Final Product Report

All Ears

Date: 4/28/2021

Sponsors: Dr. Chris Doughty and Ms. Jenna Keany

Faculty Mentor: Dr. Tomos Prys-Jones

Team Members: Bailey Erickson, Savannah Fischer, Zhijun Hu, Elijah Macaranas, and Jared Weinberger

Table of Contents

1 Introduction

African forest elephants, *Loxodonta cyclotis*, are a threatened subspecies of elephants that inhabit the densely wooded rainforests in west and central Africa. These elephants remove some of the abundant younger trees that clutter their forest habitats, freeing up the space necessary for larger trees to continue to grow and for the surviving younger trees to mature. This results in a greater net reduction of carbon dioxide from the atmosphere than would occur without their influence. African forest elephants are important to the health of the forests they live in, but the elephants' choice of habitat and the prominence of poaching makes tracking of the elephant population sizes in a traditional manner extremely difficult.

Our sponsors, Dr. Christopher Doughty and Ms. Jenna Keany, are researchers studying how African forest elephants affect forest structure, climate, and ecosystem functionalities. They track the poaching of elephants using tools like the Monitoring the Illegal Killing of Elephants (MIKE) database and are involved in research that helps find new and better ways to estimate the populations of forest elephants. One of the biggest issues they are facing is the lack of public awareness concerning the importance of forest elephants, the prominence of their poaching, and the effect of elephant loss on the climate and forest habitats. Dr. Doughty and Ms. Keany also work to support large conservation groups that focus on the conservation of forest elephants.

In an effort to assist with our sponsors’ endeavors, our team has developed a web and mobile application, called Elephant Footprint, with the primary goal of educating the public. Our applications include information about the current statistics and dangers of poaching, the benefits of the forest elephants and their continued survival, and provide donation opportunities for the public to assist accredited anti-poaching organizations. A carbon-calculation tool is included so the public can calculate their air travel emissions of carbon over user-defined distances. The calculator then suggests an amount to donate to conservation organizations based on the data that it calculates regarding distance, average number of airplane passengers, and carbon emissions. The user is then directed to a donation page that contains links to pre-existing conservation organizations, where donations can be made. The desired outcome of this application has been met and it would act as a tool to assist in continued survival of forest elephant populations, which will help lead to forest maturation and atmospheric carbon sequestration. In creating this app we have supplied Megabiota Lab with an integral tool for both their lab staff to use and the general public to utilize in order to learn about the impact of forest elephants on the climate and the dangers they face.

In the rest of this document, we will go over the major steps that were involved in our development process of our application. We will first look at our overview of the entire process, and that will lead into the initial requirements of our applications. We can then look at the architecture and implementation of our application and then the testing that we performed. That will then lead into an overview of the timeline we stuck to, as well as our future work that could build on our application and with that will conclude our document.

2 Process Overview

2.1 Team Interaction

Not long after the assignment of our project and our teams, all five members of what is now All Ears met to discuss and design standards and regulations for all of our future interactions. Comparing our academic schedules lead us to determine that weekly team meetings on Thursdays at 15:00 were preferred. It was decided that the meetings would begin with every member giving a concise summary of the tasks they had completed since the previous week, as well as current plans and encountered issues.

Disagreements and major decisions were to be handled by majority rule, and failure to participate in team meetings over a three week period and/or complete lack of progress with one’s assigned project components over a five week period was to result in the involvement of Dr. Doerry and the potential termination of the offending team member. Other infractions, such as being disruptive during meeting times or missing important meetings were to be handled by the group members on a case-by-case basis, often requiring written reflection, either about what was done wrong or about what had been accomplished by the offending group member that was not disclosed to the group during the appropriate times.

Thankfully, there were no major issues experienced between or because of group members, and very few meetings were not run with all members present. Those team members that did miss a meeting for some reason or another were very diligent in providing the necessary documentation of their accomplishments for the week to all other members by email. Thus, working in the All Ears team was fairly pleasant throughout the whole of the multi-semester experience.

2.2 Team Roles

All members fell pretty naturally into their roles in the group. Bailey was familiar with good leadership practices and was confident in her group organization skills; she naturally became the team lead. Savannah had experience with more focused English course instruction that made her the obvious choice for spelling and grammar checking on deliverables, as well as writing up the team meeting minutes as the team recorder. Jared was the most knowledgeable about technologies that we could potentially use in the development of our applications. His charisma also led to his becoming the contact between our team and our sponsors as the client contact. Zhijun was assigned the initial mapping components of our application in the delegation of tasks at the beginning of our development process, but because of his dedication and the in-depth nature in which he handled his assignment, he became the lead designer for the mapping components for our applications. Elijah was excellent at connecting the main topics of our deliverables into concise and easily understandable conclusions, and his impeccable visual design skills set him as the designer and editor for all of our team media, including recorded deliverables and logo designs.

Savannah and Jared had the most backend technology experience and became the backend and database developers. They also had experience with the web technologies that would be connected to the backend in a way to allow direct editability to the database components (unlike the mobile components that would be disallowed from direct backend alteration for security purposes); because of the connection between the backend and the web application, Savannah and Jared were delegated these specific project components.

Bailey, Josh, and Elijah were made responsible for the mobile application end of the project, Bailey because of her familiarity with Android application development, and the others to provide support. Because the necessary technologies and coding languages were largely unknown to the whole of the team, three team members were assigned to mobile development in an effort to reduce confusion and increase teamwork and problem solving.

2.3 Development Process

The very first meeting our team had with our sponsors ended with a complete overhaul of the initial project design. Whereas the initial project was meant to be an application for use by forest rangers to report poaching incidents they encounter, our newest project was adjusted to be an educational web and mobile application pair meant to inform the public about the importance of African forest elephants, about the carbon emissions they've produced by taking flight-based trips, and to encourage public donations to accredited anti-poaching organizations that have a focus on African forest elephants.

The foundation of our development process was based on the Agile Development Method: plan, design, develop, test, deploy, review, and launch. We planned out our application by collecting the necessary requirements and ideas of the expected final product from our sponsors. Once we had designed the blueprint prototype by investigating various potential technologies and selecting those that worked best with the vision we held of our final product, we started to develop the main applications. During development, our team regularly tested each component for compatibility with our gathered requirements, making adjustments until our sponsors were satisfied with the direction we were taking and the product components we were demonstrating. After we had completed the beta version of our product, we prepared it for usability testing with our target audience - the general public - and our mentor and sponsors. After we reviewed the prototype feedback, changed aspects of our product according to relevant suggestions, we then packaged our mobile application and sent it to our team's Google Drive folder, made specifically for product delivery, where the mobile application can be downloaded. Our web application, with help from our sponsors, Dr. Doughty and Ms. Keany, was assigned a domain name and is accessible to everyone through the URL "elephantfootprint.org".

