Software Testing Plan

All Ears

Date: 3/26/2021

Sponsors: Ms. Jenna Keany and Dr. Chris Doughty

Faculty Mentor: Mr. Tomos Prys-Jones

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

Table of Contents

1 Introduction

African forest elephants, Loxodonta Cyclotis, are a threatened subspecies of elephant 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 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 keeping track of the elephants' 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 both track the poaching of elephants using tools like the Monitoring the Illegal Killing of Elephants, or 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 African forest elephants, the prominence of African forest elephant poaching, and the effects of the species and threats to the species on the climate and the elephants' habitats. Dr. Doughty and Ms. Keany work to support large conservation groups that focus on the conservation of African forest elephants.

In an effort to assist with our sponsors' endeavors, our team is in the process of creating a web-based and mobile application, name pending, with the primary goal of educating the public. Our applications will 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 anti-poaching organizations. A carbon-calculation tool will be included so members of the public can calculate their air travel emissions of CO_2 over user-defined distances. The calculator will then suggest an amount to donate to conservation charities in order to save enough elephant years to offset these emissions. The user will be directed to a donation page that will contain links to pre-existing conservation organizations, where the donations can be made.The desired outcome of this application is that it function as a tool to assist in the the continued survival of forest elephant populations, which will lead to forest maturation and carbon sequestration. Graphical representations of data collected from MIKE will be utilized by our application to present a more easily understandable picture of the threat posed by poaching. A stretch goal of our product will be to integrate an application called Animaps to display fauna data based on geographical regions of a map when selected.

Testing is an essential part of the software development process and it ensures that all of the parts of the software work independently as well as integrated into the greater system. In our system we have individual sections of functionality that we must test for correct operation, and the sections must be integrated as they would be normally and tested again. The core units of our application are, in both the web and mobile applications: the backend database, the admin portal, the elephant map, the carbon calculator, the donation page, and the about page. Once each unit is tested and confirmed as functioning properly we will then perform integration testing. In our application the main units we must test for proper integration are: that the admin portal correctly edits the database, the elephant maps properly sends queries to the Backend database, and the carbon calculators properly sends API queries to the IATA API. The unit testing will be the most extensive testing portion of the project because the functionality of each unit is detrimental to our completed application if each piece is not fully operational; we would not be delivering the full functionality to our clients and ultimately the end users for the application.

In the rest of this document we will build up to a final test of the entire system of both our web and mobile applications in the order of smallest tests building to the largest tests. Further on, we will detail all of the testing we plan to complete during and after our implementation. We will first look at the complete unit testing on specific parts of our application. Then, we will detail the plans for the integration testing we will perform on the parts of our application as they work together as a whole. Finally, we will test the usability of the application from a standard user’s and then from an administrator's point of view.

2 Unit Testing

Unit testing is a thorough way of testing small units of code for functionality before integrating them into a final product, ensuring that small elements of a complete product are working. In this section of the document we will discuss the unit testing that we plan to perform and the sections we will be performing the testing on. In doing these unit tests we will ensure that the largest functioning parts of our applications are working correctly. We decided to only test the most functionally significant units of our software in order to ensure they perform as intended.

In our unit testing we will only be looking into specific portions of our applications with the most functionality and user interaction. We will not be looking at the donation pages or about pages of our web and mobile applications because they are both extremely simple, have very little functionality, and do not require more of the application than just their own formatted pages and external links to function. For the parts of code that we will be testing, we will first look at the Elephant Map and the Carbon Calculator for both the web and mobile applications. Then, we will look at the unit testing for our Backend database as well as our online Admin Portal. Ensuring each of these pieces works individually is imperative in making sure they all work together in the end.

For unit testing the Elephant Map, Admin Portal, and the Carbon Calculator, a dummy API will be run in tandem with the application; all requests from the Elephant Map, Admin Portal, and Carbon Calculator will be redirected to their specific dummy API to individually test the Map, Portal, and Calculator components without straying into unit testing for the backend.

