, while effective for many species, shows reduced accuracy for less common animals or blurry submissions. The AI occasionally misidentifies closely related species or fails to make a confident prediction. This can lead to a frustrating experience for users and potentially introduces minor errors into the dataset. **Second**, the lack of user accounts means that data cannot be personalized. Users cannot track their previous submissions or save searches, which limits long-term engagement. Future versions could incorporate simple account creation or session tracking for greater user functionality. **Third**, the system currently targets only the city of Lubbock. The backend is hosted locally, which presents challenges in handling higher traffic or expanding to cover other regions. Cloud deployment and geographic scaling will require infrastructure upgrades. Additionally, **mobile responsiveness** remains a concern. While the core UI is functional on most screens, map elements and image previews can appear misaligned on smaller devices. Furthermore, **data updates are not real-time**; users must refresh the page to view new sightings. This reduces interactivity and weakens community engagement. We also encountered technical limitations in how we store images: all photos are stored via URL links, which may break over time. A more reliable file hosting solution or integrated image database is needed. Lastly, most form and input validation is currently handled on the client side. Expanding robust backend validation and sanitization is critical for long-term security and consistency. These issues do not interfere with the core purpose of WildWatch but highlight areas that will improve its resilience, scalability, and user satisfaction in the future.

**X. Validation & Testing**

Thorough testing and validation were central to our development approach, ensuring WildWatch functions as intended across all core features. We began by testing individual components, such as the submission form, search bar, and migration chart rendering, across multiple browsers (Chrome, Firefox, Safari) and screen sizes. Every form field was verified for correct input handling. We included JavaScript checks to prevent blank or incorrectly formatted entries from being submitted. The AI validation system was then tested with a variety of sample images. We observed that common bird species such as cardinals and mockingbirds were accurately identified over 85% of the time, while more obscure or poorly lit images were less reliably categorized. In these cases, the system either offered an incorrect match or returned a low-confidence response. We also conducted tests for incorrect behavior, such as uploading unrelated photos or entering invalid text in species fields. The system handled these gracefully, often by rejecting the entry or prompting user correction. Additionally, we simulated failed API calls to ensure that the site could respond without crashing. For geographic validation, we tested both manual coordinate entry and automatic location access using the browser’s geolocation API. These tests confirmed that the mapping logic correctly displayed markers at the reported sighting locations. Lastly, we plan to incorporate formal test frameworks such as **Jest** and **Mocha** in future phases. This will allow us to automate component tests, simulate concurrent user activity, and evaluate how well the system handles real-world usage loads. While we relied heavily on manual testing for this stage, the system remained stable, responsive, and functionally accurate throughout our validation trials.