2.4 Tools and Artifacts

The major tools we used can be split into three major sections: the mobile front end, the web frontend, and the backend database. Each of the sections had a designated tool chain that we used in order to create our final product. The major mobile front end tool chain was Flutter with Syncfusion, whereas the web front end utilized Vue. The backend component tool chain included Flask to construct the backend architecture and MariaDB as the structure for our backend database. The DigitalOcean Droplet virtual machine would also be considered part of the backend tool chain, as it was used for our backends accessibility to the deployed final applications.

2.4.1 Mobile Front End

All Ears used several tools for the development of the mobile front end. Our major framework was Flutter to develop the mobile application, chosen because of its extensive online documentation as well as its support of cross compatibility between Android and iOS devices, which was a requirement from our sponsors. For the mapping portion of our mobile application we used Syncfusion; this allowed us to highlight and outline countries on the mapping components of our applications, as well as plot lines on the map for the carbon calculator. Syncfusion also allowed us to make the chart that goes along with the elephant poaching map, making it simple and easy to integrate.

2.4.2 Web Front End

The web front end was developed in Vue, which we chose because of team familiarity as well as its extensive documentation. This made it simple and easy to make a web application that would look and function similarly to the mobile app, which had been developed first. Highcharts was used in the elephant map to develop the interactive map with the selectable countries in a heat map, as well as the accompanying poaching graph that is displayed when the countries are selected. The carbon calculator page used Leaflet in order to render the manipulable world map and draw lines between the user inputted airports in order to provide better visualizations of the flight paths flight distances.

2.4.3 Backend Database

The backend component of our system is responsible for making the elephant poaching data available to the web and mobile components of our system. The backend is hosted on DigitalOcean Droplet and is divided into two parts: the publicly accessible portion that uses Flask, and the MariaDB database that stores the data in a password protected manner.

3 Requirements

Throughout the 2020 fall semester, we met with Dr. Chris Doughty and Ms. Jenna Keany about three times a month, and held weekly meetings with our mentor, Mr. Tomos Prys-Jones. During these meetings, we obtained more detailed requirements and created more possible use cases to better understand the functions brought about by these application requirements.

3.1 Functional Requirements

The functional requirements are the specific product features/functionalities that developers must implement to enable their products to accomplish the domain-level requirements that were set with Dr Doughty and Ms. Keany. We start with a broad branch of requirements, and then focus on selecting some critical functional requirements to explain in depth.

3.1.1 Display Elephant Poaching Data -- Interactive Map

Our application allows users to view the general regions where forest elephant poaching events take place. We used Syncfusion to create an interactive map that allows users to select the various regions to display the elephant poaching event information. The map displays a graph that represents the poaching incidents according to a timeline of approximately twenty years for the specific region that the user is currently viewing, so long as the data is available.

3.1.1.1 Selectable Countries on Map

One of the main features that our Interactive map includes is selectable countries. By default, the map displays all countries located in Central Africa, the primary location where African forest elephants can be found. Users can tap each country, after a user selects a country, the map displays that region's poaching statistics.

3.1.1.2 Poaching Incidents Graph

Once the user selects a country, the map zooms in to the specific country and brings up a small window that contains a line graph of poaching incidents based on the last recorded year. The graph shows the number of poaching incidents per datapoint and the trend of poaching during recorded years. The incidents graph changes based on the country selected and remains visible until the page is reloaded.

3.1.2 Display Environmental Impact -- Carbon Calculator Tool

In order to aid in the decision making of donation amounts our application has an integrated carbon calculating that can suggest a donation amount.

3.1.2.1 Calculate Flight Distance

The user selects two cities with airports, a starting point and an ending point, from a drop-down list that lists them alphabetically, which results in an estimated displacement distance using an airport distance API. Next, using MapBox, the system displays the route of the flights to the map, then the API uses the chosen airports to estimate distance traveled by plane. Subsequently, we implement a formula provided by our clients and developed from consulting similar tools on other websites to calculate carbon footprint by miles traveled by plane, among other considerations.

3.1.2.2 Offset CO footprint from flight information

The application implements a formula from the clients to calculate elephant carbon sequestration so that it can determine how many elephants will counteract the carbon footprint that results from the user’s inputted travel information. And from that, the application estimates how much money should be donated to an anti-poaching organization so as to fund the efforts necessary to save the approximate amount of elephants calculated.

3.1.3 Suggested Donation Links

Our application also provides the links for the user to go and donate to either from the home screen or from the carbon calculating tool. Since our application will not be handling the user’s personal banking information we linked them directly to the donation pages.

3.1.3.1 Donation Prompt

For users to donate before or after calculating their CO footprint offset, they can go on the page listing the anti-poaching organizations, then users will be presented with donation links with or without their calculation from the CO calculator for a voluntary opportunity to visit the individual donation portal‘s of the organizations listed below.

3.1.4 Provide Current and Accurate Data -- Administrator Data Curation

In order to keep the data for this application up-to-date and accurate, we needed a trusted administrator to curate our datasets. This includes adding new data from the MIKES database, which has no public Application Programming Interface (API), and removing incorrect data. This was only implemented in the web application.

3.1.4.1 Access Restriction

In order to access the section of the website where one can change the stored data in any way, one must first enter a master password. After one has entered the password, the administrative functions mentioned in the following sections can be accessed.

3.1.4.2 Data Upload

An authenticated administrator can upload a CSV file with the same format as one from the MIKE database available. This file was then parsed and added to the application database. Duplicate records were automatically removed, so one can either upload all of the data from the MIKE database with the new entries included or one can upload only the new entries, which one must filter for oneself.

One can upload a CSV file from the MIKE database and have its records entered into the application database. Any uploaded records which have the same unique ID (the MIKE site ID and the year) as a record in the database was replaced by those records in the database.

Any file that is either not a CSV file or one not having the same structure as the CSV available from the MIKE database available on the 5 November 2020 from the URL 190307_PikeStatsUpTo2018FusionTableFormat.csv was be rejected.