For the backend unit testing, we will be utilizing the pytest framework library, which will help us to ensure that, given specific requests or inputs, the backend will respond as expected, either with success or with error.

2.1 Elephant Map Unit Testing

2.1.1 Website:

1. Hovering over individual countries without first clicking on a country in the web application will show the most recent number of elephant deaths caused by poaching in that country as recorded in the dummy API. No information regarding total elephant poaching deaths or previously recorded elephant poaching deaths will be visible while hovering.
2. When no countries have been selected, all countries will have a color from green to yellow to red indicating the severity of elephant poaching deaths as compared to the other countries in the elephant testing map, green for low, yellow for few, and red for many - relative to the highest number of elephant poaching instances in each country's most recently recorded years according to the dummy API.
3. Clicking on a specific country will show a graph of elephant deaths caused by poaching over time, starting at the first recorded instance of poaching numbers and ending at the most recently recorded year in the dummy API.
4. After having clicked on a country, hovering over the individual countries on the elephant map will show the most recent number of elephant poaching deaths recorded in that country, while still retaining the graph specific to the clicked country.
5. Once a country has been selected, that country will be greyed-out on the elephant map, indicating that it has been selected. Even though its color will no longer be shown, the other countries will be colored accordingly to indicate the severity of elephant poaching deaths as compared to other countries, including the selected country, in the elephant testing map, green for low, yellow for few, and red for many - relative to the highest number of elephant poaching instances in each country's most recently recorded years according to the dummy API. The selected country's poaching deaths will be taken into account, even though the country will no longer be colored.

2.1.2 Application:

1. When no countries have been selected, all countries will have a color from green to yellow to red indicating the severity of elephant poaching deaths as compared to the other countries in the elephant testing map, green for

low, yellow for few, and red for many - relative to the highest number of elephant poaching instances in each country's most recently recorded years according to the dummy API.

2. Clicking on a specific country will show a graph of elephant deaths caused by poaching over time, starting at the first recorded instance of poaching numbers and ending at the most recently recorded year in the dummy API.
3. Once a country has been selected, that country will be greyed-out on the elephant map, indicating that it has been selected. Even though its color will no longer be shown, the other countries will be colored accordingly to indicate the severity of elephant poaching deaths as compared to other countries, including the selected country, in the elephant testing map, green for low, yellow for few, and red for many - relative to the highest number of elephant poaching instances in each country's most recently recorded years according to the dummy API. The selected country's poaching deaths will be taken into account, even though the country will no longer be colored.

2.2 Carbon Calculator Unit Testing

2.2.1 Website:

1. From a fresh page with empty locations, press the submit button. The form will not submit and no action will be taken on the page.
2. From a fresh page, enter one location suggested by the dummy API through one of the drop-down lists and press submit. The form will not submit and no action will be taken on the page.
3. 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 will not submit and no action will be taken on the page.
4. 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 will not submit and no action will be taken on the page.

5. 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 will not submit and no action will be taken on the page.

6. 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 will not submit and no action will be taken on the page.

7. 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 will not submit and no action will be taken on the page.

8. 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 will submit and the page will receive two latitude, longitude pairs specific to the two locations. The map on the page will display a dotted line connecting the two latitude, longitude pairs in a mimicry of a flight path. A table will 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 will appear below the table indicating the amount of 'elephant years' required to sequester the amount of emitted carbon shown in the table.

9. 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 will not submit and no action will be taken on the page.

10. 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 will not submit and no action will be taken on the page.

11. 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 will return the two latitude, longitude pairs

of both locations and a new dotted line will connect the previously existing valid location to the newly entered valid location. A table will 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 will appear below the table indicating the amount of ‘elephant years’ required to sequester the amount of emitted carbon shown in the table.

12. 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 will return the two latitude, longitude pairs of both locations and a new dotted line will connect two newly-entered locations. A table will 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 will appear below the table indicating the amount of ‘elephant years’ required to sequester the amount of emitted carbon shown in the table.