**XI. Artificial Intelligence (AI) Integration**

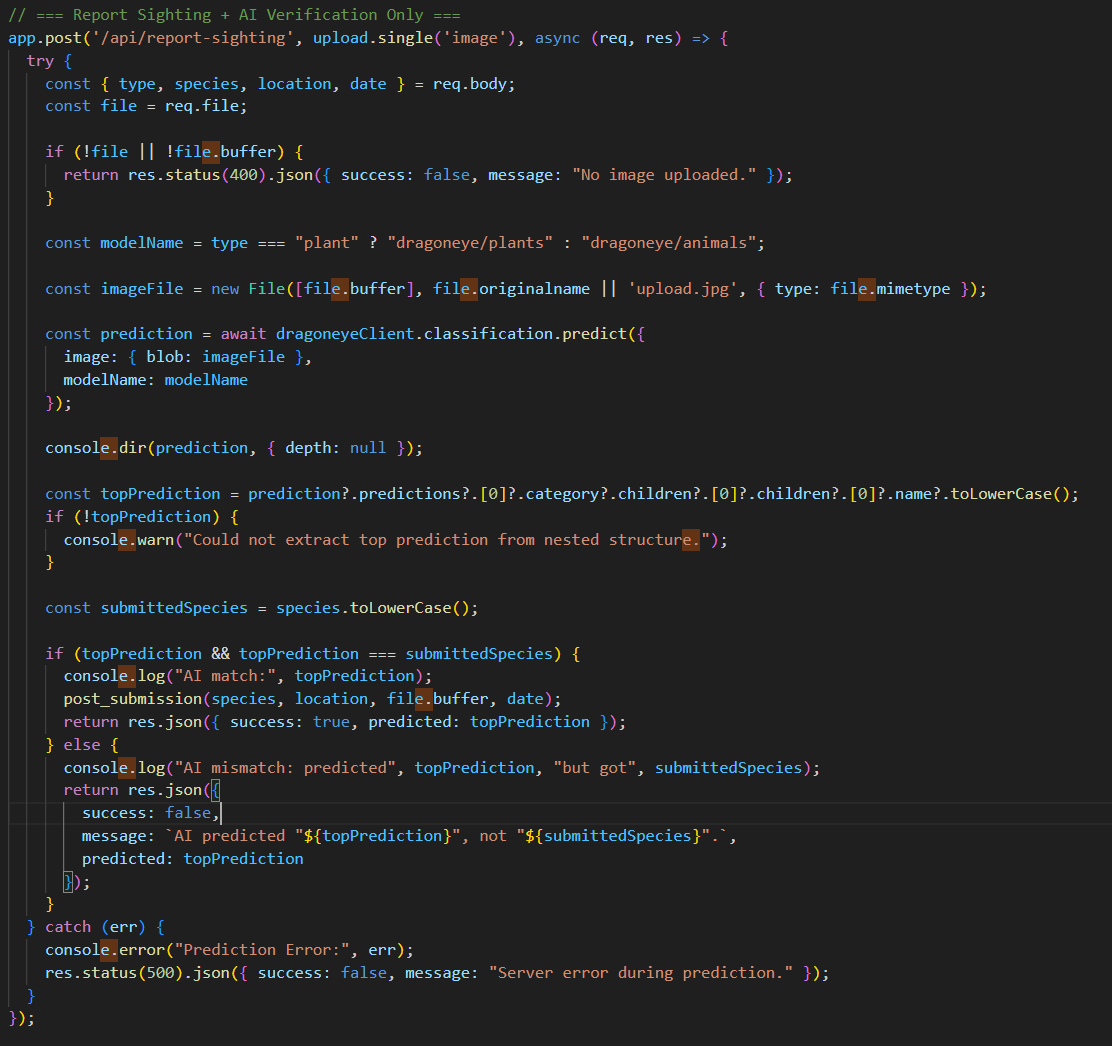
Artificial Intelligence plays a central role in enhancing data reliability within WildWatch. Our system uses the Dragoneye API, a pre-trained AI model capable of identifying plant and animal species from user-submitted images. This integration significantly improves the accuracy and scientific credibility of our crowd-sourced biodiversity data. When a user submits a sighting report that includes a photograph, the backend asynchronously sends the image to the Dragoneye API. This process is automated and runs in the background, offering a seamless experience to the user.

The API returns a predicted species name based on its analysis of the image. The system then performs a comparison between the species name provided by the user and the AI-identified result. If the two names match, the data is deemed valid and subsequently stored in the database. If a discrepancy is found, the system notifies the user, provides a suggested correction, and prompts them to revise the submission. This verification step is a crucial feature that filters out inaccurate or misidentified data, ensuring only reliable records are maintained. This is particularly beneficial in citizen science projects, where contributors may lack professional biological expertise.

As shown in Figure10.1, the AI verification logic is embedded within the server.js file on the backend. This image displays the exact logic responsible for processing the verification workflow: from parsing user input and invoking the Dragoneye API, to handling and interpreting the response, and managing error messages or corrective actions. The logic is designed to be both efficient and fault-tolerant, minimizing the likelihood of incorrect entries being stored. Moreover, the code ensures that even in cases where the API call fails or returns a low-confidence match, the system responds gracefully without crashing or compromising other parts of the application.