3.1.4.3 Data Revision

An authenticated administrator can view the records in the application database and make changes as needed; the administrator can add single records to the application database, correct fields in single records in the application database, and remove single records from the application database.

3.1.5 Provide Current and Accurate Data -- Application Database

The database must be altered from its default state in order to read and handle CSV files, as well as to function most effectively in the use scenarios implicit with creating a web and mobile application - ideally expecting traffic and application component usage from various devices simultaneously throughout the world.

3.2 Non-Functional Requirements

Accuracy and speed were the two qualities by which our application was measured apart from basic functionality. The quality of performance was also determined by the usefulness of the app. A calculator program, for example, is useless if it takes five minutes to compute 1 + 1 on a modern computer. One could say that it has that functionality, but because of the performance costs, that functionality is negligible.We show the list of our non-functional requirements below by considering that users may have never used an application like Elephant Footprint before.

3.2.1 Data Accuracy

As stated above, the data accuracy of our project would be determined by those for whom we are creating this application who are experts in the field. The accuracy of the data presented depends on their curation of the application's database and the MIKES database primarily from which our application gets its data. We, as the developers of this application can only ensure that the data served from this application is the same data that the administrator manually enters into the database or that an administrator uploads in the form of a CSV file. There is no discrepancy between an uploaded CSV file and the data in the application's database. Our team has tested this by selecting a MIKE site at random and making sure that the numbers displayed in the application are the same numbers that are in the uploaded CSV.

3.2.2 Timing Characteristics

The response time of our application would depend largely on the quality of the internet connection between the backend mobile whichever frontend, web or mobile, is being used. Therefore for all performance stipulations, we would require that the internet connection throughput for all parties be at least 100 Mb/s for download and 10 Mb/s for upload.

Additionally, the database would not be directly considered in this section, as there would be no direct user interaction with it. Its performance is integral to the rest of the application because the performance of any part of the application involving poaching data directly depends on it. Therefore, any requirements placed on the parts of the application that depend on the database would effectively constrain the database as well.

3.2.2.1 Display Elephant Poaching Data

It would take new users 10 seconds or less to access the fully-loaded interactive poaching map and 3 seconds or less to access the poaching graph of a country or MIKES site after that. Experienced users should reduce those times to 3 seconds or less and 1 second or less, respectively.

3.2.2.2 Carbon Calculator Tool

The carbon calculator tool has two components: calculating flight distance and calculating the carbon footprint of that flight. A new user would take 40 seconds or less to calculate their flight distance and calculate the carbon footprint of that flight. An experienced user would reduce that time to 15 seconds or less.

3.2.2.3 Suggested Donation Links

A new user needed to be able to find the links suggested for donation within 10 seconds after opening the application. An experienced user should be able to find the links within 5 seconds or less after opening the application.

3.2.2.4 Administrative Features -- Login, Data Upload, & Data Revision

Login **-** A new administrator needed to be able to login and gain access to the restricted administrative section of the application within 20 seconds. An experienced administrator should be able to do the same within 10 seconds.

Data Upload **-** A new authenticated administrator needed to be able to upload a CSV from MIKES database within 15 seconds or less. An experienced administrator should be able to reduce that time to 10 seconds or less.

Data Revision - A new authenticated administrator needed to be able to add a new record to the database from the web application within 40 seconds or less. An experienced administrator needed to reduce that time to 20 seconds or less. Deletion for both administrators would only take 10 seconds or less. Changing a record would take 40 seconds or less for a new administrator and 20 seconds or less for an experienced one.

4 Architecture and Implementation

In the next section we discuss the architecture of our system as well as how we went through with the implementation of our system's architecture. In designing our system, we thought it best to create a frontend/backend separation architecture. This way, we could serve our application with information for a backend seamlessly as well as keeping our frontend looking professional and aesthetically pleasing.

4.1 Architecture

Together with Dr. Doughty and Ms. Keany, we developed an overall design for our final product, which included a mobile and web application for displaying the negative impacts of poaching and give users an opportunity to offset their carbon emissions by donating to elephant conservation organizations and a database for displaying all of the information. We would go into greater detail in the next section on the implication of each.

4.1.1System Components

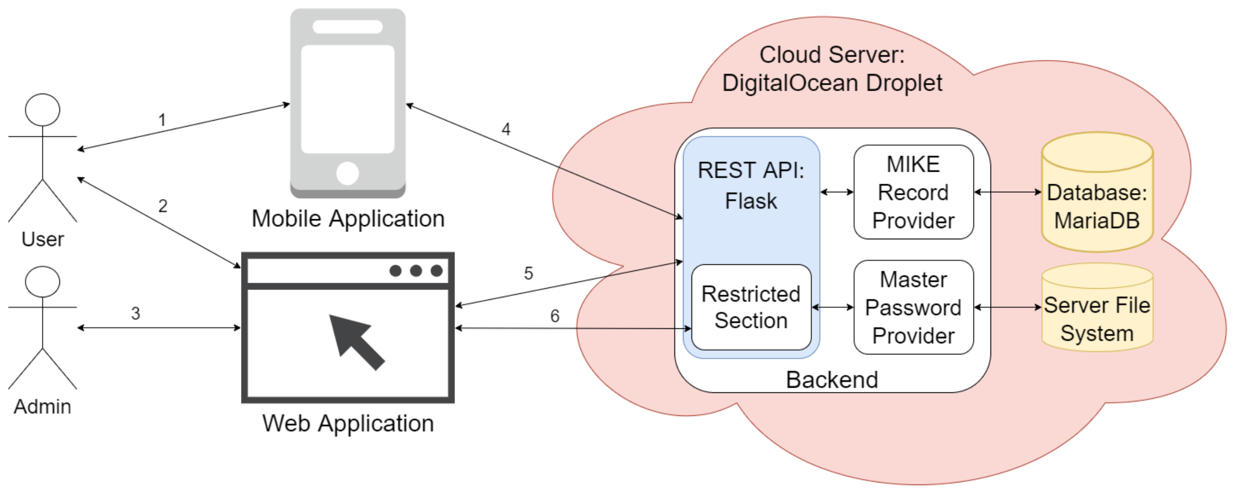


Figure 4.1.1.1 - A high-level architectural diagram of our applications functionality

As displayed in Figure 3.1, our final product is made up of three major components: a web application, a mobile application, and a backend that supports both the web and mobile applications. The web and mobile applications both interact directly with the users as displayed by lines 1 and 2 and display the information that is critical to our product. The backend supplies the mobile and web applications with data, as is seen in lines 4 and 5.