2.2.2. Application:

1. From a fresh page with empty locations, press the submit button. The form will not submit and no action will be taken on the page.
2. From a fresh page, enter one location suggested by the dummy API through one of the drop-down lists and press submit. The form will not submit and no action will be taken on the page.
3. 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 will not submit and no action will be taken on the page.
4. 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 will not submit and no action will be taken on the page.
5. 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 will not submit and no action will be taken on the page.

6. 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 will not submit and no action will be taken on the page.

7. 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 will not submit and no action will be taken on the page.

8. 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 will submit and the page will receive two latitude, longitude pairs specific to the two locations. The map on the page will display a dotted line connecting the two latitude, longitude pairs in a mimicry of a flight path. A table will 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 will appear below the table indicating the amount of 'elephant years' required to sequester the amount of emitted carbon shown in the table.

9. 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 will not submit and no action will be taken on the page.

10. 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 will not submit and no action will be taken on the page.

11. 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 will return the two latitude, longitude pairs of both locations and a new dotted line will connect the previously existing valid location to the newly entered valid location. A table will 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 will appear below the table indicating the amount of 'elephant years' required to sequester the amount of emitted carbon shown in the table.

12. 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 will return the two latitude, longitude pairs of both locations and a new dotted line will connect two newly-entered locations. A table will 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 will appear below the table indicating the amount of 'elephant years' required to sequester the amount of emitted carbon shown in the table.

2.3 Backend Database Unit Testing

1. Load a CSV file into a backend API. If the file is the correct format for the backend to accept, the system will indicate success and the CSV records will be used to populate the empty dummy backend. If the loaded CSV file does not match the correct format for the backend, the system will issue an error and the backend will remain empty and unchanged.

2. Load a non-CSV file into an empty backend API. The system will issue an error and the backend will remain empty and unchanged.

3. With an empty backend, initiate an automatic upload of a CSV file. If it is formatted in a way that is considered valid by the backend API. A message will indicate that a successful upload has taken place, and the empty backend will be populated with information from the loaded CSV. If the CSV file is formatted in a way not considered valid by the backend API. The system will issue an error and the backend will remain empty and unchanged.

4. With an empty backend, initiate an automatic upload of a non-CSV file. The system will indicate that an error has occurred, and the empty backend will remain empty and unchanged.

5. Add a new record to an empty backend API. If successful, a success message will be returned by the backend and the record will be successfully added into the backend database. If unsuccessful, the system will indicate a corresponding error message and the backend will remain empty and unchanged.

6. Edit an existing record in an empty backend API. Because the backend is empty, the system will return the corresponding error and the backend will remain unchanged.

7. Remove a record from an empty backend API. Because the backend is empty, the system will return the corresponding error and the backend will remain unchanged.

8. With a populated backend, edit an existing record so that the information in the record is altered but does not duplicate a primary key of a record not being edited. If the edited record matches the format considered valid by the backend, the system will indicate success and the record will be successfully changed. If the edited record does not match the format considered valid by the backend, the system will indicate the corresponding error and the record will remain unchanged.

9. With a populated backend, edit an existing record so that the primary key of the record matches the primary key of a record not being edited. The system will indicate the corresponding error and the record will remain unchanged.

10. Add a new record to the populated backend API that does not have information that duplicates a primary key of another record. If the record is validly formatted according to the backend API, a success message will be returned by the backend and the record will be successfully added into the backend database. If the record is not validly formatted according to the backend, the system will indicate the corresponding error and the backend will remain unchanged.

11. Add a new record to the populated backend API that duplicates a primary key of another record. The system will indicate the corresponding error and the backend will remain unchanged.

12. Remove a record from the populated backend API. If the record exists, a success message will be returned by the backend and the record will be successfully removed from the backend database. If the record does not exist, the system will indicate the corresponding error and the backend will remain unchanged.