The Dragoneye API was chosen for its practicality: it is scalable, cost-effective (priced at $5 per 1,000 uses), and easy to integrate with our Node.js backend. With a team of five, the financial cost of implementing this solution was manageable, even at scale. During testing, we observed that while the API performs strongly for common species like cardinals, blue jays, and oak trees, its accuracy diminishes with blurry images or less common species. In such cases, the API either misidentifies the species or returns a low-confidence prediction. To mitigate these limitations in the future, we plan to fine-tune the API by retraining it with region-specific datasets, which would significantly enhance its performance within our target area—Lubbock, Texas.

The integration of AI also introduces a scalable foundation for more sophisticated data analysis down the line. With AI already filtering and validating reports at the point of entry, future system updates could include automated clustering of sighting locations, seasonal trend analysis, and alerts for rare species. By embedding AI into our core reporting process, WildWatch not only ensures quality control but also positions itself as a technologically forward and scientifically relevant tool in the field of environmental conservation.



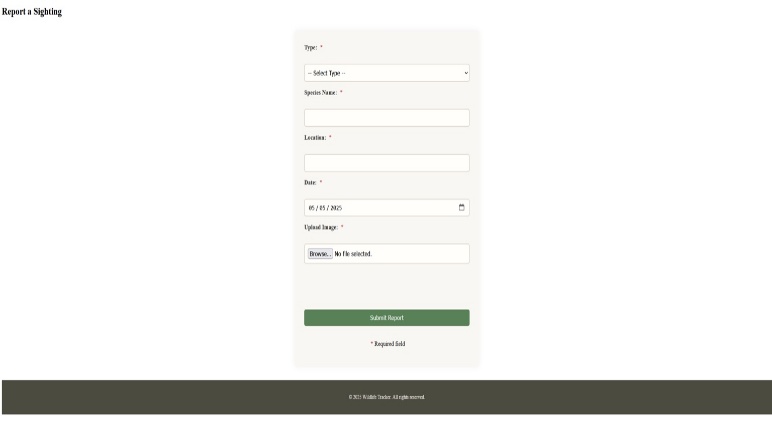
**Figure 11.1: AI Verification Logic (server.js Snippet)**

This image shows the backend logic that handles image verification using the Dragoneye API. Once a user submits a report, the backend asynchronously sends the image to the API and processes the response to determine if the user-entered species matches the predicted one. This ensures data integrity and reduces misidentification.

**XII. Frontend Development**

The frontend of WildWatch is designed to be user-friendly, responsive, and educational. Built using HTML, CSS, and JavaScript, the interface allows users to report sightings, explore a map of local biodiversity, view species information, and track bird migration patterns. Our primary goal in designing the interface was to make sure users of all backgrounds could interact with the system without needing technical expertise. A clean layout, visual consistency, and accessible controls were prioritized throughout the development process.

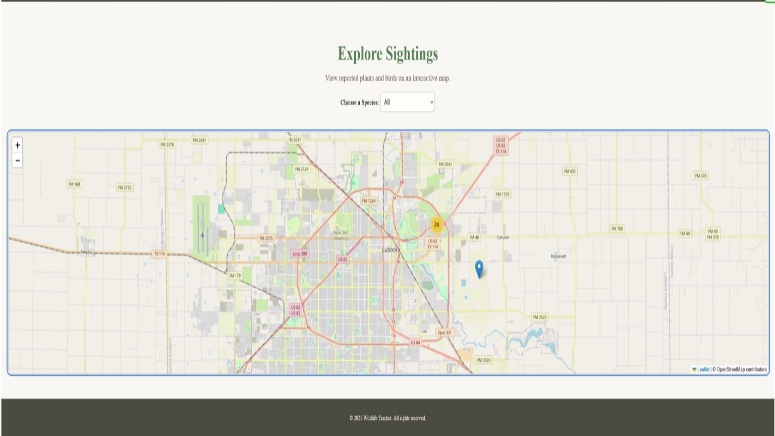
To support these goals, the team implemented multiple key frontend features. The system includes a homepage that provides users with simple navigation to all major areas of the website, including the Explore page, Migration patterns, and Report Sighting page. The Species Guide allows users to browse through entries populated from the backend database. These pages were styled using responsive design techniques to ensure compatibility across screen sizes, from desktops to smartphones.