Both of the applications provide users with information about forest elephants and how they are integral to the ecosystem of the region, an interactive map showing the poaching data of countries where they live, and graphs for each country showing elephant poaching numbers over time. It also has a carbon calculation functionality that allows users to see their carbon emissions based on air travel and suggests donation based on that emission and also includes links of organizations to donate to.

The web application has one additional functionality that the mobile does not: the administrative portal. Administrators use the password-protected admin portion of the web application to curate the elephant poaching data and to make sure the data stays current and accurate as seen by line 6 in Figure 4.1.1.1. They can login in order to: update the dataset, manually correct errors in the dataset, and add incidents that may not be in the current data provided by the MIKE database. An administrator can also use the admin portal in the web application to trigger an automated update of the application database, where the dataset provided by the MIKE database is uploaded into the local database utilized by our application.

The backend, as mentioned above, supports the mobile and web applications by providing the necessary data for the two to display poaching statistics, update the elephant heat map, and other relevant information. It also provides a major part of the security for the admin section of the web application, validating or rejecting login attempts in order to restrict who is allowed access to the database.

4.1.2 Component Communication

The mobile and web applications communicate with the backend via a REST API as seen in interactions 4, 5, and 6 in Figure 4.1. A REST API is a special way that applications with a similar structure to ours communicate using the Web. Clients request the server to send out information or perform an action. The greatest characteristic of this strategy is that each request from a client (our web and mobile apps) contains everything necessary for the server (our backend) to interpret it, meaning that the server does not need to keep track of its clients.

When either the web or mobile apps display the poaching map and subsequent graphs, they need to send requests to the backend to acquire the relevant data as seen by lines 4 and 5 in Figure 4.1.1.1. When a user tries to gain access to administrative features as displayed by line 6 in Figure 4.1.1.1, by logging in, the web application sends the password to the backend to have it checked. Every time the web app tries to do any administrative action, it sends a token which is then validated before the backend fulfills its request. When poaching data is to be updated, added, or deleted, the web application sends a request to the backend to update the application database.

4.2 Implementation

In order to develop our finished product, we split each module into its own section: the mobile frontend, the web frontend, and the backend database. Each member of our team had found their place in production between one or more of the modules. Our general implementation plan was split into two major phases: the development phase and the testing phase. We will talk further about the development phase here.

The development phase contained Vue, Flutter, Flask, MariaDB, and DigitalOcean Droplet components. We started by working on the Flutter based components because they were the most complex and were what our team had the least amount of experience with. As the semester progressed, we began developing all of the aspects of the project simultaneously; though they were all codependent on specific other components of the project (ie the Elephant Map required the database to be connected and the database would be editable through administrative features on the web, etc.), their basic functionalities were easy to develop concurrently.

Vue was the primary component of the frontend web application, as it utilized Javascript to effectively create dynamic experiences on the web page. Because cross-compatible mobile application development was not a well-practiced or well-known skill for many of our team members, we prioritized simplicity when searching for the technology necessary to build an interactive and visually pleasant mobile frontend, which we found in Flutter.

We used Flask to build the backend JSON REST API, meaning that we used it to respond to requests by sending data in the JSON text format without storing any information specific to the client across requests.

For our backend database architecture, All Ears team members had the most experience utilizing MySQL databases. However, since MySQL was deprecated, MariaDB was selected, as it was the successor to MySQL and was structured the most similarly and was supported greatly with documentation and potential company troubleshooting support in the event that our team encountered an unexpected and indiscernible error.

In order to have the backend database accessible by deployable and publicly accessible applications, it needs to be hosted on its own server. As All Ears does not have its own central location with its own servers, our team was required to utilize a pay-to-access virtual server on which to host our database, which is a DigitalOcean Droplet.

5 Testing

Testing was an integral part in the delivery of our product - ensuring that every piece of the application worked before we put them together and then making certain that they worked once assembled.

5.1 Unit Testing

Unit testing is a thorough method of testing the functionalities of small code units before integrating them into the final product. This ensures that the small elements of the entire product can work properly. In our unit tests, we only focus on the specific parts of the application that have the most features and the most user interactions. In addition, we did not spend much time on our donation pages or about us pages on the web and mobile application because they are dynamic and do not require more functionality than simply that they can display. For the part of the code we tested, we first looked at the elephant map and carbon calculator for web and mobile applications. It was vital to ensure that each of these parts worked individually to ensure that they all worked together in the end.

5.1.1 Elephant Map for Web and Mobile

- Hovering the mouse over each country without first clicking a country in the web application should display the latest number of elephant deaths recorded in the virtual API for poaching in that country.
- If no country/region is selected, the color of all countries/regions are on a scale from green to red. Compared with other countries/regions in the elephant test chart, it indicates the severity of elephant poaching deaths. Green is low, yellow is moderate, and red is many - according to data from the virtual API, the number of elephant poaching instances that influence the heat map is related to the records of each country's most recently recorded year.
- Clicking on a specific country/region displays a graph of elephant deaths due to poaching over time, starting with the first recorded instance of poaching and ending with the most recently recorded year in the virtual API.
- After clicking a country/region, hovering over each country/region on the elephant map displays the most recent number of elephant poaching deaths recorded in that country/region, while the displays chart remains specific to the country that was selected.
- After selecting a country/region, the country/region was grayed out on the elephant map, indicating that the country/region has been selected. Even if its color is no longer displayed, other countries are colored accordingly to show how the severity of elephant poaching deaths in the elephant test chart is compared with other countries (including selected countries/regions). Even if the selected country no longer has a color, the number of poaching deaths in the selected country was still taken into account.

5.1.2 Elephant Map for Web and Mobile

- From a fresh page with empty locations, press the submit button. The form does not submit and no action is taken on the page.