13. With a backend populated by an automatic upload with one record successfully edited in a way that changes information in the record but does not alter the primary key, initiate another identical automatic upload. The automatic upload will overwrite the edited record to match its unedited state.

14. With a backend populated by an automatic upload with one record successfully edited in a way that alters the primary key, initiate another identical automatic upload. The automatic upload will add in a record matching the unedited version of the record and ignore the edited record, leaving it in the populated backend.

15. With a backend populated by an automatic upload with one record successfully added to the records that already exist, initiate another identical automatic upload. The system will indicate success, but no records in the backend API will change.

16. With a backend populated by an automatic upload with one record successfully removed from the records, initiate another identical automatic upload. The system will indicate success, and the removed record will be re-added.

17. With a backend populated by the upload of a validly-formatted CSV with one record successfully edited in a way that alters the information in the record but does not change the primary key, initiate another upload of the same validly-formatted CSV. The upload will overwrite the edited record to match its unedited state.

18. With a backend populated by the upload of a validly-formatted CSV with one record successfully edited in a way that alters the primary key, initiate another upload of the same validly-formatted CSV. The upload will add in a record matching the unedited version of the record and ignore the edited record, leaving it in the populated backend.

19. With a backend populated by an upload of a validly-formatted CSV with one record successfully added to the records that already exist, initiate another identical upload of the same validly-formatted CSV. The system will indicate success, but no records in the backend API will change.

20. With a backend populated by an upload of a validly-formatted CSV with one record successfully removed from the records, initiate another identical upload of the same validly-formatted CSV. The system will indicate success, and the removed record will be re-added.

21. With a backend populated by an automatic upload of a validly-formatted CSV, initiate an upload of another validly-formatted CSV that matches the automatically uploaded CSV except for a single record that contains a matching primary key with a record that already exists in the backend API. The upload will overwrite the indicated record to match what exists in the newly uploaded CSV.

22. With a backend populated by an automatic upload of a validly-formatted CSV, initiate an upload of another validly-formatted CSV that matches the automatically uploaded CSV except for a single record that does not match a primary key of a record that already exists in the backend API. The upload will add the indicated record into the backend.

23. With a backend populated by an automatic upload of a validly-formatted CSV, initiate an upload of another validly-formatted CSV that matches the automatically uploaded CSV except it is missing a single record. The backend will indicate success, but the upload will not change the records already in the backend.

24. With a backend populated by the upload of a validly-formatted CSV, initiate an automatic upload of another validly-formatted CSV that matches the previously uploaded CSV except for a single record that contains a matching primary key with a record that already exists in the backend API. The automatic upload will overwrite the indicated record to match what exists in the newly uploaded CSV.

25. With a backend populated by the upload of a validly-formatted CSV, initiate an automatic upload of another validly-formatted CSV that matches the previously uploaded CSV except for a single record that does not match a primary key of a record that already exists in the backend API. The automatic upload will add the indicated record into the backend.

26. With a backend populated by the upload of a validly-formatted CSV, initiate an automatic upload of another validly-formatted CSV that matches the previously uploaded CSV except that it is missing a single record. The system will indicate success, but the records already in the backend will not be changed.

2.4 Admin Portal Unit Testing - Website frontend only

1. If the password entered equates to the password acceptable by the dummy API, the dummy API will issue a token to the requesting web session as a sign of success.
2. If the password entered does not match the password acceptable by the dummy API, the dummy API will issue an error and no token will be distributed.
3. If the requesting web session has received a token and been active for a specified amount of time according to the dummy API, the token will be rescinded and the website user will be returned to the login page.

3 Integration Testing

Integration testing focuses on the interface between the main modules and focuses on the correct interaction and data exchange between the modules. For example, those modules that allow access to the database, the elements involved in data exchange with App/Web-based interfaces, and those modules that rely on network-based communications. Integration testing is the process of combining all of the units together into one group and testing them as they work together in the group. Integration Testing is essential for ensuring that the delivered software functions as intended and all of it works together once integrated.