**Figure 12.1 Report Submission Form UI**

This highlights the user-facing form where sightings are submitted. Every field is required—species name, image upload, GPS coordinates, and time of sighting—ensuring data consistency. The intuitive layout supports both desktop and mobile formats, reducing user error and improving reporting flow. The form also includes front-end validations to prevent incomplete or incorrect submissions, which enhances overall data quality and reduces backend workload. JavaScript functions dynamically handle user input, providing real-time feedback such as image previews or location autofill prompts.

Additionally, animations and subtle transitions were implemented to improve usability and visual feedback. For example, once the user submits a sighting, they receive a confirmation message or an AI-generated suggestion if there's a mismatch between the submitted and predicted species. This instant feedback loop keeps the user engaged and provides transparency into the system’s verification process.



**Figure 12.2: Explore Page Interface with Biodiversity Map**

This figure shows the interactive Explore Page. It maps user-submitted sightings as visual markers using Leaflet.js and includes filtering tools for users to search by species. This interface promotes user engagement and provides scientists with real-time distribution insights. The page also includes a search bar that queries the backend for relevant reports, displaying the species name, sighting location, and time in a clean, card-based layout. Markers on the map are clickable, showing details like species images and short descriptions.

To handle a high volume of reports without slowing down the page, pagination and lazy loading were integrated. The map dynamically updates users pan and zoom, optimizing both bandwidth and user experience. Additionally, each marker is color-coded based on species type—helping users differentiate plants from animals at a glance.

Overall, the frontend of WildWatch plays a critical role in engaging users, guiding them through each part of the reporting and exploration process, and presenting biodiversity data in an accessible and impactful format. Future enhancements may include a dark mode, multilingual support, and personalized dashboards for frequent contributors.

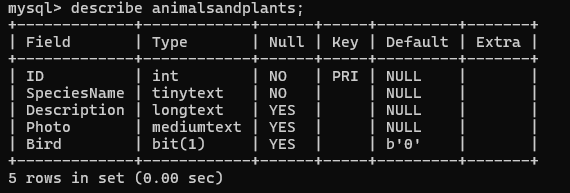
**XIII. Backend Development**

The backend is responsible for managing data flow between the user interface, database, and AI services. Built with Node.js and connected to a MySQL08 database, it handles:

* Species information and user submission storage
* AI communication and verification logic
* API routes for retrieving, inserting, and validating data

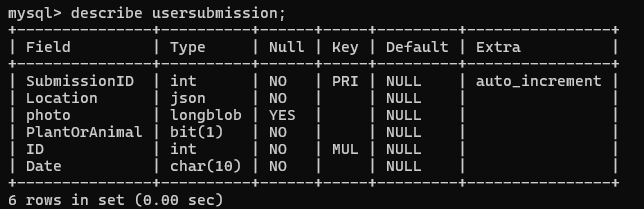
The backend plays a vital role in ensuring data accuracy, security, and responsiveness. All submission requests made through the frontend are first validated for completeness and format before being passed to the database. To maintain efficient operations, the backend was structured modularly separating API routing, database connections, and AI logic into their own components for scalability and ease of debugging.