- From a fresh page, enter one location suggested by the dummy API through one of the drop-down lists and press submit. The form does not submit and no action is taken on the page.
- From a fresh page, type in a custom location in one of the input sections not suggested by the drop-down lists and press submit. The form does not submit and no action is taken on the page.
- From a page with one suggested location in the input section, enter a custom location not suggested by the drop-down list in the untouched location input section and press the submit button. The form does not submit and no action is taken on the page.
- From a page with a custom location in one of the input sections not suggested by the drop-down list, enter a suggested location in the untouched location input section and press the submit button. The form does not submit and no action is taken on the page.
- From a page with a custom location in one of the input sections not suggested by the drop-down list, enter another custom location not suggested by the drop-down list in the untouched location input section and press the submit button. The form does not submit and no action is taken on the page.
- From a page with one dummy API-suggested location in the input section, enter the very same location in the untouched location input section and press the submit button. The form does not submit and no action is taken on the page.
- From a page with a suggested location in one of the input sections, enter another, different suggested location in the untouched input section and press submit. The form does submit and the page receives two latitude, longitude pairs specific to the two locations. The map on the page displays a dotted line connecting the two latitude, longitude pairs in a mimicry of a flight path. A table appears below the map detailing the minimum distance between the two locations, the carbon emitted from the flight, and the recommended donation amount. A note also appeared below the table indicating the amount of ‘elephant years’ required to sequester the amount of emitted carbon shown in the table.
- From a page with two valid location entries that were already submitted and a dotted line indicating the flight distance between the two latitude, longitude pairs of those locations, type in a custom location in one of the input sections not suggested by the drop-down lists and press submit. The form does not submit and no action is taken on the page.

- From a page with two valid location entries that were already submitted and a dotted line indicating the flight distance between the two latitude, longitude pairs of those locations, type a custom location in both input sections and press submit. The form does not submit and no action is taken on the page.
- From a page with two valid location entries that were already submitted and a dotted line indicating the flight distance between the two latitude, longitude pairs of those locations, enter a different valid location in one of the input sections. The system returns the two latitude, longitude pairs of both locations and a new dotted line connecting the previously existing valid location to the newly entered valid location. A table should appear below the map detailing the minimum distance between the two locations, the carbon emitted from the flight, and the recommended donation amount. A note should appear below the table indicating the amount of ‘elephant years’ required to sequester the amount of emitted carbon shown in the table.
- From a page with two valid location entries that were already submitted and a dotted line indicating the flight distance between the two latitude, longitude pairs of those locations, enter a different valid location in both of the input sections. The system returns the two latitude, longitude pairs of both locations and a new dotted line connecting two newly-entered locations. A table should appear below the map detailing the minimum distance between the two locations, the carbon emitted from the flight, and the recommended donation amount. A note should appear below the table indicating the amount of ‘elephant years’ required to sequester the amount of emitted carbon shown in the table.

5.2 Integration Testing

Our integration tests were based on the exchange of information between our backend and our two applications’ frontend, as well as the communication between our applications and the online APIs that we pull data from. If the frontend receives incorrect data from the backend or the API, there were several issues the users would encounter while using our applications. Testing the applications’ integrity would not only provide fail-proof applications for our users but also guarantee the functionality for our mobile and web applications.

We tested our applications’ three main modules based on the Software Testing Plan deliverable developed earlier in the semester. The admin portal's functionality in changing the backend database when manipulating data which then was reflected in the applications. Then the elephant map on both the web and mobile application and its ability to query data from the database and display it in a graphical depiction of poaching incidents was tested. Finally, the carbon calculator was tested for its ability to send API queries to IATA Airport API for the airport names and IATA codes to get necessary location information of an airport.

5.2.1 Admin Portal Integration Testing

The admin portal's integration testing was based solely on an admin's changes through the admin portal to the database. So if the admin tries to manipulate the values on the database, whether updating, removing or creating values, then we expect the changes to be reflected on the elephant map values on both mobile and web applications. The following tests performed are the following:

- Updating the database automatically through the MIKEs website is capable of adding another year's records into the database, or otherwise altering previous data that may have been mislabeled or changed previously.
- Removing a record in the database removed a year from the country whose records were deleted from, from the graph of that country on the elephant map. If the record removed was the most recent record for that country, the heat map of the elephant map page would also be affected, as the next most recent year would become the influence for that country's heat map color.
- Removing a set of data regarding a specific country in the database removes an African country from the elephant map.
- Creating a record in the database added a year and value to the graph for the African country affected.
- When manipulating data, a tremendous change in the poaching numbers resulted in the heat map changing its color distribution as intended, the highest having the color red and the lowest having shades of green, as indicated on the heat map's key.

5.2.2 Elephant Map Integration Testing

In order to verify the integration of the Backend API for both mobile and web elephant map modules, the following is necessary:

- The elephant map uses the Africa.json file to shape central Africa regions when the users interact with the map. Simultaneously, the elephant map module sends a query request to the Backend API when countries are selected.

- As the API passes the queried data back to the elephant map, the data from the backend holds a list that displays all the requirement information for African countries for display.
- The elephant map module correctly completed all of the unit tests involving data from the API.

5.2.3 Carbon Calculator Integration Testing

The carbon calculator map does not necessarily need integration testing with other modules of the application, but it needs integration testing with its other connections with the airports.json asset we have and the connection it has with the online airport API we utilize to get airport information.

5.2.3.1 Airports.json

We use the airports.json file to get the necessary information to query the user input into our search dropdown box. We tested the airport.json's integration according to whether the information showed up or not at all based on the input. Before testing, we noted a random airport's information that is on the json file and did the following:

1. Inputting the noted airport's city/location/IATA code on the dropdown search bar allowed the noted airport's information to be provided, rightly, as a suggestion.
2. Inputting any combination of words or letters on the origin dropdown search bar did not yield any information from the json file.

5.2.3.2 Airport API

We use the airport API to get information about an airport. The API also connects to the json file, which is connected to the input. Because the input is automatically converted to an IATA code on the json file, we tested the API's integration by testing if the input would show us the correct information between two airports and the necessary information we need. To test, we noted if the following actions resulted in the defined actions:

1. Selecting two endpoints and pressing calculate resulted in a line drawn on the map and displayed a table below that has the accurate names of the two airports, the calculated distance, CO2 emissions emitted, and the calculated donation.
2. Selecting two endpoints and pressing calculate resulted in an error message being displayed, saying “AirportName does not have any location information available”. Meaning that the airport's information is not on the API.