The majority of our integration tests are based on the exchange of data between our Backend and our two applications, as well as the communication between our applications and the APIs that we are pulling data from. There are several places where the integration can go wrong if the frontend received the wrong data from the Backend or API. Focusing our attention here will provide the most benefit for our time and guarantee the most functionality for our mobile and web applications.

We will test the Admin Portal's functionality in changing the Backend database when either deleting or updating data. We will then test the Elephant Map on both the web and mobile application that query data from the database and display it in a graphical depiction of poaching incidents. After this we will test the Carbon Calculator for its ability to send API queries to IATA Airport API for the airport names and IATA codes.

3.1 Admin Portal Integration Testing

The admin portal's integration testing will be 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.

1. Logging in the Admin Portal, updating the database with a new annual CSV file is expected to change both the elephant maps for both mobile and web applications. In the elephant map, the most prevalent change should be the scaled time, adding a year and values to the graphs for each African country. In the off chance that the change in the poaching numbers is tremendous, we should see the color of the maps change as intended

2. Logging into the Admin Portal and removing a record in the database is expected to change both the elephant maps for both mobile and web applications. In the elephant map, the change should be the scaled time of a specific country that the row was removed from, removing a year to the graph for the African country

3. Logging into the Admin Portal and creating a record in the database is expected to change both the elephant maps for both mobile and web applications. In the elephant map, the change should be the scaled time of a specific country that the row was added from, adding a year and value to the graph for the African country. In the off chance that the change in the poaching numbers is tremendous, we should see the color of the maps change as intended

3.2 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:

1. The Elephant map will use the Africa.json file to shape central Africa regions when they interact with the map. Simultaneously, the Elephant Map module will send a query request to the Backend API

2. The API should pass the queried data back to Elephant Map, and the Elephant Map module can use the returned data without any extra conversion needed. As what we expect, the data from the Backend will hold a list that displays all the requirement information for Africa countries

3. The Elephant Map module should entirely run through all the unit tests involving data from the API

The method for testing this transfer phase will require the Backend API to produce a series of varied pieces of data so that this data can be placed somewhere the Elephant Map module can locate. The Elephant Map module will then be tested for full functionality on the data provided. The expected outcome will showcase the full functionality and a functional analysis graph from scratch via the data provided.

3.3 Carbon Calculator Integration Testing

The Carbon Calculator map does not necessarily need integrated testing with other modules of the application but it will need it with its connections with the airports.json asset we have and the connection it has with the online airport API we utilize to get the airport information.

3.3.1 Airports.json

We use the airports.json file to get the necessary information to query the user input into our search dropdown box. To test its integration, we need to test if the information will show up or not at all based on the input. Before testing, note a random country's information that is on the json file and input any country of the tester's choice in the destination airport dropdown search bar:

1. In the Carbon Calculator, input the noted country's ity/location/IATA code on the origin dropdown search bar, and if the noted country's contents shows up as a suggestion, this proves that the airports.json file is integrated into the system

2. In the Carbon Calculator, input any combination of words or letters on the origin dropdown search bar and if there are no suggestions, this proves that the json file does not have the input's information whatsoever

3.3.2 Airport API

We use the airport API to get granular 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 should test the API's integration by testing the input if it will, as expected, show us the correct calculation of the distance between two points. To test:

1. By inputting and selecting two endpoints, the system will take necessary information such as the selection's name, longitude, and latitude from the API and perform actions with the information above. If a line is drawn on the map and the table below shows the accurate names and the calculated distance of the two points, we could firmly say that the API is well integrated into the system

2. By inputting and selecting two endpoints, the system will take necessary information such as the selection's name, longitude, and latitude from the

API and perform actions with the information above. If the table below the map shows “null” in the names category and zero in the distance category, this will prove that the API does not have all the necessary information for this IATA code(s)

Each of these tests may require one or two integration tests to verify. These tests may run slowly, and each test's code coverage will be fairly extensive—as it will attempt to use the integration test to generate test values beyond the unit test.