The database includes two main tables:



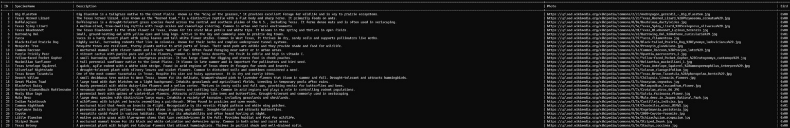
**Figure 13.1: Animals and Plants Table Design**

This figure displays the foundational animals and plants table used to populate species data. The table includes species ID, species name, image URL, and descriptions. The even/odd ID convention helps differentiate between animals and plants. This table supports accurate lookups during user report validation and improves overall backend query performance. Each entry in this table can be linked to a user submission, enabling the system to cross-reference and verify species identities with AI predictions.



**Figure 13.2: user submission Table Structure**

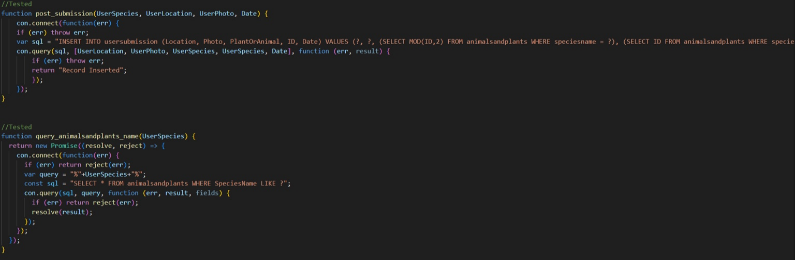
This table captures user-submitted reports and is essential for our crowdsourced data model. Each record includes a unique Submission ID, GPS coordinates, image URL, a type of flag (plant or animal), and a foreign key linking back to the species table. The figure also demonstrates how our constraints enforce complete data and eliminate null values. The use of JSON formatting for location data ensures compatibility with frontend map rendering, while the autoincrementing Submission ID facilitates traceability and avoids duplication.



**Figure 13.3 Sample Species Entries**

This figure presents a snapshot of populated records within the animals and plants table, showcasing the variety of local species tracked by Wild Watch. Each entry includes a unique ID, a Species Name, a detailed Description, a URL Photo reference, and a Bird flag for classification. These entries span flora and fauna native to Texas, including mammals, reptiles, birds, and native grasses. The structured format allows seamless integration with user-submitted data and supports AI-based validation by matching reported species against this reference set. Maintaining a curated and diverse set of species enhances the scientific utility and educational value of the platform.

Error handling is built into the backend to prevent incomplete or malformed data from entering the database. If a submission lacks any of the required fields—location, image, or species ID—it is rejected and the user is notified through the frontend interface. This strict validation improves long-term data quality.



**Figure 13.4 Submission Insert Logic (SQL Query Handling)**

Here we capture the SQL logic responsible for inserting validated user reports into the database. This logic includes conditional checks to prevent duplication and to ensure successful communication between the frontend form and backend storage layers. The backend listens for successful AI validation responses before committing the report into the MySQL database. All insertions are followed by confirmations to the user and backend logs for monitoring.

Additional backend scripts handle data retrieval for the Explore and Migration pages, delivering structured results in JSON format. These endpoints are optimized for performance and designed to support future expansion, including real-time updates and user account systems. Overall, the backend is built to be scalable, modular, and robust—supporting reliable biodiversity tracking and research.

**XIV. Conclusion**

WildWatch is a reliable and user-friendly biodiversity tracking tool that empowers individuals to contribute to environmental documentation. By combining intuitive frontend design, structured backend logic, and intelligent AI-based validation, our system offers a complete experience for users interested in learning about local species or contributing their own sightings. Our team successfully integrated all core components, from the MySQL database and Node.js backend to the AI API and interactive map display. We believe the system not only fulfills course expectations but also provides real-world value for conservation, research, and education. Looking ahead, we see the potential to expand this tool by incorporating login systems, supporting real-time data updates, scaling geographically, and improving media storage solutions. WildWatch serves as both a technical achievement and a meaningful platform for community-driven ecological engagement. With continued iteration, it can become a robust resource for scientists, educators, and wildlife enthusiasts alike.