5.3 Usability Testing

Usability testing is the process of testing the functionality of an application from a user's perspective. Since our application will be mainly used by the public and potential researchers, we needed to ensure that the usability of the end user was an important part of our testing. We needed to make sure that our products were easy to navigate and all features were very clear to both new and old users. If our application was unclear or difficult to use, it would not enable many people to understand the status and vitality of forest elephants.

For usability testing, we had two user groups to consider: the general public and administrators. We needed to design tests for each of them. Members of the general public needed to be able to effectively navigate the public portions of our application . They would need to be able to operate the elephant map and the carbon calculator effectively. It was imperative to test younger users (ages 18-35), middle aged users (ages 35-55), and older users (ages 55+) .

Administrators needed to be able to both easily navigate the public portion of our app and use the restricted admin portal. They needed to be capable of changing, adding, and deleting individual MIKE records and to both start automatic updates and upload CSV files that are in the valid format to update the application. The field of administrators are currently limited to our sponsors Chris Doughty and Jenna Keany, along with any peers they wish to involve in this project.

In order to test usability, we designed tasks for each user group, each testing a certain portion of our application. All tasks started with the user at the home page of the application with the exception of administrator tests, which began after the login test. Tasks for all users were the same for the web and mobile versions of our application. Tasks for administrators were only done on the web version of our application. The tasks began after the user had been given a short time to familiarize themselves with the app.

5.3.1 Usability Testing Task for User

- Navigation
    - Navigate to the about page
    - Navigate to the donation page
    - Return to the home page

- Elephant Map
    - Navigate to the elephant map page
    - Select a country
    - Scroll down to the graph
    - Deselect the country
- Carbon Calculator
    - Navigate to the carbon calculator page
    - Calculate the carbon emissions of a flight and the recommended donation
    - Go to the donation page

5.3.2 Usability Testing Task for Administrators

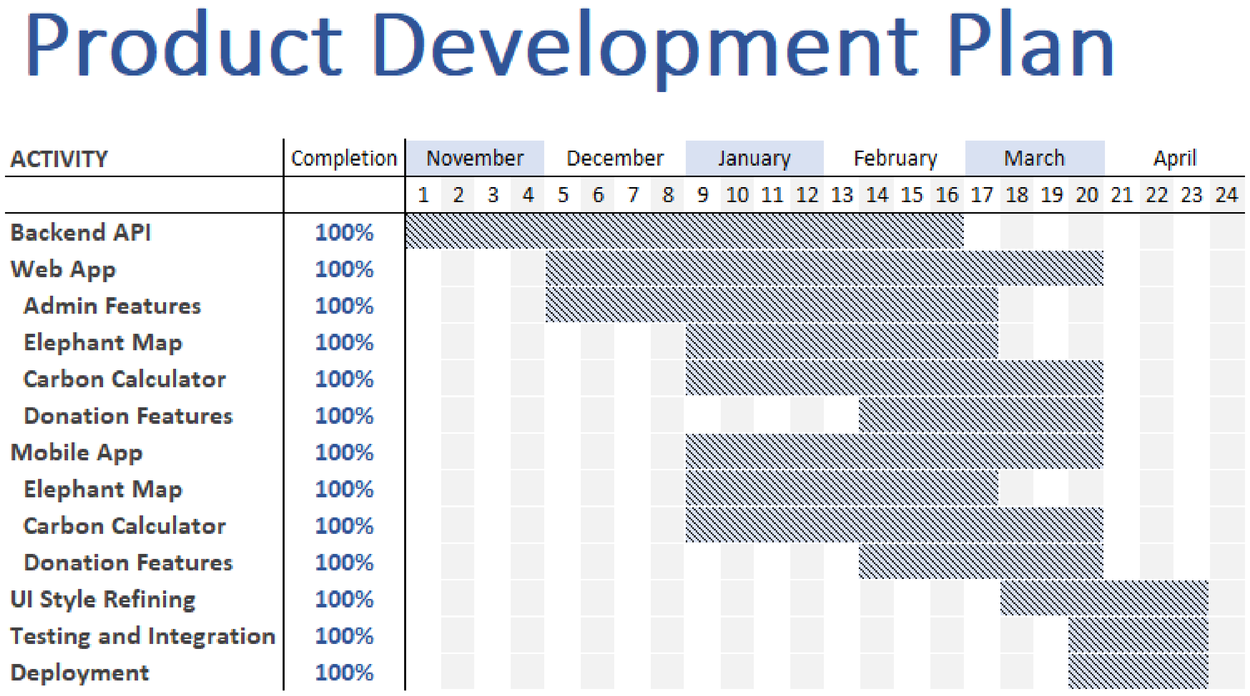
- Login
    - Navigate to the login page
    - Login
- Edit Records
    - Navigate to the edit records page
    - Delete three records
    - Add two records
    - Change three already existing records
    - Save the changes
- Upload Records
    - Download MIKE records as a CSV from https://cites.org/sites/default/files/MIKE/2020-05-14_PIKEStatsUpTo2019FusionTableFormatCITESWebPage.csv
    - Navigate to the upload records page

- Upload the CSV
- Submit

- Automatic Update
  - Navigate to the upload records page
  - Start an automatic update
  - Navigate to the edit records pag

6 Project Timeline

Figure 6.1 below shows our development timeline for the Elephant Footprint application, most of which was done in the second semester of Capstone. The non-indented items to the left are major tasks and the indented ones are subtasks.



ACTIVITY	Completion	November				December				January				February				March				April			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Backend API	100%	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Web App	100%					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Admin Features	100%					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨							
Elephant Map	100%									▨	▨	▨	▨	▨	▨	▨	▨	▨							
Carbon Calculator	100%									▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Donation Features	100%														▨	▨	▨	▨	▨	▨	▨				
Mobile App	100%									▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Elephant Map	100%									▨	▨	▨	▨	▨	▨	▨	▨	▨							
Carbon Calculator	100%									▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Donation Features	100%														▨	▨	▨	▨	▨	▨	▨				
UI Style Refining	100%																		▨	▨	▨	▨	▨	▨	
Testing and Integration	100%																				▨	▨	▨	▨	
Deployment	100%																				▨	▨	▨	▨	

Figure 6.1 - A Gantt Chart displaying our progress on the project

Our development time stretched from the beginning of November to the end of April. A three month planning period from September through November both preceded it and overlapped it. That time was taken up in understanding the problem, gathering requirements, sketching the general form of our solution, and planning our development timeline.