4 Usability Testing

Usability testing is the process of testing the functionality of the application from a user's point of view. This step of testing is generally performed by a potential user in order to identify areas of confusion and faulty functionality, as well as to offer suggestions for the improvement of the user experience. Since our application is mainly going to be used by the general public and potentially researchers, we need to ensure that end user usability is a very significant part of our testing. We need to make sure that our product is easy to navigate and that all of the functionality is very clear to new and returning users. If our application is unclear or difficult to use it will not educate as many people on the condition and vital importance of African forest elephants.

4.1 User Groups

When we consider usability testing, we have two user groups to consider: the general public and administrators. We will need to design tests for each of them. Members of the general public will need to be able to effectively navigate the public portions of our application. They will need to be able to operate the Elephant Map and the Carbon Calculator effectively. We will need to test younger users (ages 18-35), middle aged users (ages 35-55), and older users (ages 55+).

Administrators will need to both easily navigate the public portion of our app and use the restricted admin portal. They will need to be able change, add, and delete 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 task will test a certain portion of our application. All tasks will start with the user at the home page of the application with the exception of administrator tests after the login test. Tasks for all users will be the same for the web and mobile versions of our application. Tasks for Administrators will only be done on the web version of our application. The tasks will begin after the user has been given a short time to familiarize themselves with the app.

4.2 Tasks for All Users

4.2.1 Navigation

- Navigate to the About page
- Navigate to the Donation page
- Return to the Home page

4.2.2 Elephant Map

- Navigate to the Elephant Map page
- Select a country
- Scroll down to the graph
- Deselect the country

4.2.3 Carbon Calculator

- Navigate to the Carbon Calculator page
- Calculate the carbon emissions of a flight
- Go to the Donation Page

4.3 Tasks for Administrators

4.3.1 Login

- Navigate to the Login Page
- Login

4.3.2 Edit Records

- Navigate to the Edit Records page
- Delete three records
- Add two records
- Change three already existing records
- Save the changes

4.3.3 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

4.3.4 Automatic Update

- Navigate to the Upload Records page
- Start an automatic update
- Navigate to the Edit Records page

4.3.5 Logout

- Logout

4.4 Feedback

We will collect feedback from the users throughout this whole process. However, it will be unprompted until after the tasks. We will ask them about how easy or difficult the tasks where and whether or not their completion was intuitive. We will also solicit their opinion on the overall look and feel of the app and accept suggestions on improvement. This will help us as developers improve the app, making the user experience smoother and our app more successful.

5 Conclusion

African forest elephants help the African forests thrive by making more space for trees to grow and reduce carbon dioxide levels in the atmosphere.To help prevent these environmental engineers from being completely eliminated, the All Ears team is designing an application to educate the general public about how important African forest elephants are to the stability of the global climate, to convince our users to donate to the preservation of the elephants, and to mark the number of poaching events per region in Africa each year.

In this Software Testing Plan document, we hope to have presented a detailed outline of what and how our tests will be implemented for each of the components in our applications. The Unit Testing portion ensures that our applications' individual functionalities work as intended. Then, our Integrated Testing portion of the document details our plans to test how the major and minor modules of our application interact in an effort to ensure they establish proper communication with one another. This is primarily intended to test the communication between the Admin Portal, the Backend, and the Elephant Map, as well as the components within the Carbon Calculator Map. Finally, the Usability Testing section outlines how we intend to have members of the general public of various age groups, as well as our intended administrators, use our application and test how comfortable and easy it is to operate. This will be accomplished by assigning the participants tasks that users of their specific subcategory should be able to perform, and is ultimately intended to ensure we have met our main goal-- to educate users about the importance of African forest elephants. We hope that with the effort in which we put forth in this document, that we are closer to delivering an application that is error free and functional to our dear users, providing knowledge to the public how tremendously important African forest elephants are to the world and its ecology.