One month before the end of the first semester work began on the first of our application's components, the backend API. The backend API was developed first because all other components of our application depended on it. We worked on it all throughout winter break to make sure that most of its functionality was available before work on the mobile application began. We continued working on it until February, making slight adaptations as our overall application evolved.

We started the web application as soon as the first semester ended. The admin features were the first target for development. As they were the most complicated part of development collectively, they took the longest. At the beginning of the second semester of Capstone we started working on the elephant map and the carbon calculator portions of the web application. The elephant map was completed first, then the carbon calculator at the end of March. In the first part of February we started the donation features. They had a slight overlap with the carbon calculator, as the carbon calculator suggested donation amounts, so it ended at the same time as the carbon calculator. With those last components, the web application was finished.

We started the mobile application at the beginning of January. All of the subcategories for the mobile application had the same schedule as the corresponding ones in the web application. This was because roughly the same algorithms were implemented in different languages and frameworks. Given that fact, we finished the mobile application at the end of March at the same time as the web application.

We began refining the user interface of both the frontend applications (web and mobile apps) during the second week of March as both the apps became more fleshed out and more of their functionality were completed. The largest changes were completed in March as we made changes to the mobile application's content and did a total overhaul of the web application to make it mobile friendly. We ended the user interface style refining at the end of the third week of April.

We started the testing and implementation portion of our project at the end of March. "Testing" in the category name refers to integration and usability testing, as we had technically been doing unit testing all throughout the previous development phases. After integration testing, we actually integrated the application components together. That lasted until the 3rd week of April.

Deployment started during the same week as testing and implementation.The web portions of our application (the backend and the web frontend) were deployed first and developed iteratively as testing revealed more bugs to be squashed. The mobile application was not deployed until the very end during the third week of April.

After looking back on the six month development period for the Elephant Footprint application, we look now to how the application can grow in the future.

7 Future Work

We have accomplished the minimum valued product our sponsors had assigned us to implement, throughout our development process we discovered goals that were unattainable in our time frame. These goals could expand further into a whole extension of another project for future capstone groups. Here are several of the ideas that we discussed with our mentor and our sponsors during and after the development of our project.

7.1 Expansion to Include Asian and Savanna Elephants

In our project, our client hopes that our program can include more elephant data in the future. Asian elephants and Savanna elephants and our program's targeted Forest elephants occupy a considerable proportion in Africa. For this reason, our group discussed that this must be one of the important features of our program 2.0.

7.2 Conversion to More Languages

The other feature that could be improved for our team project is conversion to more languages. Our clients, Dr. Chris Doughty and Ms. Jenna Keany, wish to let more people use our application in the future and by the universal design recommendations, the best way to improve usability is conversion to more languages, such as French, which is spoken in a great number of the countries where African forest elephants exits.

7.3 More Carbon Calculator Functionality

The additional function of the Elephant FootPrint app is to add a flight transfer function to the carbon calculator page. Unfortunately, at present, the carbon calculator function of our program can only provide accurate flight data and carbon emissions for direct flights. However, long-distance flights are generally non-direct flights. Therefore, we need to improve our carbon calculation page to make our program closer to reality.

7.4 Additional Page to Change Password for Login

Another feature that could be improved in the future is to add the password change Tab. in the admin portal to allow the administrator to change the administrator password in the web app. Now, our website only allows us to change user passwords in the background. This step is cumbersome and may confuse administrators who don't know the console. Therefore, improving this problem can effectively improve the overall usability of our software.

7.5 Auto-fill Form Feature for Editing Database Records

The final feature that could be built on is the database auto-filling associated columns to match with a valid, inputted country name or code. This feature would be relatively easy to implement and would also reduce some of the tedious actions involved with editing the database records.

8 Conclusion

The continued survival of African forest elephants is imperative for African rainforests to thrive, and thus for their continued contribution to the steady removal of carbon emissions from the air. The true scale of forest elephant poaching is often unknown by members of local and world-wide populations. To educate the public about the positive environmental impact of the elephants, the threats they continue to face, and the ways of contributing to their protection through anti-poaching organizations, All Ears designed a mobile and a web applications with the purpose of doing these things.

With our product, Elephant Footprint, we ultimately aim to assist in the prevention of the indiscriminate poaching of African forest elephants by drawing attention to several pre-selected organizations by our sponsors. Overall, our applications has these main components and features:

- **About page** - a place in our application which consists of general information about our sponsors' research work, the elephants' role in the environment, and why users should donate to the elephants' cause. This serves as an introduction to our users as to what our application's purpose is.
- **Elephant Map** - an interactive map of the countries that are spanned by the habitat of African forest elephants and displays relevant information and poaching statistics about the various countries

- **Carbon Calculator** - a calculator that utilizes a mathematical function designed by our sponsors that determines a general estimate of a donation amount created by travelling a user given distance by flight and the carbon emitted by that flight
- **Donations Links** - hyperlinks to anti-poaching organizations that allow user the opportunity to donate to their cause
- **Admin Page** - only for our web application, that allows an administrator to manipulate data or update the database annually
- **Database** - where information from the MIKE database is stored independently, along with any manual alterations made by those with access to our product's admin page, and can be accessed by our web and mobile applications to display poaching statistics and relevant information

With our final product, we hope to have solved the problem of our sponsors and hope that it will not only further their research work, but will open up new opportunities and perspectives for educating the public of the importance of African forest elephants. Our sponsors can also expand our applications in their current states to include other regions that are home to more than just the African elephant subspecies.

Overall, this capstone project and the creation of Elephant Footprint was a tremendous learning experience for Team All Ears. Our team learned the importance of teamwork, time management, and communication, not only with our clients, but with each other. This experience has taught us the procedures of planning a completely functional product through the use of deliverables, pushed our coding capabilities to the extreme, and was able to create two applications that will help to further the cause of our sponsors - to save elephants from poaching. We are proud of the product we have made throughout the year, and we will absolutely use these experiences moving forward with our professional lives.

Appendix A: Development Environment and Toolchain

Hardware

For our hardware we all used Windows and macOS for development for all three sections of our application. No special hardware was required for our development process but the

Toolchain

These are the tools that we used to develop the mobile, web, and backend applications.

Mobile Development

Integrated Development Environment (IDE):

- Android Studio
- Visual Studio Code (VS Code)

IDE Extensions or Plugins:

- Flutter plugin/extension - integrated the IDE with Flutter
- Dart plugin/extension - Dart language support for the IDE

Outside the IDE:

- Flutter - the bundled Dart language and Flutter development framework
- Android Development Toolchain - Allows for Android development (included in Android Studio)

Web Development

IDE: VS Code

IDE Extensions:

- Vetur - Vue support for VS Code
- Prettier - allows Prettier to be used as a code formatter in VS Code

Outside the IDE:

- Node.js - a Javascript engine outside the browser
- Yarn - a package manager for Node.js
- Vue - a frontend Javascript framework
- Leaflet - a Javascript mapping framework
- Highcharts - a Javascript chart framework

- TailwindCSS - a component-based CSS framework
- Font Awesome - a web icons library
- Typescript - a strongly typed language that compiles into Javascript
- Day.js - a Javascript date library
- Lodash - a general Javascript utility library
- Pug - an indentation-based markup language that compiles to HTML
- ESLint - a javascript linter
- Prettier - a javascript code formatter
- Sass - an indentation-based styling language that compiles into CSS
- Windows Subsystem for Linux (WSL) 2 - a linux command line environment for windows.
- Git - the preferred enterprise version control software

Backend

IDE: VS Code
IDE Extensions:

- Python - Python support for VS Code
- Pylance - expanded Python support for VS Code

Outside the IDE:

- Python - a high-level programming language
- Flask - a lightweight Python web framework
- MariaDB - an open source database
- Argon2-cffi - a Python wrapper for a C implementation of the Argon 2 password hashing algorithm
- PyJWT - a Python Javascript Web Token library
- Requests - a Python HTTP client library
- Stringcase - a Python string manipulation library
- Pytest - a Python testing framework
- UWSGI - a Python WSGI server library built on C
- Yapf - a Python code formatter
- MyPy - a Python static type checker

Setup

This section will go over how our team prepared our environments and configured our machines for the development process. For this project most of the members used Windows and one member used macOS (to test iOS with Flutter). The following steps will go over how to set up for developing our web frontend, mobile frontend, and backend.

Mobile Development

1. **Install Git**
   Guide: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
2. **Install Android Studio**
   Guide: https://developer.android.com/studio/install
3. **Install VS Code (Optional)**
   Website https://code.visualstudio.com/Download
4. **Install Flutter**
   Guide: https://flutter.dev/docs/get-started/install
5. **Setup Android Studio for Flutter**
   Guide: https://flutter.dev/docs/get-started/editor
6. **Setup VS Code for Flutter (Optional)**
   Install the Dart and Flutter extensions for VS Code
7. **Clone the Repository**
   Repository URL: https://github.com/All-Ears/mobile-frontend.git
8. **Open in an IDE**
   Open the repository in either Android Studio or VS Code
9. **Congratulations!**
   You are now ready to develop the Elephant Footprint mobile application

Web Development

1. **Install Git**
   Guide: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
2. **Install Node.js**
   Website: https://nodejs.org/en/download/
3. **Install VS Code**
   Website https://code.visualstudio.com/Download
4. **Set up VS Code for Flutter**
   Install the Vetur, Prettier, Pug, and Sass extensions
5. **Install Yarn**
   Run `npm i yarn -G` in the terminal
6. **Fork the Repository**
   Create a fork of the main repository
   Repository URL: https://github.com/All-Ears/web-frontend.git
7. **Clone the Fork**
8. **Open in VS Code**
9. **Install Dependencies**
   Run `yarn` in the terminal
10. **Congratulations!**

You are now ready to develop the Elephant Footprint web application

Backend Development

1. **Install Git**
   Guide: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
2. **Install Python**
   Website: https://www.python.org/downloads/
3. **Install MariaDB Community**
   Website: https://mariadb.com/downloads/
4. **Install VS Code**
   Website https://code.visualstudio.com/Download
5. **Set up VS Code for Python**
   Install the Python and Pylance extensions
6. **Fork the Repository**
   Create a fork of the main repository
   Repository URL: https://github.com/All-Ears/backend.git
7. **Clone the Fork**
8. **Open in VS Code**
9. **Install the Venv Package**
   Run `pip install venv` in the terminal
10. **Set up the Virtual Environment**
    Run `python -m venv env` to initialize the virtual environment
11. **Install Dependencies**
    Run `pip install -r requirements.txt` in the terminal
12. **Set up MariaDB**
    Log into MariaDB as root and run the `init_schema.sql` file from the cloned repository
13. **Congratulations!**
    You are now ready to develop the Elephant Footprint backend

Production Cycle

Mobile Application

Edit

Follow the instructions for development above to open the application in Android Studio. Once you have the files open you can change any of the files in the “dev” folder(which contains , you will also want to check on the pubspec.yaml file which is where dependencies are stored.

Compile

To compile you just make sure you have a device loaded up in the AVD Manager and you hit the “Run ‘main.dart’” file. This will turn on your virtual device and allow for you to see what you have changed in the application. If you want to hot restart from an already running file you can hit the “Flutter Hot Reload” which will also update minor changes in the app much faster on the virtual device.

Deploy

To deploy the application to an Android app follow the instructions found here:
https://flutter.dev/docs/deployment/android
To deploy the application to an iOS app follow the instructions found here:
https://flutter.dev/docs/deployment/ios

Web Application

Edit

Change the files in the `src` directory. To have all changes compiled in real time run `yarn watch` in the terminal. We highly suggest running a dummy backend that will serve the contents of the generated `dist` directory. Expressjs is an easy one.

Deploy

1. Commit and push the changes to one’s fork.
2. Make a pull request to the main repository that one forked.
3. Once the pull request is merged, wait until GitHub’s build action finishes successfully.
4. log on to the Elephant Footprint server.
5. Log into the `elephantfootprint` account and go to its home directory.

6. Go to the `web-frontend` directory and run `git pull`. If you see any `.vue` files then checkout the `build` branch.

Backend

Edit

Change or add Python files. To use the Flask development server follow the instructions available in the `README.md` file.

Deploy

1. Commit and push the changes to one's fork.
2. Make a pull request to the main repository that one forked.
3. Once the pull request is merged, log on to the Elephant Footprint server.
4. Log into the `elephantfootprint` account and go to its home directory.
5. Go to the `backend` directory and run `git pull`.
6. Log out of the `elephantfootprint` account.
7. Run `systemctl restart elephant-footprint-backend.service` as root.